**For the**

**MEMOTECH MTX Series**

# MTX Tape to Disc Conversion Booklet

**by**

**AFW Software**

# MTX Tape to Disc Conversion Booklet

The Memotech computer range is gifted with a variety of disc systems,ie 250k,500k,1000k capacities,on 5.25" or 3.5" (1000k only) and three disc operating systems,MTX SDX Basic,MTX FDXB & SDXB3,and CP/M. As you can see this causes problems for small buisnesses as it is too expensive to cater for everyone. The tape medium is a no-go area as this suffers from loading problems,brought about by the great number of tape recorders on the market. This is why I settled for the paper medium,as it is accessible to everyone. It also has the advantage of teaching users about the system,unlike programs which do most of the work.

This booklet ,not only gives a sample of the tape to disc conversions but also trains you the user ,to convert other games not included in this booklet. I also have included a PANEL utility extension for Z80 users,however,since I wrote this booklet,I have greatly improved the PANEL utility,see Popular Computing Weekly,issue 25 & 26 (correction) vol 6.

This booklet shows you how to autosave/run 9 commercial tape games. I tried to select a wide range of the more popular software houses,as hopefully,other programs in there range will also be easily converted to disc.

I have also included a third Appendix,which deals with CP/M tape to disc conversion. And for those people starved of reading material for the Memotech,then appendix 4 is for you.

For FDX owners,who in FDXB disc basic want the full 64k to play with, should contact , UK Home Computers ,as they sell a small board,called a V-ROM. This V-ROM,contains an eprom with the SDX Disc Basic Operating System. This means that FDX Disc users can access SDX programs like Memosketch (graphics program) and EDASM (macro assembler). It also means that FDX users can run MTX 512 games from disc.

Hisoft (the old school,Greenfield,Bedford,MK45 5DE) sell a number of excellent language programs and utilities,like,PASCAL ,C,DEVPAC80,Microsoft BASIC,Fortran,etc on Memotech CPM 2.2 disc format. They also provide a disc conversion facility,which costs #10+vat.

At present Amstrad PCW programs are by far the cheapest available. A great number of the more popular CPM 2.2 programs like DBASE II are available on PCW 3" format for #100.The equivalent Memotech version costs #300. It should be possible to get Hisoft to convert this to Memotech format. To do this check to see the program to be converted is CPM 2.2 compatible and that it can run on a Televideo computer system. The software dealer will be able to tell you this from the program manual. If the program meets these conditions send the original disc plus #10+VAT and your disc type to Hisoft for conversion. The disc type is C:03 for 500k 5.25",C:07 for 1MB 5.25",and D:03 for 3.5" 1MB,all CPM.

## The Index



## Copyright Notice



## Disclaimer



## Trademarks

## CHAPTER ONE: MTX TOKENS

Have you have wondered how the Memotech stores a line of BASIC in memory? or how it keeps track of this and other lines in BASIC?

### 1.0 Introduction

Memotech BASIC,like most other forms of BASIC dialects,is made up of a series of commands or keywords,like PRINT,CLEAR,etc. The Memotech,like the ZX Spectrum,the Commodore 64 and BBC 'B' microcomputers,assign each BASIC keyword,a specific number or Token ,usually between 128 (#80) and 255 (#FF),see Table 1,see last page of this chapter. For example the token for keyword REM is 128. Note that,not all BASIC dialects use the same numbers or even the same keywords.

Long before,the high level language ,BASIC,was invented,another token type system was adopted. This is still one of the rare occasions, whereby,virtually all computer manufactures have adopted,the ASCII (American Standard Code for Information Interchange) system. See appendix 1 ,page 174 of the Memotech Manual,for the tokens and there meanings. As you will soon realise,ASCII,uses codes 0 to 127,and therefore it was logical to start the BASIC keyword token system from 128 to 255. The reason for the 0 to 255 range,is that most computers in those days,were based on 8-bit technology,and still are today.

BASIC like the majority of high level languages,is greedy on memory and in the early days when,memory was expensive,programmers developed the Token system,to alleviate these memory constraints. I will give worked examples,in order to demonstrate,how the token system works and ways of pinching a few extra bytes from the BASIC interpreter.

Please note,it would be advantageous to read up on PANEL and its commands. For more information,refer to references below and see Appendix 1.0 of this booklet.

### 1.1 Tokenising

The Memotech Operating System,MTXOS,lies between you and the BASIC Interpreter. When data is inputted via the keyboard,the information is collected together by an IN/OUT handler in the MTXOS,which distributes the information to:

a) the Visual Display Unit,VDU,or the Printer.

b) the Command Handler and/or Statement Handler, have to decide whether to insert the data into the BASIC program or to execute the line of BASIC immediately.

(1) MTX Manual,by B.Pritchard,pages 133 & 161.
(2) Memopad,vol 1,issue 4.
(3) Popular Computing Weekly,vol 6,issues 25 & 26

Distribution a) is basically a simple relay device,as it is echoing what is typed at the keyboard (input device) to the screen or printer (both output devices).

However, distribution b) has to check the incoming data against a set of guidelines. These guidelines or rules help to break down the data into components more easily recognisable by the BASIC Interpreter. There are a number of rules governing "TOKENISING".

When a line of text is typed at the keyboard,the MTXOS,checks address #FA83/4 (or 64131 ),in the system variables to see where to store it. This memory location tells the MTXOS that the keyboard buffer is located in RAM,starting at 64331. See reference (4) for more information on System Variables. From here ,the command handler sends the line to the tokeniser,so that any commands can be changed to the correct TOKEN. This involves looking through the line and replacing occurrences of keywords or their abbreviations,ie P. for PRINT,in the line by a single byte "TOKEN",with a value between 128 and 255,see table 1. Tokenising keywords has a two-fold effect:

1) it saves memory,ie GENPAT would require 6 bytes of memory but when tokenised, only uses up one byte,a saving of 5.

2) it speeds up programs considerable,because the BASIC Interpreter easily recognises the keywords and can therefore execute the command faster.

## 1.2 Tokenising & Benchmarks

A number of computer magazines,in particular PCW and BYTE,use a series of benchmarks to test the performance of the BASIC Interpreter provided in microcomputers. These benchmarks,test things like maths,FOR-NEXT loop,etc. Each test is repeated about 1000 times,so that a stopwatch can be used to time them. This repetition involves:

```
10 count=0
20 REM start
30 REM benchmark inserted here
40 let count=count+1
50 IF count<1000 then GOTO 30
60 PRINT "END,stop timing"
```

For example,BBC BASIC has an average time of ca. 14 secs over 8 benchmarks,whilst the Memotech is timed at 18 secs,which is pretty good (note that the Spectrum takes about 50 secs). The BBC BASIC Interpreter,has a more extensive TOKEN system. Not only does it tokenise keywords,but it tokenises the line numbers after GOTOs and GOSUBs. This speeds up interpretation,and can account for a few seconds over a repetition cycle of 1000 GOTOs. The Memotech Interpreter on the otherhand stores the line numbers of

(4) New Memotech Operators Manual by S.Bateson,pages 204-213.
(5) Z80 Assembly Language Programming by M.C.Moore,chapter 1.
(6) Acorn/BBC BASIC ROM user guide by M.Plumbley.

GOTOs and GOSUBs in the ASCII format. For example,GOTO 1000. The number 1000 is stored as 4 ASCII bytes rather than the tokenised form of two bytes (LSB/MSB) for any number. When the Memotech program is run,the interpreter has to decode these ASCII bytes into machine code,and this takes time.

## 1.3 Memory Reset

Before,working through a number of examples of how the Memotech stores BASIC,in memory,using techniques like tokens,the problem of clearing memory has to be discussed. The Memotech,can be RESET in three ways:

1) Switch off computer and switch on again,or the on/off approach.

2) Press the two keys either side of the space bar,or Warm Reset

3) Finally,type NEW <RET>.

Approach 1), wips RAM (Random Access Memory) totally clear,and on switch on, performs a RST #00. This resets all the system variables and loads either #4000 to #4007 or #8000 to #8007, depending on the memory configuration with the numbers 8 to 1.

Approach 2),performs the RST #00,except that none of the actual user RAM is affected apart from the first 8 bytes. This has the effect of overwriting the first line of BASIC in the program you have just reset,see later for more on this. But,the rest of the program is still resident in memory,although invisible to BASIC,because the system variables were reset.

Approach 3),only resets part of the system variables,ie the part which stores details about BASIC,and overwrites the first 8 bytes of user RAM as well.See references (4),(5) and (6) for programs which allow you to recover NEWed or RESET programs. Remember these approaches for subsequent sections.

## 1.4 Single Statement Lines

A single statement line is simply,one line of BASIC with one command or keyword. The simplest,command is probably REM,as this is the easiest to demonstrate. Note that ,the keyboard,has been configured as an input device with each character on the keyboard translated into the appropriate,ASCII code. However,the Function keys,are not catered for by the ASCII system. Memotech have mapped,codes 128 to 143,as the Function keys. As you can see from Table 1,these codes have also been assigned to keywords REM to DATA. Therefore,if you press <F1> followed by <RET>,the keyword REM should be displayed and it is.

(7) Memopad,vol 1,issues 7 & 8.
(8) Popular Computing Weekly,vol 6,issue 21,pages 22 & 23.
(9) M.O.C.,vol 3,issue 7

Now type in the BASIC listing below:

**listing 1 :**

```
10 REM
```

To see how the BASIC Interpreter stores this in RAM,invoke the built-in monitor/disassembler with PANEL <RET>. Now press D for display and enter 4000 <RET>. Note that if you have a MTX 500,then change all references of `4' to `8'. You are interested only in the memory or HEX block at the bottom of the screen. This block should now have updated from 0000 to 4000 (or 8000). The memory location and its contents should also be shown at the bottom of the screen,see figure 1.

Figure 1 : Screen dump of the HEX block of PANEL.

```
          3FF0:  not of interest
          3FF8:  not of interest
          4000:  06 00 0A 00 80 FF 08 01
          4008:  FF FF FF FF FF FF FF FF
          4010:  FF FF FF FF FF FF FF FF
          4018:  FF FF FF FF FF FF FF FF
```

```
4000  06
```

To avoid changing the contents of location 4000,as PANEL is in EDIT mode,press the <BRK> key. Instead of 4000 to 4007,containing the numbers 08 to 01 at switch on,the BASIC interpreter has used these to store line 10. As you can see,7 bytes have been used by the interpreter to store line 10. In actual fact the seventh byte,tells the interpreter where to store the next line of BASIC and isn't really part of line 10. Note that the eighth byte,01 is left over from the RST #00. The 6 bytes used to represent line 10 mean:

```
0600 = The number of bytes used to describe line 10 in HEX.
0A00 = These two bytes are the line number in HEXADECIMAL. The
       line number is stored in the LSB/MSB format,see ref (5).
       Therefore,on the Memotech,line numbers can be in the
       range 0 to 65535.
80   = This is 128 in decimal and represents the token for the
       keyword REM.
FF   = This is a marker used by the BASIC interpreter. It is used
       to indicate the end of the BASIC line.
```

If you now increase the size of the example program to listing 2:

```
10 REM
20 GOTO 1
```

The BASIC interpreter updates RAM to accommodate the new line. As you can see from figure 2,line 20 has been inserted into RAM at

#4006,after the line 10 end of line marker. As before,the first 5 bytes of the new code represents the line length,number and keyword token. This is the standard format of all new lines.

**Figure 2:** The HEX dump of listing 2.

```
4000:  06 00 0A 00 80 FF 07 00
4008:  14 00 96 31 FF 08 FF FF
4010:  FF FF FF FF FF FF FF FF
4018:  FF FF FF FF FF FF FF FF
```

But what is #31 ? If you look up page 174 of your Manual,and goto #31 or 49 decimal,you will see this represents the number ONE. If you were to change line 20,using EDIT 20 <RET>,to GOTO 100,then the new memory dump for line 20 would change

```
FROM :  07 00 14 00 96 31 FF          =  GOTO 1
TO   :  09 00 14 00 96 31 30 30 FF    =  GOTO 100
```

As mentioned earlier in section 1.2,the Memotech BASIC interpreter doesn't tokenise the line numbers after GOTOs and GOSUBs. But rather stores them as ASCII bytes. Apart from the increased memory usuage,the program speed is reduced because of ASCII decoding to machine code,which is longer the than line number token decoding to machine code. Therefore to speed up programs and save a few extra bytes,store subroutines at low line numbers,ie GOSUB 10 instead of 40000.

## 1.5 Multistatement Lines

Take the example given in listing 3,and its resulting memory data. This is a special case whereby the Memotech BASIC interpreter,only tokenises the REM keyword. The other keywords are stored as ASCII bytes. However,all other keyword combinations are stored in the tokenised format. For Example try the following:

```
10 VS 5:PAPER 4:INK 1
```

Note that reference (11),shows you how to speed up programs by at least 30%,which is a big increase.

**listing 3:**

```
10 REM:GOTO 1

memory data : 0D 00 0A 00 80  3A 47 4F 47 4F 20 31 FF => 13 bytes
              13    10    REM :  G  O  T  O  sp 1  end
```

where sp=space.

Note that if you are interested in writing your own BASIC interpreter or compiler, then reference (10) is essential reading.

(10) Writing Interactive Compilers & Interpreters,P.J.Brown
(11) Faster Basic,Popular Computing Weekly,v6,issue 30,page 23.

**Table 1:** This is a list of the main keywords as used by MTX BASIC and their corresponding token numbers and ROM addresses. Note that MTX BASIC,is stored on ROM C,on page 1. This means you have to copy ROM C to RAM,then disassemble with PANEL,see reference (12 & 13).

| TOKEN | KEYWORD | ROM Address (in decimal) |
|---|---|---|
| 128 (#80) | REM | 10154 |
| 129 (#81) | CLS | 05525 |
| 130 (#82) | ASSEM | 10116 |
| 131 (#83) | AUTO | 02567 |
| 132 (#84) | BAUD | 03384 |
| 133 (#85) | VS | 05622 |
| 134 (#86) | CONT | 10119 |
| 135 (#87) | USER | 64137 |
| 136 (#88) | CRVS | 05610 |
| 137 (#89) | CLEAR | 10235 |
| 138 (#8A) | CLOCK | 10996 |
| 139 (#8B) | ATTR | 05602 |
| 140 (#8C) | COLOUR | 05555 |
| 141 (#8D) | INK | 05507 |
| 142 (#8E) | CSR | 05507 |
| 143 (#8F) | DATA | 10154 |
| 144 (#90) | PRINT | 11144 |
| 145 (#91) | DIM | 10295 |
| 146 (#92) | ADJSPR | 05563 |
| 148 (#94) | NEXT | 10865 |
| 149 (#95) | FOR | 10343 |
| 150 (#96) | GOTO | 10383 |
| 151 (#97) | GOSUB | 10396 |
| 152 (#98) | INPUT | 10460 |
| 153 (#99) | IF | 10593 |
| 156 (#9C) | LET | 10714 |
| 158 (#9E) | LOAD | 10990 |
| 159 (#9F) | LPRINT | 10870 |
| 163 (#A3) | NEW | 00517 |
| 164 (#A4) | PAPER | 05514 |
| 165 (#A5) | NODDY | 11076 |
| 166 (#A6) | ON | 11087 |
| 167 (#A7) | OUT | 02557 |
| 169 (#A9) | PANEL | 11256 |
| 170 (#AA) | GENPAT | 05539 |
| 172 (#AC) | PHI | 05642 |
| 173 (#AD) | POKE | 11266 |
| 174 (#AE) | RAND | 11273 |
| 175 (#AF) | RETURN | 11281 |
| 176 (#B0) | READ | 11380 |
| 177 (#B1) | VIEW | 05594 |
| 178 (#B2) | RESTORE | 11404 |
| 180 (#B4) | RUN | 11439 |
| 181 (#B5) | SAVE | 11011 |
| 194 (#C2) | CODE | |

(12) Memotech Memory,YOUR COMPUTER,March 1984,page 99.
(13) Memory Map,Memopad,vol 1,issue 1,page 20.

## CHAPTER TWO: MTX DISGUISE

In this chapter,the discussion centres on the four criteria for copy protection.

### 2.0 Disable the Break Key

This can easily be done from BASIC: `POKE 64862,13`

Disabling the break key is just as simple in Z80 assembly language:

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

Note that INTFFF is a system variable,used by the MTXOS,to keep track of a number of specific tasks. INTFFF is an one byte memory location at #FD5E or 64862. Every 125th of a second,the Z80 interrupts the BASIC Interpreter,checks memory location #FD5E,to see if any routine or USER specified routines are to be executed before returning to the BASIC interpreter. The INTFFF byte is responsible for the following checks:

```
BIT 0: Routine for continuous sound.
    1: Break key enabled/disabled.
    2: Keyboard auto-repeat enabled/disabled.
    3: Cursor flash & Sprite movement enabled/disabled.
    4: bits 4-6,are used to indicate that the keyword USER has
    5: been enabled/disabled.
    6:
    7: The clock bit is toggled every 125th of a second.
```

The only bit of interest to you at the moment is bit 1,but for a fuller description refer to references (4,5,14). On the Memotech,the break key,<BRK>,is disabled by resetting bit 1 to zero. Note that,the other 3 lower bits are enabled,and bits 4-6 are disabled as you haven't defined a USER command yet. Remember,bit 7,toggles on and off every 125th of a second on the Memotech.

```
BIT     :  0  1  2  3  4  5  6  7
VALUE   :  1  0  1  1  0  0  0  0/1    = 13 or 141
```

When disabling the <BRK> key,it is easier to remember 13. When writing programs,it is more convenient to include the <BRK> key disable at the start of the program code or to include it in a BASIC header,as used by Continental Software.

(14) Memopad,vol 1,issue 2,page 40.

## 2.1 Invisible BASIC

### 2.1.0 Introduction

On the ZX Spectrum,some companies used the following one liner to try and disguise the BASIC loader,which contained details of where the machine code was loaded and where it auto-ran from:

```
10 INK 7:PAPER 7:BORDER 7
```

This had the effect of producing a blank white screen when listed,as the INK and PAPER colours were the same,simple and sometimes effective. Nowadays,most up and coming HACKERS,pounch on protection techniques like these. However,the approach used by some Memotech software houses is much more igneous. Listing 4 below,demonstrates this approach.

**Listing 4:-**

```
1 CODE

       LD HL,#FAA4  ;point to TOP of NODDY,system varaible.
       LD DE,#FAA5  ;where to start copying to.
       LD BC,#000C  ;11 bytes to be cleared.
       XOR A        ;clear register A.
       LD (HL),A    ;reset system variable.
       LDIR         ;reset the next 11 variables as well.
       LD HL,#FA9E  ;point HL,to start of FEXPAND.
       LD (HL),#C7  ;jump to #0000,before defaulting to PANEL.
       RET          ;Return to BASIC.
```

Switch the computer off then on again,then type in listing 4. Once this has been typed in successfully,enter PANEL <RET>. Press D 4000 <RET> (or 8000 on a MTX 500),followed by <BRK>,to avoid overwritting the code at #4000. If you now list the code just entered,ie press L 4007 <RET>,the listing above should be displayed at the top left hand side of the PANEL screen. Note that the code between #4000 and #4006 is the TOKEN,line number and length,see chapter 0ne.

To understand how this short routine works requires a brief understanding of how the BASIC Interpreter keeps track of the BASIC program in memory and a brief description of the Memory Map.

### 2.1.1 The Memotech Memory Map

The Memotech memory is mapped as follows:

ROM-A or the Monitor ROM contains the operating system and the default system variables,and all RST commands are held in ROM-A. The RST #10,is used to access the extensive graphics commands.

RST #28,handles number crunching,etc. At startup,ROM-A is loaded into RAM at #0000-#1FFF.

ROM-B,or the Front PANEL ROM contains all the code for the built-in Z80 assembler,disassembler,monitor and handles all the graphic/RST 10 commands which start at #2E71 in RAM. This is loaded into RAM at #2000-#3FFF.

ROM-C,or the BASIC ROM contains the code to check program syntax,BASIC,etc. The Memotech,uses a ROM paging system. This system,reserves RAM,#0000-#1FFF for the Monitor ROM-A,on all available 8 pages,see references (12,15 & 16). However,RAM #2000-#3FFF,is used for other ROMs,like BASIC,PASCAL,SDX BASIC,etc. What happens is that you invoke the ROM you want with the ROM n command,where n=0 to 7. This initialises the ROM and loads it into RAM at #2000. Whenever,BASIC,is run,ROM-C and ROM-B,are interchanging where necessary,to run the program.

RAM is also paged. Of which BASIC grabs RAM at #C000-#FA51 as workspace for calculations,arrays,variables,etc,see reference (11). Please note that any code stored here will be overwritten,only when you are using BASIC . RAM at #FA52 through to #FFFF is used to store all the system variables. The default system variables are stored on ROM-A,and relocated in high RAM,at startup.

RAM from #4000 to #BFFF,is reserved for the user. This part of RAM is paged like #2000-#3FFF,except that 16 pages of 32k are available to the user or 512k of RAM. The MTXOS and the BASIC Interpreter,don't invade this part of RAM as it is reserved for BASIC or Assembly language programs.See references (12,15 & 16) for a graphical approach to memory.

### 2.1.2 Keeping Track of BASIC

Returning to the example in listing 4;enter PANEL <RET> and press D FAA4 <RET>,followed by <BRK>. The information located between #FAA4 and #FAAF,12 bytes in all,are used by the BASIC Interpreter to keep track of the current BASIC program. Figure 3,gives a brief description of the 12 program status variables.

**Figure 3:** Twelve program status variables for keeping track of BASIC within RAM. Note that the highlighted memory locations are stored in LSB/MSB format.

```
FAA4 & FAA5  :   1B 40   : Keeps track of  the TOP of NODDY.
FAA6         :   00      : Number of NODDY pages used in program.
FAA7 & FAA8  :   1B 40   : Keeps track of the TOP of BASIC.
FAA9         :   00      : Number of BASIC pages used in program.
FAAA & FAAB  :   00 40   : Start of User RAM on MTX 512. On a
                           MTX 500,User RAM starts at #8000.
FAAC to FAAF :1B 40 00 00:Stores other information about BASIC,
```

(15) Memotech Manual by B.Pritchard,pages 245-247.
(16) New Memotech Manual by S.Bateson,page 244.

References (7,8 & 9),show you how to use this information to RECOVER accidentally erased BASIC listings,very useful.

Whenever you enter a line of BASIC,the MTXOS stores this in the keyboard buffer. The BASIC interpreter,takes this text,tokenises it and if it has a line number,it inserts the BASIC text into memory at the desired location. The BASIC line can be inserted at one of three locations:

1) START of memory.
2) TOP of BASIC.
3) Somewhere between 1) and 2).

If you are just starting a BASIC program,then the TOP of BASIC and the start of User RAM (Bottom of BASIC) will be the same. If this is the case,the BASIC interpreter will load the details of the BASIC line (see chapter one) from #4000 onwards and update the 12 memory locations above.

If the line number of the BASIC listing is higher than the line number of the code at the TOP of BASIC,then the new line will be stored there and the TOP of BASIC will be increased accordingly.

If the line lies somewhere in the middle of the program,then using a special algorithm,the Interpreter will find the correct location in memory. Once this has been located,the code above this address is moved up by the length of the new line,then the new line is inserted between the old position and the moved position and the TOP of BASIC is updated.

Note that if you decided to include a couple of pages of NODDY,then the variables used to keep track of NODDY,would be updated accordingly.

### 2.1.3 The Example

Returning to the example in listing 4. Listing 4,does two things:it fools and Blinds the BASIC Interpreter. It does this by resetting all twelve program status variables at #FAA4. This resetting has the effect of redefining the new start of User RAM,at #0000. So when the code performs the RST #00,the program in memory isn't lost as expected by performing a normal RESET but hides itself before auto-saving and runing. This makes the program unlistable and unrecoverable as it is invisible to the BASIC Interpreter. Rather cunning eh!

## 2.2 Auto-Run Programs

This is very easy to perform as shown in **listing 5:**

```
10 SAVE "PROG" or USER SAVE "PROG.BAS" or DISC SAVE "PROG.BAS"
20 the rest of the program.
```

When this program is RUN <RET>,the BASIC at line 10 saves the code to TAPE,or SDX DISC BASIC or Memotech CP/M DISC BASIC. Once the whole program has been saved to TAPE or DISC,the program status variables are already pointing at line 20,which is executed by the BASIC Interpreter. The same happens when you reload the program. When the program has loaded,the status variables are already pointing at line 20,which is executed by the BASIC Interpreter immediately. Thus line 10 has completely been bypassed and the actual program has AUTO-RAN. Simple.

## 2.3 MTX 500 or MTX 512

The MTXOS loads the system variable at #FA7A or 64122,with the RAM configuration. For a MTX500,this location is ZERO and on a MTX512 system it is ONE. Therefore to fool the MTXOS into thinking it is a MTX500,enter:

```
POKE 64122,0 <RET>
NEW <RET>
```

This resets the RAM configuration to MTX 500 mode. Note that if you press the RESET keys,the MTXOS will reload the real configuration at #FA7A,so you'll have to change it again,only after switch on and RESET.

The reason this works is simply because of the common RAM layout of both the MTX 512 and 500 between #8000 and #FFFF,see section 2.1.1,memory map earlier. One of the more commonly used techniques for deciding which machine is in use is to check the system variable at #FA7A,and move up or down according. Note that most code is loaded at #8000 first and then moved down,if test says MTX 512 in use. Listing 6,shows how this is done:

**Listing 6:-**

```
100 CODE

        DI              ;disable BASIC interrupts.
*       LD SP,(#FA96)   ;save the start of machine stack.
*       LD HL,#FD4F     ;this is the USRRST variable which is
                        ;called everytime there is an interrupt.
                        ;If the variable is occupied,the code at
                        ;location pointed to by the variable USRRST
                        ;is automatically executed.
*       LD DE,START     ;this is pointing to the start of the code.
*       LD (HL),E       ;now store the start of the program code
*       INC HL          ;at the USRRST variable. In other words
*       LD (HL),D       ;after every interrupt call program.
        LD A,(#FA7A)    ;test RAM configuration.
        OR A            ;this tests if MTX 500 or MTX 512.
        JR Z,START      ;if zero,then MTX 500,don't move code.
```

```
        LD DE,END       ;else move the code
        LD HL,#4000     ;to start of MTX 512 User RAM.
        LD BC,LENGTH    ;length of the code to move.
        LDDR            ;move the code down now.
        JP START        ;goto code start at #4000 START.
START:  start of program like TOADO.
```

Note that the * lines are not always used.

## 2.4 Summary

Therefore to copy protect a program,just follow this simply procedure:-

```
1   CODE             ;basic invisible code
10  SAVE "PROG"      ;save program to TAPE or DISC.
100 CODE             ;check MTX 500 or 512 then program start.
```

If you now type RUN <RET>,then the above program will hide itself,save to TAPE or DISC (if you insert USER or DISC in front of SAVE),and auto-run when reload.

## CHAPTER THREE: MTX RECOVER

The concepts introduced in the previous chapters will now be used to deprotect a number of commercially available tape games to run from disc.

### 3.0 Introduction

This section has been written for users with a SDX Disc Basic Controller and any disc drive type. It is also applicable to FDX and FDX CP/M owners with a V-ROM attached to the cartridge port. The V-ROM runs the SDX Disc Basic Rom on a FDX unit. This will cover a great many Memotech users. However,owners of FDX units without the V-ROM and SDX CP/M owners will require to read appendix 3.0.

Where possible,all conversions were carried out in MTX 500 mode. To mimic a MTX 500,type POKE 64122,0:NEW <RET>. This chapter will use TOADO as a worked example as this comes free with the Memotech range.

### 3.1 System Setup

Switch on computer system. MTX 512 and RS 128 owners change to MTX 500 mode. Initialise the disc system using ROM 3 <RET>. Now load in TOADO, LOAD "" <RET>.

The program should have loaded successfully by now,if not,rewind tape and try a different volume setting or try the copy on side 2 of the tape. Once TOADO has loaded and run successfully,press the RESET keys,either side of the space bar,thus presumably erasing the program. At this stage load in the PANEL extension into the area usually reserved for BASIC workspace.

```
USER READ "PANEL.COD",54524        <RET>
RAND USR(54524)                    <RET>
CLEAR                              <RET>
```

This will load the PANEL dump utility from disc,and relocate itself in high RAM. Now,swap this diskette with a blank formatted diskette,and press <CTRL> C together to initialise the new disc.

### 3.2 What to Look for

At this stage,enter PANEL <RET> and press D 8000 <RET>,then press <BRK>,so as not to alter the contents of memory. Then type L 8007 <RET>.,see figure 4 for screen dump.

Note that MTX 500 owners only will find that 8000 to 8007 isn't as below but 08 07 06 05 04 03 02 01. This is due to RESET,see MTX Tokens chapter.

**Figure 4:** PANEL Screen Dump of TOADO from #8000 to #8028.

```
8007        LD HL,#FAA4
800A        LD DE,#FAA5     AF 0000 F3
800D        LD BC,#000C     BC 0000 F3
8010        XOR A           DE 0000 F3
8011        LD (HL),A       HL 0000 F3
8012        LDIR            IX 0000 F3
8015        LD HL,#FA9E     IY 0000 F3
8018        LD (HL),#C7     SP 0000 F3
801A        RET             PC 0000 F3
801B
801?

DI

    8000: 1B 00 01 00 C2 13 00 21
    8008: A4 FA 11 A5 FA 01 0C 00
    8010: AF 77 ED 80 21 9F FA 36
    8018: C7 C9 FF 0D 00 0A 00 B5
    8020: 22 54 4F 41 44 4F 22 FF
    8028: 10 20 64 00 C2 08 20 F3
```

#8000 to #8006,gives all the necessary information about the first line of BASIC in TOADO,ie it requires 27 bytes,its line 1 and the keyword is CODE. This piece of code is the invisible code,see page 10,listing 4. Line 1 ends at #801B as signified to the BASIC interpreter by the use of the end marker,FF.

At this stage toggle the I key. Memory loactions #8020 to #8026,will show the program name " T O A D O ". The keyword and line number for this section of code is SAVE and line 10. Thus the following line structure can be built up: 10 SAVE "TOADO".

Now comes the important part. The first two bytes of the next section of memory are crucial for disc conversion. As this contains the length of the actual program code. The next 5 bytes tell you that its BASIC line 100 and the keyword is CODE.

Now list #802F,see listing 6,page 13. This is the start of TOADO. TOADO checks #FA7A,to see what machine configuration is in use. This part of the code has to be disabled,so that the disc version loads TOADO at #8000,no matter the RAM configuration.

This is done by changing two memory locations only. What is needed is to replace the OR A,with NOP and set the program start to #8000+ and not #4000+. The OR A command,test the RAM configuration and if ZERO;then a flag is set. Depending on the condition of the flag (ie SET or NOT SET) depends on where the program is loaded. The NOP or no operation command erases the RAM test. This is changed as follows:

```
PANEL          <RET>
D 8040         <RET>
00             <RET>  ;select NOP.
18             <RET>  ;this changes a JR Z,Start to a JR Start.
.              <RET>  ;thus avoiding the zero flag test.
B
Y
```

## 3.3 Disc Conversion

At last,you are able to transfer TOADO to disc. This is a simple task. Find the length of the actual TOADO code,ie #2010 or 8208,and convert this and where it is found in memory (#8028 or 32808) to decimal,see highlighted values above. Now save this to disc as a byte file:

```
USER WRITE "TOADO.COD",32808,8208     <RET>
```

You can reload this byte file using USER READ "TOADO.COD",32808 <RET> followed by RAND USR(32815) <RET>. However,you will soon find as the number of conversions increase the more likely you are to mix up the starting addresses or even lose them. A better way is to use a BASIC loader to do all this for you and make life a lot simpler.

The main drawback of a BASIC loader is that it requires some memory at #8000. This is okay for programs which don't start at #8000 but can cause problems for programs which do,ie MANIC MINER,where you will have to remember the starting address. The BASIC loader,used herein,reloads the byte file 1k higher up in memory and using a 18 byte machine code routine,moves this code back to its original position, thus overwriting the disc load commands and auto-starting it. This method is very effective,see listing 7.

**Listing 7:** TOADO BASIC header for auto-run Disc program.

```
0 CODE

#8007  LD HL,#8410  ;load TOADO 1k higher up in memory.
       LD DE,#8028  ;then move it to its right place.
       LD BC,#2010  ;8208 bytes to be moved.
       LDIR         ;move it.
       JP #802F     ;Auto-Run it.

10 USER SAVE "TOADO.BAS"
20 USER READ "TOADO.COD",33808,8208
30 RUN
```

Once you have checked the program thoroughly,insert the same diskette that holds the TOADO.COD. Now type GOTO 10 <RET>. This will save the BASIC header to disc as TOADO.BAS. Once this has saved to disc,the header reloads the byte file at 33808,and executes the RUN command. This clears all variables and moves the memory pointer back to line 0. This moves the code at 33808 down to 32808 thus overwriting lines 10 to 30. It then jumps to the start of TOADO and executes it. If TOADO runs successfully,then you have followed the method okay,else redo.

Turn to Appendix 2.0 for other worked examples.

3.4 Finally - Other Tecniques

Other programs use other techniques but they are in the same theme. A common practice is to load in a BASIC header like:

.....BASIC program for screen header,followed by:-

```
100  CODE

LD HL,start of program code in RAM.
LD DE,length of this code.
LD A,1        ;this sets the built-in tape functions to LOAD.
LD (#FD68),A
CALL #0AAE
JP START      ;run the program.

200 SAVE "prog"
210 RUN
```

Some programmers use JP #0AAE instead of CALL #0AAE,in order to try and diguise where the program code starts from,eg MANIC MINER.

Other programmers use HOOKS to catch out hackers. A hook is an address which has been preloaded,via a header or at the start of the program,with a particular value. This location is accessed by the program to see if a LEGAL copy,else it crashes. This HOOK is usually located somewhere in the system variables which is erased after a every RESET,so as to hide its true identity to a hacker. This is an effective deterrent,because you are unable to recover this flag or hook which is necessary for disc conversion.

However,if someone could devise a MAGIC button as on the ZX Spectrum then disc conversion would be easy. This button has the effect of freezing the program. The area of RAM not used to store the ROM is then dumped to the disc drive. The program is saved at the current stage of play and when reloaded is already up and running.

There are many other variations on the above and its upto you to investigate. But before I finish,I'll give you one more tip. All continental s/ware programs with the screen display header,can be rerun after a RESET using RAND USR(33014).

**Appendix 1.0: PANEL UTILITY**

This short assembly language program allows the front PANEL monitor screen to be dumped to any EPSON compatible printer. PANEL can be used to disassemble the Memotech ROM or to HACK into commercially available programs or aid in the development of your own assembly language programs. The value of a hardcopy of the PANEL screen is unlimitedless and also makes the code portable. Portability means that you can work on the disassembly at lunch time or on the bus,train,etc. I have used this utility,to obtain hardcopy of the key addresses and lengths of commercially available code. For more advanced Z80 programmers,a customised PANEL COPY utility can be found in reference (3).

The following PANEL utility,should be typed in as written and saved to disc as USER SAVE "PANELCOD.INF" (for SDX users) and DISC SAVE "PANELCOD.INF" (for any CP/M disc users).

**Listing 8**:- PANEL utility

```
0 CODE

#8007  JPANEL: LD A,#C3        ;SET FEXPAND TO JP
               LD (#FA9E),A    ;AND SAVE JP CODE AT THIS ADDRESS.
               LD HL,#D507     ;START OF PANEL CODE IN HIGH RAM.
               LD (#FA9F),HL   ;SAVE THIS START AT FEXPAND+1 & +2.
       PSTART: CP "P"          ;IS PANEL DUMP SELECTED.
               RET NZ          ;RETURN TO BASIC INTERPRETER.
       PSCRN:  LD HL,(#D53A)   ;START OF SCREEN AT THIS LOCATION.
               PUSH AF         ;SAVE AF
               LD A,L          ;SEND POSITION ON THE SCREEN.
               OUT (2),A       ;TO THE VDP GRAPHICS CONTROLLER.
               LD A,H          ;THIS INCREMENTS AUTOMATICALLY.
               AND #3F         ;THIS SELECTS READ SCREEN.NB: OR #40
               OUT (2),A       ;SELECTS WRITE TO SCREEN.
               POP AF          ;RESTORE AF
               LD HL,#D534     ;POINT TO LINEFEED DATA.
               LD C,6          ;SET IT AND SET THE COL WIDTH AS
       POUT:   LD B,(HL)       ;WELL. SEND THIS DATA VIA REGISTER B
               CALL #0CE3      ;TO THE BUILT IN PRINT ASCII DUMP.
               INC HL          ;MOVE THROUGH THE LF AND CW CONTROL
               DEC C           ;CODES AND KEEP TRACK OF THE DATA
               JR NZ,POUT      ;LEFT TO SEND.
               LD DE,(#D53C)   ;NUMBER OF SCREEN POSITIONS TO SEND
      PSEND:   IN A,(1)        ;TO THE PRINTER. START READING THE
               LD B,A          ;SCREEN AND SEND IT AS AN ASCII
               CALL #0CE3      ;CODE TO THE PRINTER.
               DEC DE          ;DECREASE POSITION COUNTER
               LD A,D          ;UNTIL ZERO.
               OR E            ;
               JR NZ,PSEND     ;
               RET             ;FINISH PRINTING,RETURN TO PANEL.
      LF:      DB 27,"A",12    ;LINEFEED DATA.(#D534)
      SCRNST:  DW 7168         ;START OF PANEL SCREEN IN
                               ;VRAM.(#D53A)
      LENPAN:  DW 960          ;LENGTH OF THIS SCREEN.(#D53C)
```

```
10 CODE

      MOVECODE:LD HL,#8007    ;START OF PANEL PROGRAM ABOVE.
               LD DE,#D4FC    ;NEW START OF PROGRAM IN HIGH RAM.
               LD BC,66       ;ONLY 66 BYTES TO MOVE.
               LDIR           ;MOVE THEM
               RET            ;RETURN TO BASIC.

20  RETURN
100 GOSUB 10
110 USER WRITE "PANEL.COD",54524,66  or DISC WRITE "PANEL.COD",ETC
120 REM RAND USR(54524) TO INITIALISE PROGRAM IN HIGH RAM
130 REM ONCE INITIALISED,PRESS P WHILST IN PANEL FOR A SCREEN
140 REM DUMP.
```

INSTRUCTIONS:-

You should now have a BASIC information file on disc as "PANEL.INF". This type of file is useful for editing mistakes or adding extensions or improvements to the original program. Reload the PANEL.INF file and type GOTO 100 <RET>. This initialises the PANEL expansion system variable at FEXPAND. It then moves the whole of the line 0 code to high RAM. This code is then saved to disc as a byte file.

The advantage of a byte file over a normal BASIC file is that it doesn't reset any system variables or erase the program in use at present; unless you load the byte file in a part of RAM used by BASIC or another assembly language routine. It is therefore transparent to the BASIC interpreter. It can be accessed at any time using RAND USR(program start).

This PANEL dump utility is loaded from disc as follows:

```
USER (or DISC) READ "PANEL.COD",54524    <RET>
RAND USR(54524)                          <RET>
```

This will load the byte file at #D4FC or 54524 and will initialise it. Once initialised this program is added to the list of PANEL commands,like D for Display. You access the printer dump routine by switching the printer on and pressing P.

USES:

The byte file is loaded below the SDX disc rom,which is found at #D700. This means you can use this utility to print the disassembly of anything in RAM or ROM. However,as the utility is loaded into an area of RAM normally reserved for BASIC workspace,it is likely to be overwritten by any BASIC you type as a program. But for HACKING,this utility is situated high up in RAM to avoid commercial program code.

Load the program to be hacked in the usual way,ie LOAD "". Then press the RESET keys and reload PANEL.COD from disc. Note that this is only applicable to owners of SDX disc Basic and FDX with V-ROM systems. Once loaded,and initialised using RAND USR(54524) ,it will remain there until switched off. Remember that if you RESET the machine:- to restart the utility. Now its up to you to browse through the program at your leisure,copying the key sections to the printer.

**Appendix 2.0: WORKED EXAMPLES for SDX & FDX (with V-ROM) Users**

Please note that some of the continental software titles,like TOADO are being redistributed by OXFORD Data. You can tell the newer versions as the opening screen or title screen has the Oxford data logo. If this is the case,you will have to change the length ofthe program code to be saved. I found with the TOADO version that the newer version is longer,ie increases from #2010 (8208) to #2039. So beware.

a) TOADO

```
POKE 64122,0                                    <RET>
NEW                                             <RET>
LOAD ""                                         <RET>
press the RESET keys,when loaded                <RET>
PANEL                                           <RET>
D 8040                                          <RET>
00                                              <RET>
18                                              <RET>
.
B
Y

USER WRITE "TOADO.COD",32808,8208               <RET>
```

Now type in the following BASIC header to auto-run the program code:

```
0 CODE

     LD HL,#8410
     LD DE,#8028
     LD BC.#2010
     LDIR
     JP #802F

10 USER SAVE "TOADO.BAS"
20 USER READ "TOADO.COD",33808,8208
30 RUN
```

Now type GOTO 10 <RET>. This will save the BASIC header to disc and then it will reload and run TOADO.COD. Therefore,everytime you type USER LOAD "TOADO.BAS" <RET>,the program code of TOADO is loaded from disc and automatically run.

b) KILOPEDE

```
POKE 64122,0                                    <RET>
NEW                                             <RET>
LOAD ""                                         <RET>
```

```
press the RESET keys,when loaded            <RET>
PANEL                                       <RET>
D 8043                                      <RET>
00                                          <RET>
C3                                          <RET>
.
B
Y

USER WRITE "KILOPEDE.COD",32811,5392           <RET>
```

Now type in the following BASIC header to auto-run the program code:

```
0 CODE

     LD HL,#8413
     LD DE,#802B
     LD BC.#1510
     LDIR
     JP #8032

10 USER SAVE "KILOPEDE.BAS"
20 USER READ "KILOPEDE.COD",33811,5392
30 RUN
```

Now type GOTO 10 <RET>. This will save the BASIC header to disc and then it will reload and run KILOPEDE.COD. Therefore,everytime you type USER LOAD "KILOPEDE.BAS" <RET>,the program code of KILOPEDE is loaded from disc and automatically run.

c) QOGO

```
POKE 64122,0                                <RET>
NEW                                         <RET>
LOAD ""                                     <RET>
press the RESET keys,when loaded            <RET>
PANEL                                       <RET>
D 806F                                      <RET>
00                                          <RET>
C3                                          <RET>
.
B
Y

USER WRITE "QOGO.COD",32864,5754            <RET>
```

Now type in the following BASIC header to auto-run the program code. NOTE THAT: there is enough room below the start of the QOGO code without having to resort to moving code about.

```
10 USER READ "QOGO.COD",32864
20 RAND USR(32893)
30 USER SAVE "QOGO.BAS"
40 RUN
```

Now type GOTO 10 <RET>. This will save the BASIC header to disc and then it will reload and run QOGO.COD. Therefore,everytime you type USER LOAD "QOGO.BAS" <RET>,the program code of QOGO is loaded from disc and automatically run.

d) DRAUGHTS

```
POKE 64122,0                                    <RET>
NEW                                             <RET>
LOAD ""                                         <RET>
press the RESET keys,when loaded                <RET>
PANEL                                           <RET>
D 802C                                          <RET>
00                                              <RET>
18                                              <RET>
.
B
Y

USER WRITE "DRAUGHTS.COD",32788,9614            <RET>
```

Now type in the following BASIC header to auto-run the program code:

```
0 CODE

     LD HL,#8403
     LD DE,#801B
     LD BC.#2587
     LDIR
     JP #801B

10 USER SAVE "DRAUGHTS.BAS"
20 USER READ "DRAUGHTS.COD",33788,9614
30 RUN
```

Now type GOTO 10 <RET>. This will save the BASIC header to disc and then it will reload and run DRAUGHTS.COD. Therefore,everytime you type USER LOAD "DRAUGHTS.BAS" <RET>,the program code of DRAUGHTS is loaded from disc and automatically run.

e) QUAZZIA

```
POKE 64122,0                                    <RET>
NEW                                             <RET>
LOAD ""                                         <RET>
press the RESET keys,when loaded                <RET>
PANEL                                           <RET>
D 807D                                          <RET>
00                                              <RET>
18                                              <RET>
```

.
B
Y

USER WRITE "QUAZZIA.COD",32878,20370 <RET>

Now type in the following BASIC header to auto-run the program code:

```
0 CODE

     JP 32885

10 USER SAVE "QUAZZIA.BAS"
20 USER READ "QUAZZIA.COD",32878
30 RUN
```

Now type GOTO 10 <RET>. This will save the BASIC header to disc and then it will reload and run QUAZZIA.COD. Therefore,everytime you type USER LOAD "QUAZZIA.BAS" <RET>,the program code of QUAZZIA is loaded from disc and automatically run.

f) FIREHOUSE FREDDIE

In this example,BASIC is used as part of the program. It is therefore possible to reload FIREHOUSE FREDDIE and break into the program without losing it with RESET. SET your computer to MTX 500 mode and start the LOAD procedure,ie LOAD "FRANTIC.500" <RET>. You will have to be alert here,as you are required to press the <BRK> key just as the program stops loading,ie at the instance the load sound stops. If you have done this properly then the whole program should still be intact,ie you should be able to list it,without lots of ????? or @@@@@@@.

If you have been successful,type CLEAR <RET>. The edit line 498 with EDIT 498 <RET>,and insert USER infront of SAVE.

Now type CLEAR:GOTO 495 <RET> to save the program to disc.

Note MTX 512 owners are required to select MTX 500 mode before reloading as:

USER LOAD "FRANTIC.500" <RET>

**FOR MTX512 OWNERS ONLY**

g) SEPULCRI

```
LOAD ""                                    <RET>
press the RESET keys,when loaded           <RET>

USER WRITE "SEPULCRI.COD",16477,23291      <RET>
```

Now type in the following BASIC header to auto-run the program code:

```
0 CODE

     JP 16484

10 USER SAVE "SEPULCRI.BAS"
20 USER READ "SEPULCRI.COD",16477,23291
30 RUN
```

Now type GOTO 10 <RET>. This will save the BASIC header to disc and then it will reload and run SEPULCRI.COD. Therefore,everytime you type USER LOAD "SEPULCRI.BAS" <RET>,the program code of SEPULCRI is loaded from disc and automatically run.

h) AGROVATOR

```
LOAD ""                                    <RET>
press the RESET keys

USER WRITE "AGRO.COD",16401,32332          <RET>
```

Now type in the following BASIC header to auto-run the program code:

```
0 CODE

     LD HL,#40E0
     LD DE,#4018
     LD BC.#7E45
     LDIR
     JP #4018

10 USER SAVE "AGRO.BAS"
20 USER READ "AGRO.COD",16601,32332
30 RUN
```

Now type GOTO 10 <RET>. This will save the BASIC header to disc and then it will reload and run AGRO.COD. Therefore,everytime you type USER LOAD "AGRO.BAS" <RET>,the program code of AGRO is loaded from disc and automatically run.

i) MURDER AT THE MANOR

```
LOAD ""                                          <RET>
press the RESET keys,when loaded                 <RET>
```

Now type in the following BASIC header to auto-run the program code:

```
10 USER READ "MURDER.COD",32768
20 RAND USR(32775)
30 POKE 64853,0:POKE 64854,0
40 USER WRITE "MURDER.COD",32768,16759
50 USER SAVE "MURDER,BAS"
60 RUN
```

Now type GOTO 30 <RET>. This will save the BASIC header to disc and then it will reload and run MURDER.COD. Therefore,everytime you type USER LOAD "MURDER.BAS" <RET>,the program code of MURDER is loaded from disc and automatically run.

Note that this program is loaded at #8000,and doesn't allow you to insert a BASIC header before the program code.

## Appendix 3.0: WORKED EXAMPLES for CP/M Users

The Utility Programs

First of all type in listings A & B,save both to tape.

**Listing A:** Enter CP/M,then A>MTX,to return to MTX mode. Now enter the following listing:

```
10 REM **SAVECODE utility**
20 CODE

8022  LD HL,32768                ;Insert start of code.
8025  LD DE,00000                ;Insert length of code.
8028  LD A,0
802A  LD (#FD67),A
802D  LD A,0
802F  LD (#FD68),A               ;select SAVE.
8032  CALL #0AAE.
8035  RET                        ;return to BASIC
```

The above listing,should be saved to tape as a BASIC file,using SAVE "SAVECODE". Remember to Verify the program. When you use this program for tape to disc conversion,remember to edit the start and length of code in the above to that of the program to be converted. The same should be applied when reloading with listing B.

**Listing B:** Enter CP/M,then A>MTX,to return you to MTX mode. Now enter the following:

```
10 REM **LOADCODE utility**
20 CODE

8022  LD HL,32768                ;Insert start of code.
8025  LD DE,00000                ;Insert length of code.
8028  LD A,0
802A  LD (#FD67),A
802D  LD A,1
802F  LD (#FD68),A               ;select LOAD.
8032  CALL #0AAE.
8035  RET                        ;return to BASIC.
```

Now save this to tape: SAVE "LOADCODE" <RET>. You can use this for other conversions.

Introduction to Disc BASIC on CP/M systems.

The Memotech series can be broken down to three types of system:

1) MTX mode or tape mode.
2) SDX mode or single disc system.
3) SDX and FDX CP/M modes.

Type 1) is the mode which everyone should be familar with. If you have a MTX 500,free ram starts at 32768 or #8000,and if you have a MTX 512,free ram starts at 16384 or #4000. Most commercial games software,are written so that it will work on either ram system,but a few do require the full complement of 64k ram,like Agrovator.

The MTX Tape to Disc Booklet,was originally written for type 2) systems who have just upgraded from the standard type 1) system. However,as a number of CP/M users,were complaining about conversion difficulties and as I upgraded,my own system to full FDX CP/M,1 Mb drives,I decided to write this Appendix to help CP/M owners. The following description,is for CP/M owners who have disc BASICs,FDXB,SDXB3 or FDXB07.

FDXB07,is only for FDX CP/M,1mb disc drive systems and is equivalent to FDXB,which is found on FDX CP/M 500k systems and SDX CP/M 3.5",1mb systems. SDXB3,is found on 56 column CP/M disc operating systems. Unlike type 2)s disc BASIC which is loaded from ROM to high memory,at #D700,all other disc BASICs are loaded from DISC and are loaded at #4000. This means that,CP/M owners require a minimum of 64k. After the disc BASIC has loaded,you are left with only 32k to play with,from #8000. This means that on CP/M systems we are limited to MTX 500 game conversions. But as most games programs are less than 24k,(remember the MTXOS needs some memory as does the BASIC Interpreter,for storing variables;ie if BASIC is used in part,as used in Firehouse Freddie).

The Next problem,we encounter is after every RESET,we have to boot back into CP/M,then into the appropriate disc system,which is time consuming. It also means that any code from #8000 to #88FF,is automatically overwritten,by the system at start up. As is any game code that we might have stored there. However,I have devised a method around this it may be a bit long but it is worth it in the end.

The Procedure

Before preceding,please disconnect your disc system. This is because my FDX system interfered with the conversion. Also,the time consuming rebooting of CP/M after every RESET,and re-entering of MTX mode is removed. To disconnect your disc system,please ensure the power is off. Now remove the SDX interface from the MTX computer or remove the 60-way cable from the bottom of the FDX unit.

As already stated,only programs that run on the 32k MTX 500 computer will run on a CP/M disc system. This is because the disc BASIC is stored in RAM at 16384 (#4000) to 32676 (#7FFF).

We will use the famous TOADO as a worked example to demonstrate the techniques used.

```
POKE 64122,0:NEW                        <RET>
LOAD "TOADO"                            <RET>
```

Once loaded,press the RESET keys and:

```
PANEL                                   <RET>
D 8040                                  <RET>
00                                      <RET>
18                                      <RET>
.
B
Y     ;you should now be back in BASIC.
```

Rewind tape to start of SAVECODE and load this in,note that the MTXOS will load this utility at the start of free RAM which is reset to #4000. Now enter the assembler to make a few changes to the SAVECODE at #4022:

```
AS.10                                   <RET>
e                                       <RET>

4022 LD HL,32808                        <RET>
4025 LD DE,#2010                        <RET>
```

<CLS> <RET> <CLS> <RET> to return to BASIC. The start and length of TOADO has been inserted into the SAVECODE utility. Please note,people who have a version of TOADO with Oxford data on the title screen should change the length to #2039.Move the tape forward to a clear bit of tape. Press the record keys on the tape recorder and type:

```
RUN                                     <RET>
```

This will save the TOADO code as a byte file on tape. Once saved,we must perform two checks to see if the code is uncorrupted. These two checks will be used throughout:

i) To check the code in memory is uncorrupted type:

```
    RAND USR(32815)                     <RET>
```

    If the TOADO program reruns successfully then code in memory is okay.

ii) To check the code on tape is okay,type:

```
    LOAD "LOADCODE"                     <RET>
```

    Now edit the code at line 20,and insert the start and length of the program code. In the case of TOADO,start=32808 and length=#2010 or #2039. Now rewind tape to start of byte file press play and type: RUN <RET>

This will reload the TOADO code. To check it is okay,use check i). If this runs then okay,else repeat again.

Once loaded successfully,switch off the power and reconnect the disc system.Boot up CP/M,in the usual way then install your disc BASIC,ie:

```
A>FDXB07 40 or FDXB 40 or SDXB3 40.    <RET>
```

Once in 40 column mode disc system,retrieve:

```
LOAD "LOADCODE"                        <RET>
```

Enter the assembler at line 20 and edit lines #8022 & #8025,to LD HL,32808 & LD DE,#2010 or #2039. Now return to BASIC and enter PANEL and select M for MOVE and follow the prompts:

```
Move>8022
End >8035
To  >DF22
```

This will move the save LOADCODE to high RAM. Rewind the tape to the start of the TOADO byte file. Now type RAND USR(57122) <RET> and press play when the byte file has loaded in,save it to disc as:

```
DISC WRITE "TOADO.COD",32808,8208      <RET>
```

As a further check of the code saved to disc:Switch the computer off then on,reboot CP/M,select disc BASIC and type:

```
DISC READ "TOADO.COD",32808            <RET>
RAND USR(32815)                        <RET>
```

This should restart the TOADO program okay,if not try again.

Now that we have a working TOADO byte file on disc,all that is left is the BASIC header.

```
1 DISC READ "TOADO.COD",32808
2 RAND USR(32815)
3 DISC SAVE "TOADO.BAS"
4 RUN
```

Type:GOTO 3 <RET>. This will save the header to disc as TOADO.BAS. This should run the TOADO program successfully.

Thus to reload the TOADO game use:

```
DISC LOAD "TOADO.BAS"                  <RET>
```

when in disc BASIC mode.

The Worked Examples

Switch on your computer disc system and boot up CP/M. Take a blank disc and FORMAT it and copy the system tracks onto it. Refer to your disc operating manual for details on this. Now copy the following CP/M utilities onto it:

CONFIG.COM ,PIP.COM ,STARTUP.COM ,STAT.COM ,FDXB07.COM or FDXB.COM or SDXB3.COM

To do this refer to your disc operating manual. Once you have copied the above utilities,insert this disc into drive A (or default drive) and initialise by pressing <CTRL> C. Now follow this procedure carefully:

either for 1000k (1mb) disc systems:

```
A>STARTUP CONFIG B:07\FDXB07 40        <RET>
A>STARTUP                              <RET>
```

or for 500k disc systems:

```
A>STARTUP CONFIG B:03\FDXB 40 or SDXB3 40  <RET>
A>STARTUP                              <RET>
```

The above program is added to the CP/M startup utility. Now when you insert the games disk into drive A after a RESET,the system will auto-boot into the 40 column disc BASIC. Please note that,the games will only be displayed on a T.V. or via the computer monitor port next to the Hi-Fi socket.

Also,all games converted whether by the CP/M method or by the easier SDX disc BASIC method,are interchangeable. Therefore,if you upgrade from SDX to SDX/FDX CP/M,then all the <32k disc conversion will run automatically,even though they use different commands,ie USER or DISC

| DESCRIPTION | TOADO | KILOPEDE | QOGO | QUAZZIA |
|---|---|---|---|---|
| 01. Disconnect disc system when the power is off. | | | | |
| 02. Switch computer on and switch to TV mode. | | | | |
| 03. POKE 64122,0 | | | | |
| 04. LOAD"" | | | | |
| 05. press the RESET keys. | | | | |
| 06. PANEL | | | | |
| 07. D | 8040 | 8043 | 806F | 807D |
| 08. 00 | | | | |
| 09 | 18 | C3 | C3 | 18 |
| 10. . | | | | |
| 11. B then Y to goto BASIC | | | | |
| 12. LOAD "SAVECODE" | | | | |
| 13. AS.20 then E to edit | | | | |
| 14. 4022 LD HL, | 32808 | 32811 | 32864 | 32878 |
| 15. 4025 LD DE, ** | 8208 | 5392 | 5754 | 20370 |
| 16. <CLS> <RET> <CLS> <RET> now in BASIC again. | | | | |
| 17. Move the tape to a clear blank area of tape.Press the record keys & type | | | | |
| 18. RUN ,this will save the code in memory to tape as a byte file. | | | | |
| 19. Once saved,perform the following two checks | | | | |
| 20. RAND USR This should restart the game.Else goto step 3 | (32815) | (32818) | (32893) | (32885) |
| 21. Switch the computer off. then on again and | | | | |
| 22. LOAD "LOADCODE" | | | | |
| 23. AS.20 and e ,to edit code | | | | |
| 24. 4022 LD HL, | 32808 | 32811 | 32864 | 32878 |
| 25. 4025 LD DE, ** | 8208 | 5392 | 5754 | 20370 |
| 26. <CLS> <RET> <CLS> <RET> now in BASIC again. | | | | |
| 27. Rewind tape to start of the program byte file. | | | | |
| 28. RUN and press play. | | | | |
| 29. This loads the program into RAM. To check its OK | | | | |
| 30. RAND USR | (32815) | (32818) | (32893) | (32885) |
| 31. If the program restarts then okay.Else step 3. | | | | |
| 32. Switch system off and connect your disc system | | | | |

| DESCRIPTION | TOADO | KILOPEDE | QOGO | QUAZZIA |
|---|---|---|---|---|
| 33. Switch power on & insert the autoboot games disc You should now switch to TV mode as 40 column disc has been autobooted. | | | | |
| 34. LOAD "LOADCODE" | | | | |
| 35. AS.20 and e ,to edit code: | | | | |
| 36. 4022 LD HL, | 32808 | 32811 | 32864 | 32878 |
| 37. 4025 LD DE, ** | 8208 | 5392 | 5754 | 20370 |
| 38. <CLS> <RET> <CLS> <RET> now in BASIC again. | | | | |
| 39. Rewind tape to start of the program byte file. | | | | |
| 40. RUN and press play. | | | | |
| 41. This loads the program into RAM.Now save to disc: | | | | |
| 42. DISC WRITE "TOADO.COD",32808,8208 | or | | | |
| 43. DISC WRITE "KILOPEDE.COD",32811,9614 | or | | | |
| 44. DISC WRITE "QOGO.COD",32864,5754 | or | | | |
| 45. DISC WRITE "QUAZZIA.COD",32878,20370 | . | | | |
| 46. Once saved to disc check the code on disc is okay | | | | |
| 47. Repeat step 33. | | | | |
| 48. To reload the disc files above,use steps 42-45, except change WRITE to READ.Once the program has reload,test it thus | | | | |
| 49. RAND USR This should restart the game.Else goto step 33 | (32815) | (32818) | (32893) | (32885) |
| 50. The program codes have been saved to disc successfully. Now we will use a samll BASIC header to call up the code and autorun the game for us. | | | | |

**,Note that some of the Continental games have been redistributed with Oxford Data on the tiltle screen. In these instances it is worth checking the length of the source code. I have found with TOADO that the length of the code has increased from #2010 (8208) to #2039. So beware.

The BASIC Headers

Now that we have working byte files of the above programs on disc,all that is left is the BASIC header. Once you have typed in the HEADER,to save it to disc ,type: GOTO 3 <RET>.To reload the autorun programs use:

DISC LOAD "PROGRAM.BAS" <RET>

This will load the code into RAM and autorun it for you.

TOADO.BAS

```
1 DISC READ "TOADO.COD",32808
2 RAND USR(32815)
3 DISC SAVE "TOADO.BAS"
4 RUN
```

KILOPEDE.BAS

```
1 DISC READ "KILOPEDE.COD",32811
2 RAND USR(32818)
3 DISC SAVE "KILOPEDE.BAS"
4 RUN
```

QOGO.BAS

```
1 DISC READ "QOGO.COD",32864
2 RAND USR(32893)
3 DISC SAVE "QOGO.BAS"
4 RUN
```

QUAZZIA.BAS

```
1 DISC READ "QUAZZIA.COD",32878
2 RAND USR(32885)
3 DISC SAVE "QUAZZIA.BAS"
4 RUN
```

Please note that because DRAUGHTS starts at 32788,there isn't enough room for this type of BASIC header. However,you can still save and load the byte file using DISC WRITE "DRAUGHTS.COD",32788,9614 to save and use DISC READ "DRAUGHTS.COD",32788 to load and run it from RAND USR(32795).

## Appendix 4.0: References and User Groups

User Groups

**MEMOPAD** published by ORION s/ware.

The Northbridge Centre,Elm Street,Burnley.

12 issues at #18 (UK price).

**Memotech Owners Club**

c/o Phil Eyres,13 Copse Road,TownHill Park,Southampton.

10 issues at #7 (UK price),send SAE for more details.

Note that Popular Computing Weekly also support Memotech material. They pay #25 per page published. Send all those useful short utilities to:

Duncan Evans,Popular Computing Weekly,Focus Magazines,Greencoat House,Francis street,London SW1P 1DG,telephone 01-834-1717.

Hardware & Software Suppliers

**Memotech Computers Ltd**

Unit 24,Station Lane Indusrtrial Estate,Witney,Oxon,OX8 6BX.

**UK Home Computers**

82 Churchward Ave,Swindon,Wilts,SN2 1NH. Tel 0793 695034.

**Memopad**

address as user group.

**M.O.C.**

address as user group.

References

(01) MTX Manual by B.Pritchard,pages 133 & 161.

(02) Using the MTX Front PANEL,Memopad,vol 1,issue 4,pages 7-10.

(03) Popular Computing Weekly,vol 6,issues 25 & 26.

(04) New Memotech Operators Manual by S.Bateson,pages 204-213.

(05) Z80 Assembly Language Programming by M.C.Moore,chapter 1.

(06) Acorn/BBC BASIC ROM user guide by M.Plumbley.

(07) Utilities,Memopad,vol 1,issues 7 & 8.

(08) Popular Computing Weekly,vol 6,issue 21,pages 22 & 23.

(09) M.O.C.,vol 3,issue 7.

(10) Writing Interactive Compilers & Interpreters by P.J.Brown.

(11) Faster BASIC,Popular Computing Weekly,vol 6,issue 20,page 23.
[also,Memopad,vol 3,issue 9,pages 15-18]

(12) Memotech Memory,Your Computer,March 1984,page 99.

(13) Memory Map,Memopad,vol 1,issue 1,page 20.

(14) Memopad,vol 1,issue 2,page 40.

(15) Memotech Manual by B.Pritchard,pages 245-247.

(16) New Memotech Operators manual by S.Bateson,page 244.

Note that an excellent BOOK guide is given in MEMOPAD,vol 3,issue 7/8,pages 47-50.