SAB Diskette Utility Index

Stewart A. Berman

Mr Berman is an excutive with 20 years of experience in all aspects of MIS development and management who has yet to decided what he wants to be when he grows up. He has worked as an electrical engineer (digital corelation equipment and televison systems), programmer/analyst (electron tube simulation), systems programmer (MFT, MVT, MVS, VM, IMS, CICS, IDMS), and a manager and partner in a "Big Eight" (now "Big-Six") accounting firm. Along the way he has picked up a bachelors and masters in electrical engineering a masters in professional accounting, and passed the CPA and CDP examinations.

Mr Berman is currently enjoying life -- if not making a living -- as an independent consultant in New York. His specialty is cleaning up the mess others leave behind. He will even do windows.

Mr Berman can be reached via CompuServe mail addressed to 76366,1664. He can also be reached via snail mail addressed to Stewart A. Berman, 34 Adler Place -- Suite B, Bronx, NY 10475. Mr Berman is also a BBSer and is known to hang out on InterLink in the C, Assembler, Technical, and Windows conferences and on P/NET in the C and Assembler conferences.

Copyright

Disclaimer

THIS PRODUCT IS OFFERED ON AN "AS-IS" BASIS. THE AUTHOR MAKES ABSOLUTELY NO CLAIM THAT IT WILL WORK AS DESCRIBED OR EVEN WORK AT ALL. THE AUTHOR DISCLAIMS ALL RESPONSIBILITY FOR DAMAGES CAUSED BY THE FAILURE OF THIS PRODUCT. YOU HAVE A TEN DAY EVALUATION PERIOD TO DETERMINE WHETHER OR NOT THIS PRODUCT MEETS YOUR NEEDS. REGISTRATION OF THIS PRODUCT GIVES YOU A LICENSE TO CONTINUE USING IT. IT DOES NOT PROVIDE ANY GUARANTEE THAT IT WILL CONTINUE TO WORK OR EVEN CONTINUE TO RUN.

YOUR USAGE OF THIS PRODUCTS INDICATES YOUR WILLINGNESS TO ACCEPT COMPLETE RESPONSIBILITY FOR ITS USE.

Introduction

SAB Diskette Utility is a Windows 3 application that provides the user with a set of services that makes the copying, comparing, and formatting of diskettes a pleasure instead of a pain. It makes use of the Input/Output (IOCTL) interface to read/format/write a track at a time.. It will copy, in a single pass, the entire contents of a diskette, at the sector level, into an image either in memory or in a tempory hard disk file. The image can then be used to produce multiple copies of the original diskette. The system makes use of the Windows 3 Messaging and Timers to work cooperatively with other Windows 3 applications. It can be run entirely in Iconic *mode.*

License

HIS PRODUCT IS NOT FREE. IT IS OFFERED ON A "SHAREWARE" BASIS. YOU HAVE A TEMPORARY LICENSE TO USE THIS PRODUCT FOR TEN DAYS TO EVALUATE ITS USEFULNESS TO YOU. IF YOU WISH TO KEEP USING IT YOU MUST REGISTER IT.

The difference types of licenses available are:

Home Use: This license gives the user the right to install a copy of SAB Diskette Utility on one non-commercial machine used at home. The cost of this license is $10 (US).

Business - Single User: This license gives the user the right to install a copy of SAB Diskette Utility on one CPU used in the performance of the user's business. The cost of this license is $15 (US).

Business - Site: This license gives the user the right to install a copy of SAB Diskette Utility on all machines used in the performance of the user's business at one location. The cost of this license is $100 (US).

Business--Unlimited: This license gives the user the right to install a copy of SAB Diskette Utility on all machines used in the performance of the user's business anywhere in the world. The cost of this licens is $250 (US).

The business licenses also give the users of those machines the right to install SAB Diskette Utility on a non-commercial machine used at home. This right to use SAB Diskette Utility at home ceases if the user leaves the company holding the business license.

To register it select the registration option on the main menu, fill in the information, print the completed form, sign it, and send it with a check for the appropriate amount to the address shown on the form.

Read

The read command will use the Input/Output Control read (IOCTL READ) subfunction to read all of the sectors on a diskette into an image in memory or an image on hard disk in a single pass. The decision to use a memory image or a hard disk image can be made automatically by the system based on available resources or it can be forced by the user through the disk spooling option that can be set using the configure command The read command uses Windows 3 messaging and Timers to work cooperatively with other WIndows 3 applications in sharing the systems resources.

To invoke the read command "click" on the read menu item or press the Alt key and then the R key. A window will open with instructions to insert the diskette into the appropriate diskette drive and press the button corresponding to the drive. A cancel button is also available to terminate the read command at this point.

The system will read the contents of the diskette a track at a time. It reads all of the tracks on a cylinder before using Windows 3 messaging and Timers to give up control to other Windows applications.

The read command will display a window with a completion notice when it finishes reading all of the sectors on the diskette. It will also enable the menu items that require a completed diskette image to work (compare and write).

Compare

The compare command compares the contents of an image in memory or an image on hard disk with the contents of a diskette. It uses the Input/Output Control read (IOCTL READ) subfunction to read all of the sectors on a track at a time into memory. It then compares the contents of the track with the stored image. If there are no differences it will proceed to the next track. If there are any differences it will ask the users whether to stop the compare function or to continue anyway with the next track. It compares all of the tracks on a cylinder before using Windows 3 messaging and Timers to give up control to other Windows applications.

To invoke the compare command "click" on the compare menu item or press the Alt key and then the C key. A window will open with instructions to insert the diskette into the appropriate diskette drive and press the button corresponding to the drive. A cancel button is also available to terminate the compare command at this point.

The compare command will display a window with a completion notice when it finishes comparing all of the sectors on the diskette.

Format

The format command formats an entire diskette. The format command uses Windows 3 messaging and Timers to work cooperatively with other WIndows 3 applications in sharing the systems resources.

To invoke the format command "click" on the format menu item or press the Alt key and then the F key. A window will open with instructions to insert the diskette into the appropriate diskette drive and press the button corresponding to the drive. A cancel button is also available to terminate the format command at this point. If the selected diskette drive supports more than one format mode a pop-up menu will be displayed listing the available formatting modes.

The format command will attempt to read the first sector and analyze the Diskette Parameter Table (DPT) before formatting. If it can read the first sector and the format mode of the diskette does not match the format mode requested it will terminate the format operation.

Formatting is done using the Input/Output Control format (IOCTL FORMAT) subfunction to format a track at a time. It formats all of the tracks on a cylinder before using Windows 3 messaging and Timers to give up control to other Windows applications.

The format command will display a window with a completion notice when it finishes formatting the entire diskette.

NOTE: The format command will stop if there are bad sectors on the diskette. It will prompt for a retry/cancel response from the user.

Write

The write command will use the Input/Output Control write (IOCTL WRITE) subfunction to write all of the sectors on a diskette from an image in memory or an image on hard disk in a single pass. The write command uses Windows 3 messaging and Timers to work cooperatively with other WIndows 3 applications in sharing the systems resources.

To invoke the write command "click" on the read menu item or press the Alt key and then the W key. A window will open with instructions to insert the diskette into the appropriate diskette drive and press the button corresponding to the drive. A cancel button is also available to terminate the write command at this point.

The system will write the contents of the diskette a track at a time. It writes all of the tracks on a cylinder before using Windows 3 messaging and Timers to give up control to other Windows applications.

The write command will display a window with a completion notice when it finishes writing all of the sectors on the diskette.

Exit

The exit command terminates SAB Diskette Utility. It will also save the users preferences in the SAB.INI file.

To invoke the exit command "click" on the exit menu item or press the Alt key and then the X key.

Cancel

The cancel command can be used to stop any of the long running SAB Diskette Utility functions. It is only enabled during read, compare, format, and write command operations.

To invoke the cancel command "click" on the cancel menu item or press the Alt key and then the N key.

Help

The help command provides access to the on-line help for SAB Diskette Utility. It also provides access to the "About Box" that identifies the copyright owner.

To invoke the help command "click" on the help menu item or press the Alt key and then the H key.

Configure

The configure command provides the user with the ability to select the appropriate settings for the maximum and minimum Timers, the desired image spooling option, the desired format while writing option, and to change diskette drive definitions.

The maximum timer defines the maximum number of milliseconds (.001 seconds) that the system will allow other applications to run before resuming the current operation. The default setting of 9999 disables the use of this timer. The way the system is designed it should not be necessary to use this timer unless there are a number of not well behaved Windows applications running and Windows is not providing enough time for the system to complete its operations in a reasonable time. It can also be used to convert the system into a not well behaved Windows application by setting it to zero. Please note that setting the maximum timer to zero will also disable the ability to cancel an operation in process.

The minimum timer defines the minimum number of milliseconds that the system will wait before even attempting to get scheduled. The default setting of zero disables the use of this timer. The way the system is designed it should not be necessary to use this timer unless there the interruption caused by repeated diskette operations is disruptive to time critical applications running at the same time. The use of this timer will increase the time between diskette accesses. Unfortunately Windows 3 does not provide the ability to have other tasks use the CPU while one task is waiting for a diskette operation to complete. Since diskette operations take a relatively long time a series of closely executed ones might prove disruptive to other applications running at the same time.

The A: and B; drive types define the way the system thinks the drives can be used. It should not be necessary to change this unless a drive is replaced, added, or removed. However, it is possible that the actions of other applications may have changed the drive definitions just before the first use of the system and it therefore failed to define them properly. If it is necessary to change a definition simply "click" on the new one.

The hard disk spooling option defines whether the system will always use the hard drive for the diskette image, never use the hard drive for the diskette image, or only use the hard drive for the diskette image if insufficient memory is available.

The format option defines whether the system will always format the output diskette when writing to it, never format the output diskette, or format the output diskette only when it cannot read it.

Register

The register command is used to enter the user's name, company name (if not a personal use license), address, phone number, and the registration key and to print a registration form. The first time the system is used the user's name and address should be entered and saved.

The user key is generated by the system at the time the registration information is saved or printed. It will be used as a check that the user information has been correctly processed during the generation of the registration key.

The registration key will be sent to the user after the registration agreement and fee have been received by the author.

The registration type should be selected by "clicking" on the option desired. Please note that the choice of a home use registration will force the company name to "Personal Copy".

After all of the information, except the registration key, has been entered a registration form can be printed by pressing the Print button. That will print a registration form and save the user's information. Pressing the Save button will save the information without printing a registration form. Pressing the Cancel button will terminate the registration command without saving the information.

The information is saved in the SAB.INI file.

First Use

The first time SAB Diskette Utility is used it will display a screen that describes the evaluation terms and one that explains the disclaimer. The user has the option to stop the installation process at that time if they do not want to abide by the terms shown by pressing the Cancel button. Pressing the OK button accepts the terms and continues the installation.

The menu bar will have all of the menu items disabled except the Exit and Registration commands. The Registration command should be used to enter and save the user's name and address. Once that is done the other commands will be enabled.

Standard Initialization

The SAB Diskette Utility uses a private INI file to store parameters in. The name of the file is SAB.INI. It will be created in the Windows directory the first time the system is used.

During initialization the system will obtained the size and position of the main application window the last time the system was used. It will also obtain information stored during the use of the configuration and registration commands. These items will be used to initialize the window and diskette drive control structures.

Windows Messages

Windows' applications process and send "messages". A "message" contains information about an event that has occurred. For example, "clicking" on a menu item generates a "message".

An application can also generate and process user defined "messages". Each portion of the application can be viewed as a closed subsystem that receives a "message", performs a task, and returns control. Part of the task performed might be the generation of another "message".

SAB Diskette Utility makes use of the Windows messaging system to control the flow of control through the application. Consider the read command -- it is made up of three sections. The initialization section receives the "message" generated when the read menu item is "clicked". It prompts for the diskette and invokes a service routine to analyze the Diskette Parameter Table (DPT) and determine the number of cylinders, heads, and sectors/track. It then sets the current cylinder, head, and track variables to zero and sends a "message" to the read cylinder section.

The read cylinder section reads all of the sectors on a cylinder and stores them. It then increments the cylinder variable and checks to see if there are more cylinders to read. If there are more cylinders to read it sends a "message" to itself to schedule the next cylinder read. If there aren't any more cylinders to read it sends a "message" to the read termination routine which displays the read completed message and then ends without sending any "messages".

The "messages" aren't sent immediately. They are placed into a holding variable and only sent if Windows indicates that there isn't any other work available for it to schedule. At that point the system places the message into the applications queue and it is processed and the function scheduled.

Timers

SAB Diskette Utility can use Windows' timers to control the scheduling of its internal operations. The way the system uses the timers is to start a timer and request that Windows send a special "message" when the time interval ends. That "message" causes a function to execute. Consider the maximum timer available through the configure. Let us assume that a read cylinder operation has just completed and that there are more cylinders to read. The system puts a read cylinder "message" into a holding variable. The "message" will normally be sent the next time Windows has nothing to do. However, if the maximum timer value has be set the system will also start a timer. If Windows has nothing to do before the timer expires the "message" will be sent and, as part of the application code that does that, the timer will be stopped. If Windows does not run out of other things to do before the timer expires a timer "message" will be sent to the application. When the application receives the "message" it will check and see that it has a "message" to send and send it at that time scheduling the next read cylinder cycle.

Iconic Operation

SAB Diskette Utility can operate completely in the Iconic mode. The Iconic mode is when an application's window has been minimized. It then normally displays an Icon in the lower portion of the display.

The system monitors changes to and from the Iconic mode. When the user puts the system into the Iconic mode it modifies the system menu by adding all of the menu items that would normally appear on the menu bar. It removes the menu items when the user takes the system out of Iconic mode.

The system will also use the space normally occupied by an Icon to display the current cylinder for read, compare, format, and write operations. Otherwise it will display its own Icon.

IOCTL

Input/Output Control (IOCTL) is a method of communicating directly with a device driver. SAB Diskette Utility uses the set of subfunctions associated with generic I/O control for block devices. The IOCTL interface is accessed through an interrupt call (INT 21H -- the general DOS interrupt -- with AH(function) = 44H, AL(subfunction) = 0DH, BL = drive number, and CH = 08H) using a Parameter Block pointed to by DS:DX. The minor subfunctions used are:

CL = 40H	Set Device Parameters
CL = 41H	Write track on logical drive
CL = 42H	Format and verify track on logical drive
CL = 60H	Get Device Parameters
CL = 61H	Read track on logical drive

Diskette Parameter Table

The Diskette Parameter Table (DPT) is located at the begining of the first physical sector on a diskette. It can be mapped in C using the following structure:

```
#pragma pack(1)
typedef struct
        {
        unsigned char   DSKJMP[3];
        unsigned char   DSKID[8];
        unsigned short DSKSECBY;
        unsigned char   DSKCLUSC;
        unsigned short DSKRESSC;
        unsigned char   DSKFATS;
        unsigned short DSKROOTD;
        unsigned short DSKSECTS;        Total sectors
        unsigned char   DSKFMTID;
        unsigned short DSKFATSC;
        unsigned short DSKTRKSC;        Sectors per track
        unsigned short DSKHEADS;        Number of heads
        unsigned long   DSKSPEC;
        unsigned long   DSKBIGTL;
        unsigned char   DSKPHYDR;
        unsigned char   DSKRESER;
        unsigned char   DSKEXNTD;
        unsigned long   DSKSRLNO;
        unsigned char   DSKVOLLB[11];
        unsigned char   DSKFATTP[8];
        } DSKPARAMS ;
#pragma pack()
```

Note the pack(1) pragma. Otherwise the C compiler will align the long variables on an even boundary and the mapping will fail.

IOCTL Parameter Blocks

The IOCTL Parameter Blocks can be mapped in C with the following structures:

```
#define IOCTLSETPARAMETERS     0x40
#define IOCTLWRITETRACK        0x41
#define IOCTLFORMATTRACK       0x42
#define IOCTLGETPARAMETERS     0x60
#define IOCTLREADTRACK         0x61
#define IOCTLVERIFYTRACK       0x62

#pragma pack(1)
#ifndef PARAMETER_BLOCK_SWITCH
#define PARAMETER_BLOCK_SWITCH
typedef struct
        {
        BYTE PB_SpecialFunction ;
        #define PB_SPCFUNC_USECUR   0x01
        #define PB_SPCFUNC_TRKONLY 0x02
        #define PB_SPCFUNC_SECSAME 0x04
        BYTE PB_DeviceType ;
        #define PB_DEVTYPE_0320 0x00
        #define PB_DEVTYPE_0360 0x00
        #define PB_DEVTYPE_1200 0x01
        #define PB_DEVTYPE_0720 0x02
        #define PB_DEVTYPE_SD8I 0x03
        #define PB_DEVTYPE_DD8I 0x04
        #define PB_DEVTYPE_FXDK 0x05
        #define PB_DEVTYPE_TPDR 0x06
        #define PB_DEVTYPE_1440 0x07
        #define PB_DEVTYPE_OTHR 0x08
        WORD PB_DeviceAttribute ;
        #define PB_NOREMOV 0x0001
        #define PB_DRLOCK   0x0002
        WORD PB_Cylinders ;
        BYTE PB_MediaType ;
        #define PB_MEDTYPE_1200 0x00
        #define PB_MEDTYPE_0320 0x01
        #define PB_MEDTYPE_0360 0x01
        WORD PB_BytesPerSector ;
        BYTE PB_SectorsPerAllocationUnit ;
        WORD PB_ReservedSectors ;
        BYTE PB_FATS ;
        WORD PB_RootDirectoryEntries ;
        WORD PB_TotalSectors ;
        BYTE PB_MediaDescription ;
```

```
        WORD PB_SectorsPerFAT ;
        WORD PB_SectorsPerTrack ;
        WORD PB_Heads ;
        DWORD PB_HiddenSectors ;
        BYTE PB_Reserved[10] ;
        WORD PB_SectorsInTrack ;
        struct
            {
            WORD Number ;
            WORD Size ;
            } PB_SectorTable[18] ;
        } PARAMETER_BLOCK ;
typedef PARAMETER_BLOCK FAR *LPPB ;

typedef struct
        {
        BYTE PBF_SpecialFunction ;
        WORD PBF_HeadNumber ;
        WORD PBF_CylinderNumber ;
        } PARAMETER_BLOCK_FORMAT ;
typedef PARAMETER_BLOCK_FORMAT FAR *LPPBF ;

typedef struct
        {
        BYTE PBW_SpecialFunction ;
        WORD PBW_HeadNumber ;
        WORD PBW_CylinderNumber ;
        WORD PBW_SectorNumber ;
        WORD PBW_SectorCount ;
        LPBYTE PBW_TransferAddress ;
        } PARAMETER_BLOCK_WRITE ;
typedef PARAMETER_BLOCK_WRITE FAR *LPPBW ;

typedef struct
        {
        BYTE PBR_SpecialFunction ;
        WORD PBR_HeadNumber ;
        WORD PBR_CylinderNumber ;
        WORD PBR_SectorNumber ;
        WORD PBR_SectorCount ;
        LPBYTE PBR_TransferAddress ;
        } PARAMETER_BLOCK_READ ;
typedef PARAMETER_BLOCK_READ FAR *LPPBR ;
#endif
#pragma pack()
```

Note the pack(1) pragma. Otherwise the C compiler will align the word

variables on an even boundary and the mapping will fail.

IOCTL Get Drive Parameters

Set DS:DX to point to a full IOCTL Parameter Block, set CL to 60H, set the registers for subfunction 0DH and execute the interrupt.

IOCTL Set Drive Parameters

First use IOCTL Get Drive Parameters to prime an IOCTL Parameter Block. Then make the appropriate changes. These would normally include the device type, number of sectors per track, and total number of sectors. Also set the number of sectors in track in the word at offset 26H and follow it with a pair of words for each sector. The first word is the sector number starting with one and the second word of the pair is the number of bytes in the sector. It should always be 512 (200H). Set the special function field -- offset 00H -- to 05H (it seems to work). Point DS:DX to the parameter block. Set CL to 40H. Set up the other registers for subfunction 0DH and execute the interrupt.

IOCTL Read

First use IOCTL Set Drive Parameters to set the diskette drive to the right mode for the diskette to be read. Set the head, cylinder, and first sector field of an IOCTL Read Parameter Block to the value for the first sector to be read. Set the number of sectors field to the number of sectors to be read. Place the address of the input buffer in the Transfer address field. Point DS:DX to the parameter block. Set CL to 40H. Set up the rest of the registers for subfunction 0DH and execute the interrupt.

IOCTL Write

First use IOCTL Set Drive Parameters to set the diskette drive to the right mode for the diskette to be written. Set the head, cylinder, and first sector field of an IOCTL Write Parameter Block to the value for the first sector to be written. Set the number of sectors field to the number of sectors to be written. Place the address of the output buffer in the Transfer address field. Point DS:DX to the parameter block. Set CL to 41H. Set up the rest of the registers for subfunction 0DH and execute the interrupt.

IOCTL Format

First use IOCTL Set Drive Parameters to set the diskette drive to the right mode for the diskette to be formatted. Set the head and cylinder fields of an IOCTL Format Parameter Block to the value for the track to be formatted. Point DS:DX to the parameter block. Set CL to 42H. Set up the rest of the registers for subfunction 0DH and execute the interrupt.

SAB.INI File

The SAB.INI file is used to store information from one execution of the system for use by another execution of the system. The section of the SAB.INI file that is used by SAB Diskette Utility starts with a [SABDU]. The items stored in the file are:

Xpos=	upper left corner of window
Ypos=	upper left corner of window
Width=	width of window
Height=	height of window
LastSize=	normal, iconic, or maximized code
CompareCompleted=	number of completed compares
CompareCancelled=	number of cancelled compares
FormatCompleted=	number of completed formats
FormatCancelled=	number of cancelled formats
ReadCompleted=	number of completed reads
ReadCancelled=	number of cancelled reads
WriteCompleted=	number of completed writes
WriteCancelled=	number of cancelled writes
UserName=	name of user
UserCompany=	company name
UserAddress1=	street address line 1
UserAddress2=	street address line 2
UserCity=	state
UserZip=	zip code
UserTelephone=	telephone number
UserKey=	user key
RegKey=	registration key
RegType=	type of registration code
InstallTime=	time/date of initial installation (seconds from 01/01/70)
DriveA=	type of drive code
DriveB=	type of drive code
Timer1=	maximum timer value
Timer2=	minimum timer value
Spool=	disk spooling option code
Format=	format option code

Definitions

- Boot Sector
- File Allocation Tabel (FAT)
- Diskette
- Drive
- Sector
- Track
- Head
- Cylinder
- Enable
- Disable
- Memory Image
- Hard Disk Image
- Disk Spooling
- INI Files
- Format Mode

Diskette

A form of removable storage media -- sometimes also called a floppy disk. It consists of an outer protective envelop around a thin circular piece of magnetic media. It is inserted into a diskette drive that contains two sets of read/write heads -- one for the top layer of the magnetic media and one for the bottom layer. The read/write heads can only move along a single line from the outer edge of the diskette toward the center and back. The heads move in fixed increments. The diskette rotates in the drive and this allows the heads to access a circular section of the magnetic media for each position.

Sector

A sector is the basic unit of storage on diskettes. It consists of a single block of data -- usually 512 characters -- written or read as a group. The normal format of a diskette has the same number of 512 character sectors on each track. Sectors are first created on a diskette by formatting it. This must be done before data can be stored on the diskette.

Track

A track consists of the circular area that a read/write head can access from one position as the diskette revolves in the drive.

Head

A head is the electromagnetic device that reads/writes the magnetic patterns on the diskette. A diskette drive has two heads -- one for each side of the magnetic media.

Cylinder

A cylinder consists of the circular area that the read/write heads can access from one position as the diskette revolves in the drive.

Boot Sector

The boot sector is the first physical sector on the diskette. It is on the first cylinder on the side of the diskette accessed by the first head. It contains a parameter table that describes the physical structure of the diskette (number of sectors per track and number of cylinders) and its logical layout (reserved sectors, File Allocation Table (FAT) size, number of directory entries in the root directory, etc.). It also contains the "boot program". When an IBM compatible microcomputer starts it checks the A: drive for a diskette. If there is one the systems reads the boot sector into memory and begins executing the code in it. If the diskette has an operating system on it the boot sector will contain a program that will begin loading the operation system.

File Allocation Table (FAT)

The File Allocation Table (FAT) contains one entry for each logical cluster on a diskette. (A logical cluster on a diskette usually consists of one sector.) A file's entry in the directory will contain a pointer to the first cluster of the file. The corresponding entry in the FAT will contain a pointer to the next cluster of the file. The FAT entry for the last cluster of the file will contain hex FFs to indicate that there aren't any more. An entry in the FAT for an unallocated cluster will contain binary zeros.

Diskette Drive

A diskette drive is the device that the floppy diskette is placed into to read or write. It can be internal to the computer case or in a standalone case. The normal sizes for IBM compatible drives are 3 1/2 inches wide and 5 1/4 inches wide. Each drive has two read/write heads one of which is positioned on each side of the floppy diskette.

Menu Item Enable

A menu item is enabled if it respond to it's selection by generating a message to the application. Menu items that are enabled are dark in color.

Menu Item Disable

A menu item is disabled if it does not respond to it's selection by generating a message to the application. Menu items that are disabled appear gray.

Diskette Memory Image

If the user has not forced disk spooling of the diskette image and there is sufficient memory available the sectors read from the diskette will be stored in memory buffers. Each buffer will contain the contents of one track . The memory is obtained from Windows' global memory pool and must be locked before each use and unlocked after each use.
Diskette Memory Image

Diskette Hard Disk Image

If the user has forced hard disk spooling of the diskette image or there is insufficient memory available the system will store the sectors read from the floppy disk in a temporary field on the hard disk. The file will be created in the directory pointed to by the TEMP= environment variable.

Disk Spooling

See Diskette Hard Disk Image

INI Files

An INI file is a file used by a Windows' application to store data between executions. It can be accessed using the ReadPrivateProfileString and WritePrivateProfileString functions. The file would normally be created in the Windows directory.

Format Mode

The format mode is the mode in which a diskette will be formatted.
For 3 1/2 inch diskettes it is either High Density (1.44 MB in 2880 sectors) or Dual Density (720 KB in 1440 sectors).
For 5 1/4 inch diskettes it is either High Density (1.2 MB in 2400 sectors) or Dual Density (640 KB in 1280 sectors).