IBMDISC - Memotech to MSDOS Disc Transfer

IBMDISC Version 1.0
A Program to Read and Write MSDOS 3.5" 720K discs
on a Memotech MTX512 CP/M System

1. Introduction

This program was written mainly to obtain access to Public Domain CP/M software. It provides the Memotech with a standard format for interchange of software, specifically the MSDOS 3.5" 720K disc format.

The program is written for a Memotech MTX512, with 64K of memory and a 3.5" CP/M system consisting of one 3.5" drive (drive B:) with SDX controller, and one 512K RAM disc (drive F:). It should be possible to adapt the program for other hardware combinations, both Memotech based and otherwise. See section 6 for further comments.

The system uses the 3.5" drive to read and write the MSDOS format discs, and transfers files to and from the RAM disc to provide the interchange with CP/M. There are options to list the directory and contents of the MSDOS disc, copy files to and from the MSDOS disc, and to erase files on the disc.

The current version of the program does not support sub-directories on the MSDOS disc, but it will not damage any existing sub-directory structure. There is no option for formatting the disc, it must be formatted on an MSDOS machine.

2. Using the Program

If it is intended to use the program to transfer files, then the RAM disc must be formatted before invoking the program. Also any files to be transfered to the MSDOS disc must have been copied onto the RAM disc. I normally have CP/M cold-booted from the RAM disc, but this is not essential.

Invoke the program from the CP/M prompt, without any parameters, thus:

```
     A>IBMDISC
```

You will be requested to insert the MSDOS disc into the drive, and press any key. Once you have done so, the BIOS Parameter Table, File Allocation Table (FAT), and Root Directory will all be read from the disc (mapping the disc). This information will subsequently retained in memory, and be used to update the disc. Therefore do not change the disc unless you use the CHANGE or EXIT commands. A summary of the BIOS Parameter Table will be printed on the screen. If an error occurs while reading the disc map, the program will exit.

Once the disc has been mapped, the "> " prompt is displayed. Any of the following commands may then be issued:

CHANGE

This command is used if you wish to change MSDOS discs. Change the discs when prompted to do so, then press any key. The new disc will then be mapped. The program will exit if an error occurs while reading the disc map.

DATE dd-mmm-yy

This command is used to set the date that will be used for date stamping files written to the MSDOS disc. The default date used by the program is 1-Jan-80. This version of the program does not have any option for setting the time stamp, which will default to midnight.

DEL filename.ext or ERA filename.ext

This command deletes the specified file from the MSDOS disc. The directory entry of the file is displayed as the file is deleted. Wildcards (* or ?) may be used in the filename and/or extension, in which case every file matching the specification will be deleted.

DIR or DIR filename.ext

Lists the MSDOS directory entry for all the files, or the specified file(s). The information listed is the file name and extension, the file size in bytes, and the date and time the file was last modified.

The directory listing is paged in blocks of 20 lines. Pressing <CTRL+C> at the end of a page returns you immediately to the command prompt. Any other key will display the next page of the listing.

ERROR ON or ERROR OFF

This is used to control the processing of read and write errors on the MSDOS disc. When error on is set (the default) then every sector read or write that produces an error will be reported. If error off is set then only the total number of errors will be displayed. Read and write errors on the disc map area will always be reported.

EXIT

This command is used to return to CP/M. Remove the MSDOS disc when prompted to do so. If the CP/M default drive is the 3.5" drive, then replace the MSDOS disc with the CP/M system disc. Then press any key.

GETA filename.ext

This command is used to transfer ASCII files from the MSDOS disc onto the RAM disc. Wildcards may be used, in which case all

matching files will be transfered. Do not specify a drive letter, the drives for both the MSDOS disc and the CP/M disc are fixed by the program. The CP/M file(s) will be given the same name as the corresponding MSDOS file(s). If a file of the same name already exists on the RAM disc, then it will be deleted, and replaced by the file from the MSDOS disc.

If the MSDOS file already contains an end of file mark (<CTRL+Z>) then the CP/M file will be truncated at this point. If not, then one will be added at the end of the file.

    GETB filename.ext

This command is used to transfer binary files from the MSDOS disc onto the RAM disc. Wildcards may be used, in which case all matching files will be transfered. The CP/M file(s) will be given the same name as the corresponding MSDOS file(s). If a file of the same name already exists on the RAM disc, then it will be deleted, and replaced by the file from the MSDOS disc.

Unlike GETA, there is no <CTRL+Z> processing. The size of the MSDOS file is rounded up to the next CP/M record (128 bytes).

    LIST filename.ext

This command lists to the screen the contents of the specified MSDOS file(s). The listing is paged every 20 lines. Pressing <CTRL+C> at the end of the page returns you to the command prompt. Any other key will display the next page.

Wildcards may be included in the filename, in which case all the matching files will be listed in turn.

    PUTA filename.ext

This is used to transfer ASCII files from the RAM disc onto the MSDOS disc. Wildcards may be used, in which case all matching files will be transfered. Do not specify a drive letter, the drives for both the MSDOS disc and the CP/M disc are fixed by the program. The MSDOS file(s) will be given the same name as the corresponding CP/M file(s). If a file of the same name already exists on the MSDOS disc, then it will be deleted, and replaced by the file from the RAM disc.

If the CP/M file contains an end of file mark (<CTRL+Z>), then the MSDOS file will be truncated to the byte preceeding the end of file mark.

    PUTB filename.ext

This is used to transfer binary files from the RAM disc onto the MSDOS disc. Wildcards may be used, in which case all matching files will be transfered. The MSDOS file(s) will be given the same name as the corresponding CP/M file(s). If a file of the same name already exists on the MSDOS disc, then it will be

deleted, and replaced by the file from the RAM disc.

Unlike PUTA, there is no <CTRL+Z> processing. The size of the MSDOS file is determined by the number of CP/M records.

```
     SAVEMAP filename.ext
```

This command was included mainly for debugging purposes. It saves the current map of the MSDOS disc (prameter table, FAT, and root directory) from memory as a CP/M file.

```
     SECTOR nnn
```

This command reads the specified logical sector on the MSDOS disc and displays its contents in both hex and ASCII. The sector number may be specified either in decimal, or if proceeded by a hash (#), in hex.

This version of the program does not recognise the MSDOS read-only, hidden, or system attributes. These files are listed, copied, or deleted exactly as any other files. Sub-directory files are not listed and cannot be accessed in any way (except by the SECTOR command). Although the program cannot access any files in sub-directories, it will not harm them.

3. Reversibility of Transfers

Transfers of normal ASCII files (such as program source code) using GETA and PUTA should be completely reversible. However if the file contains an end of file mark part way through its allocated space, then the file will be truncated.

Transfers of CP/M binary files using PUTB followed by GETB should also be completely transparent, allowing program files, overlays, word processor files etc to be transferd.

Transfers of MSDOS binary files, using GETB followed by PUTB will result in the size of the file being rounded up to the next multiple of 128 bytes. This may cause problems with MSDOS .EXE files.

4. Error Messages

The following error messages are produced by the code that performs physical reads and writes on the MSDOS disc:

"Track zero error" - An error occured when performing a seek to track zero.

"Seek error" - An error occured when performing a seek to the specified track.

"Read error" - An error reading a sector from the disc.

"Write error" - An error writing a sector to the disc.

These error messages are followed by a list of all the bits set in the FDC1793 status bytes. The possible error bits are:

- Not Ready.
- Write Protect.
- Head Loaded.
- Seek Error.
- CRC Error.
- Track Zero.
- Deleted Record.
- Record not Found.
- Lost Data.

These are followed by the track, side, and sector number at which the error occured, and then by the message "Continue, Retry, or Abort ...". Selecting Continue (pressing C) causes the error to be ignored, and the operation continued. Selecting Retry causes the operation to be retried. If the retry is successful then the operation will be continued, otherwise a new error message will be produced. Selecting Abort terminates the current operation and returns to the command prompt.

The following messages are produced when mapping the disc structure. They indicate either that the disc is too large, or that it is not a valid MSDOS disc:

"Error reading disc map." - A non-recoverable read error occured while trying to read the disc map.

"Disc not valid MSDOS format" - The boot sector of the disc does not start with an 8086 near or short jump. It is therefore not a valid MSDOS format disc. A CP/M disc by mistake?

"File Allocation Table too large." - The disc contains too many clusters. Usually implys the disc is too large.

"Root Directory too large." - The root directory contains too many directory entries.

Any of these errors result in prompting to remove the disc and then terminate the program. This also occurs if there is a non-recoverable read error while reading the disc map.

The following errors may occur when traversing the MSDOS file structure:

"Track register overflow." - An attempt was made to access a track beyond 255. Either the disc is too large or the BIOS parameter table is corrupt.

"Unexpected end of FAT chain." - The end of the chain of allocated clusters was encountered before the end of the file indicated by the size in the directory entry. The structure of the MSDOS disc has become corrupted.

"Broken cluster chain." - An unallocated cluster was encountered in the middle of a cluster chain while deleting a file. The MSDOS disc structure is corrupt.

The following messages are produced by the command interpreter:

"Invalid command." - The command has not been recognised. Check spelling and syntax.

"Invalid parameter." - The parameters of the command are not valid.

"Error in date format." - The format of the date specified in the set date command is incorrect. It should be in the form "dd-mmm-yy".

The following errors occur during file transfer:

"Zero length file." - An attempt was made to transfer a file of zero length. The file is not transfered.

"Directory full." - The MSDOS root directory is full. Space can be created using the DEL command.

"Disc full." - All the clusters on the MSDOS disc have been allocated. Space can be created using the DEL command.

"nnn Read error(s)" - A number of read errors occured on the MSDOS disc while transfering a file.

"nnn Write error(s)" - A number of write errors occured on the MSDOS disc while transfering a file.

"Error opening CP/M file" - Usually caused by the directory on the CP/M disc (RAM disc) being full.

"Error closing CP/M file." - Error accessing the CP/M directory.

"Write error on CP/M file." - Usually caused by the CP/M disc (RAM disc) becomming full.

5. Code Description

The code is written for the Z80 processor. It makes use of the index registers, alternate register set, and Z80 specific opcodes.

The code assumes that the BDOS and BIOS both preserve the index registers and alternate register set. This can be expected of BDOS, which was written for the 8080, which does not have any of these registers. The BIOS on other Z80 machines may, however, make use of these registers.

All access to the keyboard, screen, and RAM disc are made using

standard CP/M 2.2 BDOS calls. Screen control is of the glass teletype variaty, no special emulations are required.

The only machine specific code is that required to read and write 512 byte sectors from the MSDOS disc. This code is localised in the file IBMBIOS.Z80.

The source code is divided up into files as follows:

BDOS.Z80

This file just contains a range of standard equates for CP/M, and a few ASCII control codes.

SCREEN.Z80

These are my standard screen and keyboard handling routines. In general, all these routines preserve the main register set. There are alternate entry points that do, or do not, preserve the alternate register set.

DECODE.Z80

These are my input decoding routines. They include utilities for extracting decimal and hex numbers, for identifying key words, and parsing file names.

IBMBIOS.Z80

This file contains all the machine specific code. It is the code for performing physical reads and writes to the MSDOS disc. The Memotech 3.5" SDX disc controller uses an FDC1793 controller chip with programmed I/O.

The following variables are public:

IBMDMA - (16 bits) This variable points to the start in memory of the sector to be written to disc, or the buffer in memory to receive the sector read from disc.

IBMTRK - (8 bits) Contains the track number to read from or write to.

IBMSID - (8 bits) This variable contains 0 or 1 depending on which side of the disc is to be read or written to.

IBMSEC - (8 bits) Contains the physical sector number to read or write.

FDERC - (16 bits) This variable contains a running total of the number of hard errors that occur on the MSDOS disc. The most significant bit (msb) is used to control error handling. If the msb is set, then hard errors are ignored (but counted). If the bit is clear, then a hard error produces a Continue, Retry, or Abort message. If this code is being repaced for another machine

then this variable need not be implemented, but it must be defined.

There are three entry points to this code that are used externally:

IBMRD - This routine reads the sector specified by the IBMTRK, IBMSID, and IBMSEC variables into memory at the location specified by IBMDMA.

IBMWR - This routine writes the data from memory at the location specified by IBMDMA to the disc at the sector specified by the variables IBMTRK, IBMSID, and IBMSEC.

FDOFF - This routine is used to turn the floppy drive motor off after a series of sectors have been read or written to.

None of these routines preserve the main register set. They should however preserve the alternate set and the index registers.

IBMDOS.Z80

This file contains the routines to traverse the File Allocation Table, and to convert clusters to logical sectors, and logical sectors into physical track, side, and sector numbers.

None of this code is specific to the 720K disc. All the dimensions of the disc are obtained from the BIOS parameter table in the first sector. The code does however make various assumptions about the "smallness" of the disc. The main limitations are:

a) The FAT table is assumed to be 12 bits per cluster. Thus the maximum cluster number is 4080.

b) The maximum logical sector number is 64K-1.

c) The maximum number of sectors per track is 127.

d) The maximum number of disc heads is two.

e) The maximum number of tracks is 255.

f) The maximum number of entries in the root directory is 128. This may be increased up to 255 by changing an equate statement in the program.

g) The maximum size of the FAT table is 3 sectors. This may be increased by changing an equate statement in the program.

IBMDIR.Z80

This file contains all the code for manipulating the MSDOS root directory. This version of the program does not support sub-

directories.

The file size printed by the directory listing routine is restricted to 999,999 bytes. This limitation is purely in the output formatting. The actual file size is maintained to a full 32 bits, it is only reduced modulo 1 Mbyte when displayed.

IBMDISC.Z80

This is the main program code. It contains the startup code, the command parser and jump table, and various utility routines. All the other source files are referenced as (nested) include files from this file.

IBMGET.Z80

This is the code for extracting files from the MSDOS disc, either to he screen or to the RAM disc.

IBMPUT.Z80

This file contains the code for writing files to the MSDOS disc.

To those who attempt to modify and re-assemble the program, my appologies. I wrote my own assembler and so the source code may be slightly exotic. The following notes should help:

a) All the Z80 opcodes should be as standard.

b) All numeric constants are decimal unless preceeded by a hash sign (#), in which case they are in hex.

c) All constant expressions are evaluated using 16 bit signed integer arithmetic. A single character within quotes evaluates to its ASCII value. The arithmetic operators (+,-,*,/) obay the ususal precedence rules. The & operator is a bitwise and.

d) The DEFB and DEFW pseudo ops initialise byte and word variables.

e) The DEFS and DEFC pseudo ops define ASCII strings. The DEFC operator sets the most significant bit of the last character in each string. The backslash character, followed by a pair of hex digits, is used as an escape sequence to enable non printing characters, such as control codes, to be included within the strings.

f) The BYTE and WORD pseudo ops reserve space for the specified number of byte or word values, without initialising them.

g) The EQU pseudo op assigns a numeric value to a label.

h) The INCLUDE pseudo op includes the entire text of the nominated file at its location. INCLUDE statements may be nested as required.

6. Adapting Code to other Computers

Although the code as written is specific to the Memotech with 3.5" disc drive, it should be fairly easy to adapt it to other machines.

The fact that the program copies files to and from the Memotech RAM drive is not of any great significance. It is the only other drive available on my Memotech, and is being accessed via standard CP/M calls. Its one main advantage is that it does not use the floppy disc controller chip.

Memotech users with 5.25" disc drives and an SDX controller have two problems. Firstly, the second drive is another 5.25" driven by the same controller chip. Reading and writing to the two drives may therefore interfere with each other. Secondly, the IBM 360K disc is only 40 track (1.2M discs are probably too high a density to read on the Memotech). Therefore it will be necessary to double step the head stepper motor. The controller is, however, essentially the same as that for the 3.5" drive. Disassembling the disc driver starting at #FFF3 and #FFF6, should provide the necessary information.

The Memotech FDX disc controller, I believe, differs substantially from the SDX controller for both 3.5" and 5.25" drives. This controller uses DMA access rather than programmed I/O. Again I would suggest starting with dissassembly of the Memotech disc controller software.

For owners of other Z80 based machines, the only machine specific code is that in IBMBIOS.Z80, which is responsible for physically reading and writing to the disc. This will have to be replaced by alternate code which performs the same function. If you are fortunate enough to have a disc controller that uses one of the FDC1793 family of chips then the existing code may provide a good starting point. Once again, examination of the code for your machines disc BIOS should prove illuminating.

If you don't have a Z80, then I wouldn't start from here if I were you.

Good luck.

7. Copyright Statement

This program, source code, documentation, etc are all copyright. They are made freely available for use, modification, and/or distribution AT YOUR OWN RISK subject to two conditions:

a) The copyright statements are retained in all source code, documentation, etc.

b) No profit is made from the use, modification, or distribution of this software.