WINDISK: A File and disk Editor

Table of content

Windisk is an Windows 9x/ME/NT4/2000/XP/Vista application.
ME stands for Millennium Edition of **Windows 98**.

Some of the implemented functions include:

- Display of disk geometry (real or simulated, depending on the system).
- Loading Disk Raw sectors and patching them. Saving them to a file or loading them from a file.
- Loading Disk Partition Sectors and patching them.Saving them to a file or loading them from a file.
- Loading file and browsing it.
- Patching of the loaded file.
- Saving the new data to the same file or to another file name.

HTML Help **cannot run** form the application if the application runs from shared network.

You are now looking at the **WINDISK** help. Last change: 2007/09/14

Windisk Introduction

Caution:

Windisk should be used with care, because some operation may be fatal to your file or system if you use it and save data to file or disk without knowing what you are doing. Editing a system file can **damage** your system. Saving data to a disk sector can also damage your system. So, if you don›t know, view data, files and sectors, but don›t save to them if you are not sure.

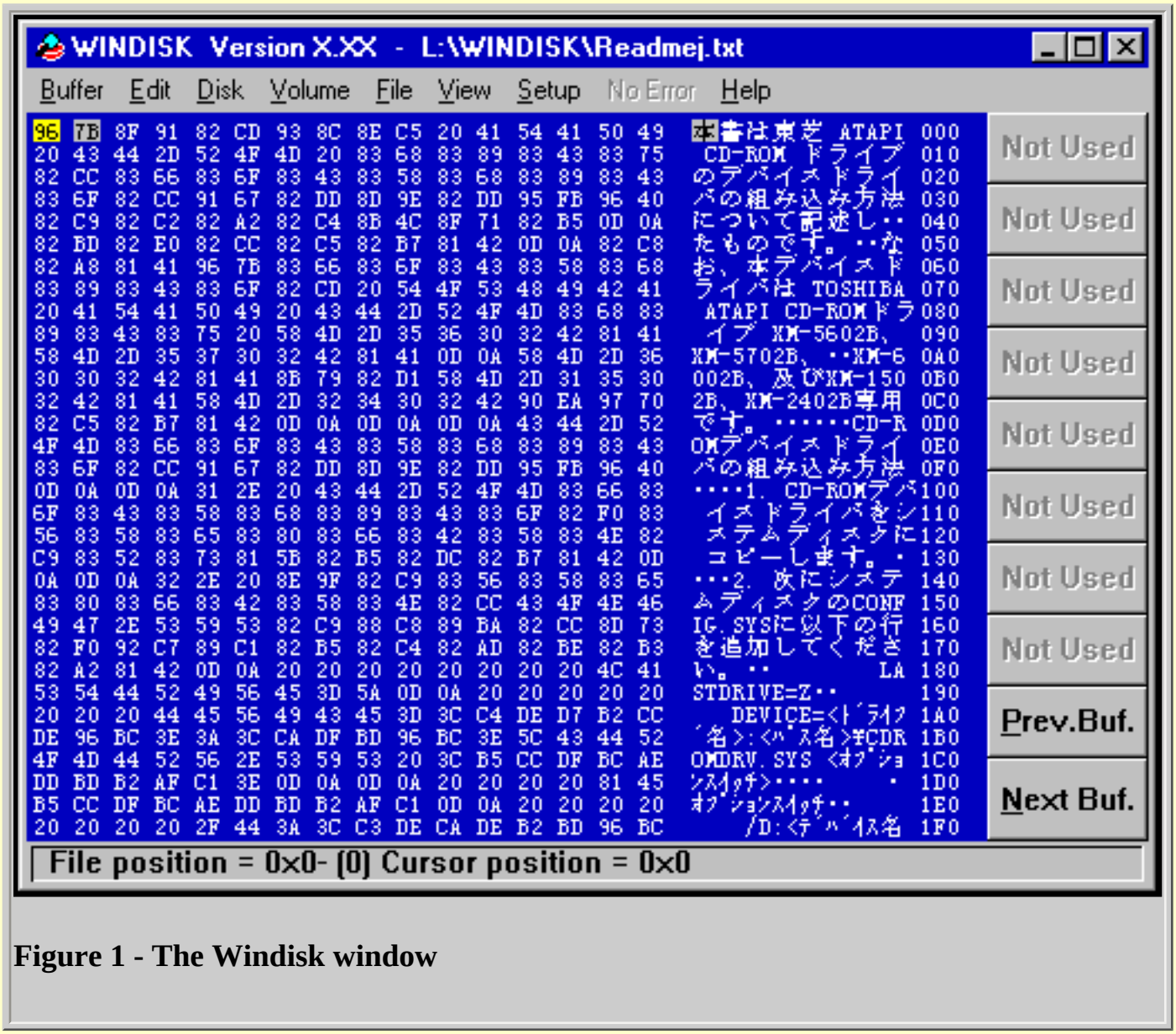


Figure 1 - The Windisk window

Contextual Help:

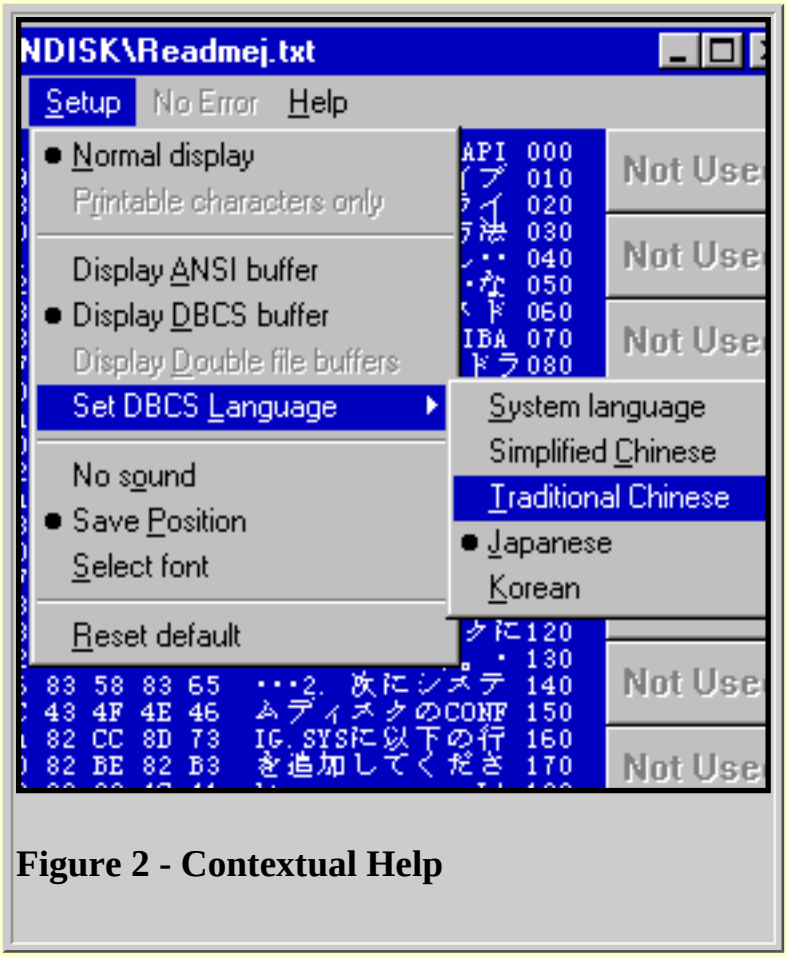


Figure 2 - Contextual Help

You can get **contextual help** from most menu item. Just select a menu item with either the **Alt key** and the **arrow-keys**, or with the mouse button, and press **F1**.

Program modes:

This program can do:

- File edition
- Disk sector edition
- Disk partition viewing
- Disk geometry
- Volume edition

The menu commands and buttons are only active for some of these operations. Some operations may have limitations due to the operating system. So you will see some differences if you operate the program under Windows 9X or under Windows NT.

When operations depend on one of the above mode, you may be asked to first load a file or select a

disk or volume.

Some functions are not yet implemented, and the menu options are disabled for these functions.

Most of file system internal code is undocumented.

Windisk File load

Content

File editing

Windisk can be used to view and edit files.

Go to the file menu and press “Load File A”

A file open dialog is presented to you to select the file you want to see.

Be default, the program uses an **ANSI** code page, but if you run a **DBCS** enabled system, you can go to the “**Setup**” menu and select “**Display DBCS buffer**”, then select the language to **“Simplified Chinese”, “Traditional Chinese”, “Japanese”** or **“Korean”.**

If you are in Japan and want to display a Japanese file, you can just use the default **“System language”.**

If you are working on a Japanese enabled system and want to work with **Simplified Chinese** files, go to **setup** and under **“Setup”->”Set DBCS language”** set **“Simplified Chinese”.** This will work if your system supports this language.

Sector edition

Content

Disk sector edition:

To be in **Disk edition** mode, you must use the **“DISK”** menu option and select one of your **hard disk**.. Even if you have only one **hard disk**, this is necessary to switch to **disk edition** mode as opposite to file editing or volume editing.

Then you can use the **Next sector** button or **previous sector** button, **page up** or **page down** to increment or decrement disk **sector offset** from the beginning of the disk.

To go directly to one particular sector, you can use the **“Go to sector…”** menu option under **“Edit”**

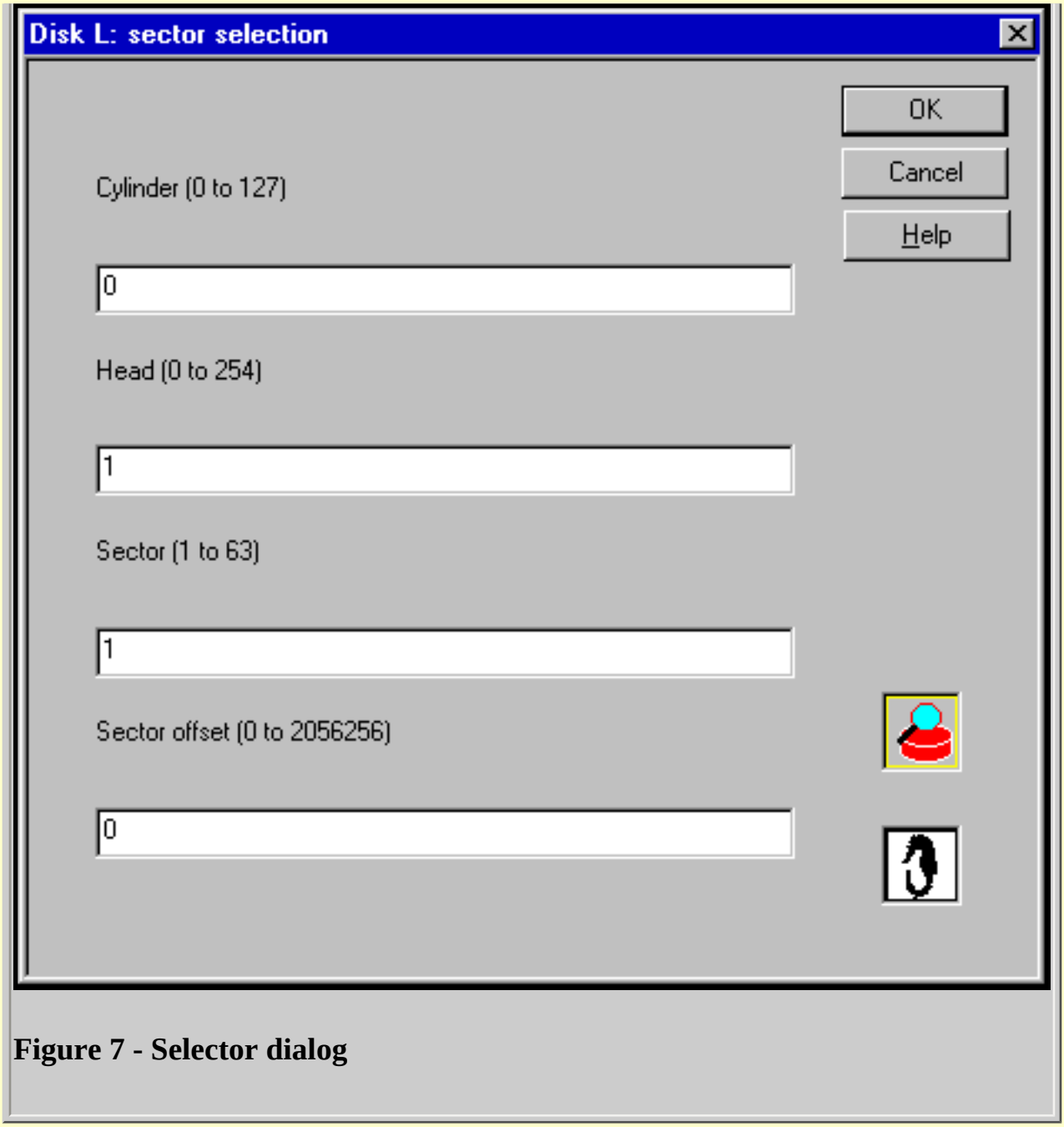


Figure 7 - Selector dialog

The Sector selection dialog provides four entry fields:

- **Cylinder** (0 to n)
- **Head** (0 to n)
- **Sector** (1 to n)
- **Sector offset** (o to n)

There is a unique **sector offset** value for any valid set of **Cylinder/Head/Sector** values on a particular disk.

Any **Sector Offset** value gives a unique value for **Cylinder, Head** and **Sector** on a particular disk.

You can type in any of the four fields, and the other fields are set accordingly.

The valid values depend on the emulated geometry of the disk being edited.

If you set a too large value in one of the fields, the maximum value is set automatically.

Clicking on Ok updates the current sector and shows it.

The PC BIOS (Basic Input Output System), your PC FIRMWARE, addresses the disk in Cylinder/Head/Sector mode. But the system has no need to know how the data is organized inside the disk. It is simpler to see it has a large linear memory with 512 byte blocks. This is accessed with Sector offset.

The beginning of the disk is offset 0 and any data is in a sector at a precise offset up to the end of the disk.

The last disk sector is at (Total Sector – 1) if the disk capacity is (Total Sector * 512)

Partition information

Content

Partition information:

This dialog shows you some values:

- Number: This is the order in which the partitions are seen by the system. Then the system will assign letters to the accepted partitions:
  - First primary partitions from any hard disk,
  - Then the secondary partitions.
- Types:
  - **0x01** - < 16 meg FAT partition
  - **0x02** – **XENIX** partition
  - **0x04** – 16 to 32 meg partition
  - **0x05** – extended partition (holder for logic partitions)
  - **0x06** – **huge** DOS V4 partition
  - **0x07** – **HPFS** partition
  - **0x08** – BOOT **AIX** partition
  - **0x09** – **AIX** partition
  - **0x0A - Boot manager** (OS/2)
  - **0x0B** – **FAT32** up to 2047 GB
  - **0x0C** – **FAT32** with **LBA** int13 extension (logical block address)
  - **0x0E** – huge with **LBA** int13 extension
  - **0x0F** – extended with **LBA** int13 extension
  - **0x11** – 0x01 type **hidden**
  - **0x12** – 0x02 type **hidden**
  - **0x14** – 0x04 type **hidden**
  - **0x15** – 0x05 type **hidden**
  - **0x16** – 0x06 type **hidden**
  - **0x17** – 0x07 type **hidden**
  - **0x18** – 0x08 type **hidden**
  - **0x19** – 0x09 type **hidden**
  - **0x35** – **OS:2 LVM** Partition
  - **0x41** – **Power PC** boot partition
  - **0x42** – Windows 2000 LDM (Logical Disk Manager) Partition
  - **0x63** – **UNIX** partition
  - **0x80** – **NTFS** partition
  - **0x82** – **LINUX** swap partition
  - **0x83** – **LINUX** native partition

    - **0xA0** – Laptop Hibernation Partition
    - **0xA5** – UNIX **FreeBSD** file system partition
    - **0xA6** – OpenBSD
    - **0xA9** – NetBSD
    - **0xC0** – **NTFS** partition
    - **0xFE** – IBM **diagnostic** partition
- Start offset: offset of the beginning of the partition in sectors
- Length – length of the partition in sectors
- Hidden sectors – unusable sectors before the start of the partition
- Boot indicator: The boot partition is the first partition of the first disk with this indicator on.

Disk geometry dialog

Content

How to:

To display your system disk geometry, click on the "**View**" menu and select a disk.

Windows 9x:

Under Windows 9x, you may see a two-column dialog:

- The **Physical** geometry column shows the real disk geometry.
- The **Simulated** geometry column shows the disk geometry as seen by the system.

If the Physical column shows all zero value, your system was not able to see the real disk geometry.

Windows NT/2000/XP/Vista:

Under Windows NT/2000/XP/Vista, only the **simulated** geometry is available.

Windisk File save

Content

Saving:

There are two ways for saving the edited file:

- Save to the original file. This is NOT the recommended method, because, you may want to return to the original file.
- Save to another file (creating a file with another name, or erasing an existing name of the same name as the new one).

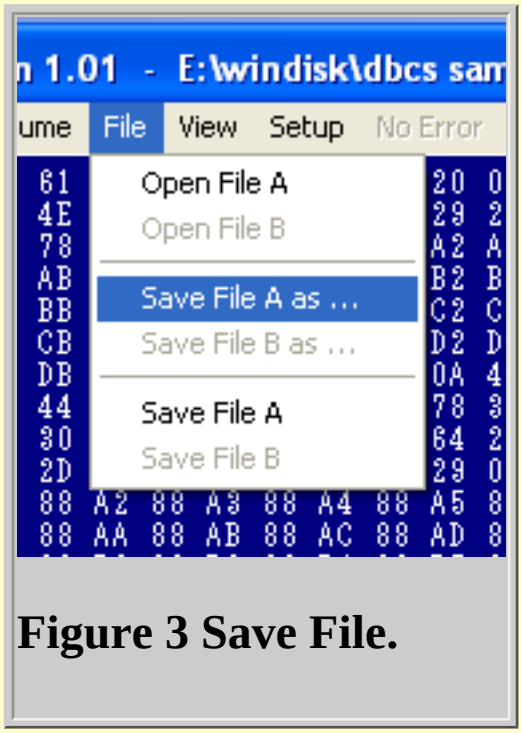


Figure 3 Save File.

DBCS files

Content

DBCS (Double byte character set).

DBCS is part of **MBCS** (Multiple byte character set) and is used in ***China***, ***Japan*** and ***Korea***. **DBCS** is different from **UNICODE**.

DBCS describes a character set where characters can be defined by *one* or *two* byte data (The **"Leading byte"** and the **"Ending byte".**

If your system supports **DBCS**, you can use the **DBCS** option in the "***Setup***" menu and select a **DBCS** language if your file has a different language than your system.

Displaying **DBCS** in tabular form is rather complex. So this program uses the following trick to accommodate evenly and oddly aligned **DBCS** characters:

If the **16th** character is a valid or not valid **DBCS** character ***leading*** byte, the character is written in the **16th** and **17th** column. So on the next line, the ***ending*** byte of the **DBCS** character is not shown. Clicking on this particular area will select the corresponding character.

So you see the right display column on 17 positions, but character are only shown once. Editing the last or fist position of a buffer may require using the previous or next buffer, some **DBCS** character being split across two buffers. The real buffer is **513** bytes long so that you can see the last character if this is a double byte character. The first byte is not displayed correctly if it is the ***ending*** byte of a **DBCS** character.

Do not use the **DBCS** option with **SBCS** files or binary files, because, if your system has support for **DBCS** it will search for **DBCS** characters and will find invalid **DBCS** sequences. This may produce unpredictable display and operations

SBCS files

Content

SBCS (Single byte character set).

SBCS as opposed to **DBCS** define character set where any character is defined by a single byte data.

Do not use the **DBCS** option with **SBCS** files or binary files, because, if your system has support for **DBCS** it will search for **DBCS** characters and will find invalid **DBCS** sequences. This may produce unpredictable display and operations.

The setup should be set to **ANSI** to get **SBCS** display.

Saving the view to a file

Content

Save view to file:

This save a view to a file as if it was displayed in **ANSI (SBCS)**mode.

This is not very different that printing to a file with the Generic (Text only) printer.

The output is limited to the file size though.

DBCS language selection

Content

Language selection:

DBCS string display relies on the selected locale language.

By default, when you select the **DBCS** option, the system locale language is used, but you may want to display a file which language is not your system default language.

Be default, the program uses an **ANSI** code page, but if you run a **DBCS** enabled system, you can go to the "**Setup**" menu and select "**Display DBCS buffer**", then select the language to **"Simplified Chinese", "Traditional Chinese", "Japanese"** or **"Korean".**

If you are in Japan and want to display a Japanese file, you can just use the default **"System language".**

If you are working on a Japanese enabled system and want to work with **Simplified Chinese** files, go to **setup** and under **"Setup"->"Set DBCS language"** set **"Simplified Chinese".** This will work if your system supports this language.

Print view

Content

Print view:

Printing the program view is done through the system printing system. You can use any graphic printer available on your system.

You can also print to file and send the file to another printer (either on your network or not.).

If you use the Generic (text only) printer and print to file, you can import the output file in your word processor.

If you want to use a text only printer, be sure to set the program setup to **ANSI**and **Printable characters only**

The **Font** used for printing is the font used for displaying the data. Only **fixed pitch** fonts are available for displaying data.

The font size is not selectable when printing the view. Font size is computed with respect to paper size and paper orientation.

Printer setup

Content

Printer setup:

This is the standard system printer setup tool.

Save active buffer to a file

Content

Save active buffer to a file:

This saves the viewed buffer to a file (512 byte).

If the file already exists, you are asked if you want to over-write it.

If the buffer has never been saved, you are prompted for a file name and path.

Append active buffer to a file:

The file must exist to be appended. You may need to create it by first **“Saving as…”** to a file before appending.

DISK editing:

Windisk can be used to view and edit disk or volume sectors

When in disk edition mode, you can set your editor at a particular sector offset with the “Go to sector…” option under “Edit” menu.

Then you can replace the sector data with data from a file on your disk or from another disk or diskette. **You must realize however that this can be a problem if you take the data from the same disk area your are editing. Writing data to a hard disk with a program like WINDISK can be very dangerous for your system integrity, if you do not understand exactly what you are doing. Do not change any sector used by your operating system or any running application.**

WINDISK uses direct disk addressing to read and write sectors, but WINDISK also uses the operating system to load/save file and sector to disk as files. These are incompatible operations if the system cannot be aware of direct disk access operations. Only disk-locking operation allows this, but an application cannot lock a disk with opened files on it. This is the reason why applications doing direct disk access are much safer if run in DOS.

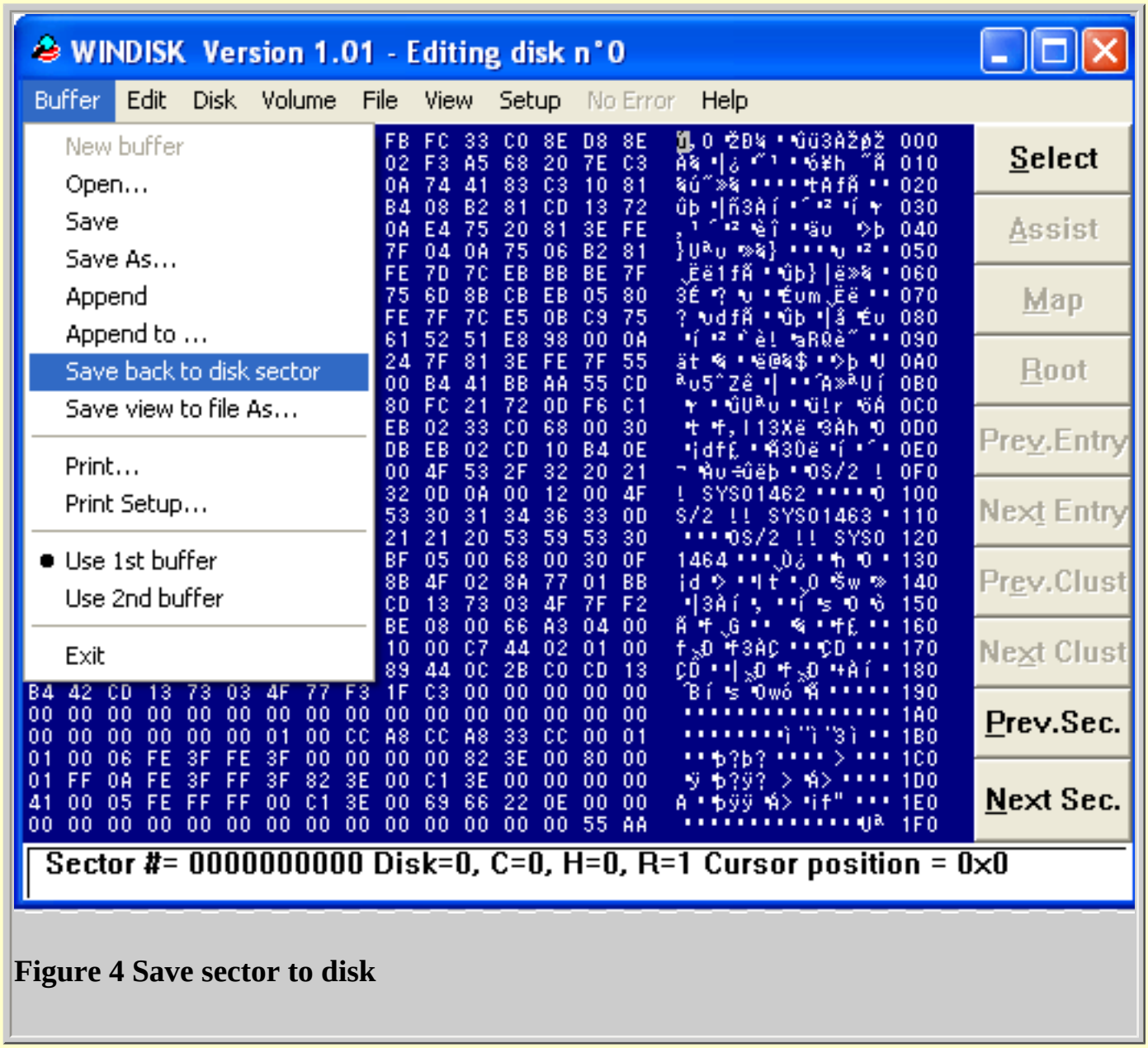


Figure 4 Save sector to disk

Save the active buffer to a new file

Content

Save active buffer to a new file:

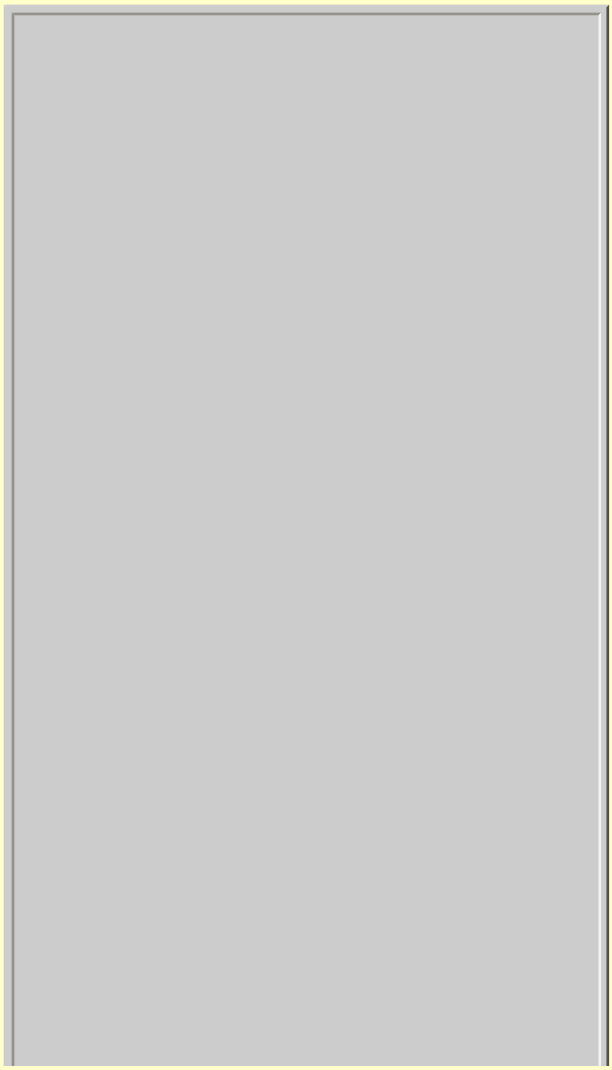
This saves the viewed buffer to a file (512 byte).

If the file already exists, you are asked if you want to over-write it.

You are always prompted for a file name and path.

Append active buffer to a file:

The file must exist to be appended. You may need to create it by first **“Saving as…”** to a file before appending.



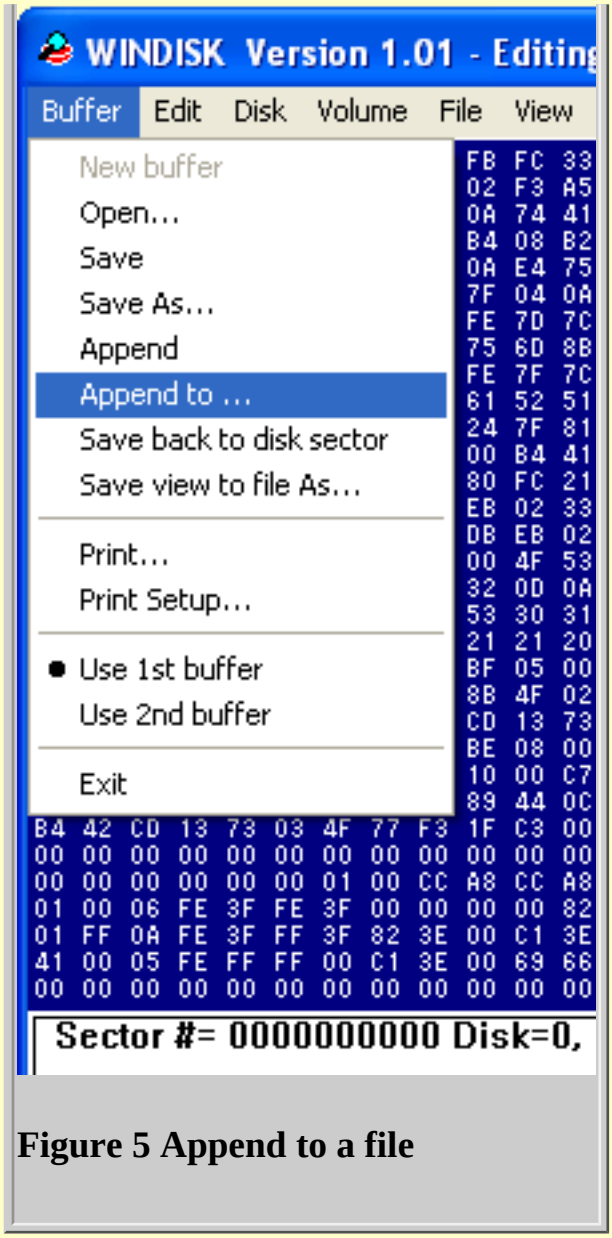


Figure 5 Append to a file

DISK editing:

Windisk can be used to view and edit disk or volume sectors

When in disk edition mode, you can set your editor at a particular sector offset with the “Go to sector…” option under “Edit” menu.

Then you can replace the sector data with data from a file on your disk or from another disk or diskette. **You must realize however that this can be a problem if you take the data from the same disk area your are editing. Writing data to a hard disk with a program like WINDISK can be very dangerous for your system integrity, if you do not understand exactly what you are doing. Do not change any sector used by your operating system or any running application.**

WINDISK uses direct disk addressing to read and write sectors, but WINDISK also uses the operating system to load/save file and sector to disk as files. These are incompatible operations if the system cannot be aware of direct disk access operations. Only disk-locking operation allows this, but an

application cannot lock a disk with opened files on it. This is the reason why applications doing direct disk access are much safer if run in DOS.

Open saved buffer

Content

Open saved buffer:

This loads the **current** buffer with data from a file (*usually a previously saved buffer, but this can be any 512 byte long data.).*

If the file exists, its data **replace** the current buffer data..

You are always prompted for a file name and path.

There must be a current buffer to enable this operation, so you cannot open a file if you are not editing a file or a disk sector.

If the opened file is **shorter** or **longer** that 512 byte, the data is read up to its end or up to 512 byte, whichever occurs first.

If you are editing a **DBCS** file, the 513-rd byte is cleared.

When the buffer is partially displayed (end of file in the buffer), the length of the part displayed is not changed by this operation.

Edit color convention

Content

Edit color convention:

Colors are fixed in this program.

- Background is deep **Blue**.
- Selected position is **Yellow** if the position has not been changed.
- Selected position is **Red** if the position has been changed.
- Matching selected position (on the other side of the window) is **Gray**.
- Changed positions are **Green**.

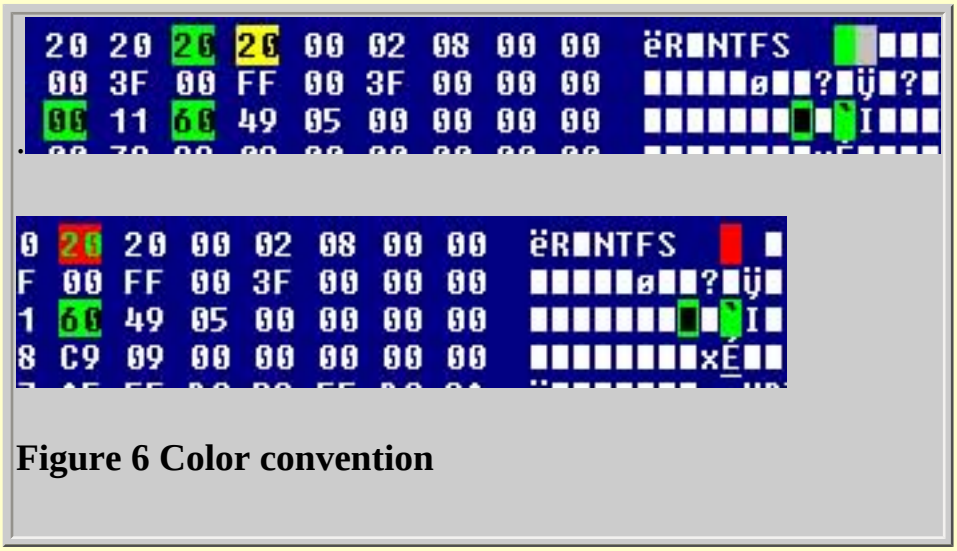


Figure 6 Color convention

Working with two buffers

Content

Working with two buffers:

The program provides two working buffers:

- Buffer 1 is the default buffer.
- Buffer 2 is the alternate buffer.

When the program starts Buffer 1 is the active buffer.

You may need to get data in the alternate buffer and then insert it in the current edition.

For example, you can get the **DISK D:** boot sector, patch it to match **DISK C:** geometry, then edit **disk C:** and save it to **disk C**:

To get this done:

- Activate ***Buffer 2***
- Edit **DISK D:**
- Load boot sector in Buffer 2
- Patch data in the Bios Parameter Block
- Select ***Buffer 1***
- Edit **DISK C:**
- Point to the Boot sector
- Select ***Buffer 2***
- Save buffer to **disk. C:**

Shift first char (DBCS disk buffer)

Content

Shift first valid DBCS char (disk editing):

Unlike file, where the file is first scanned to get a bitmap of valid **DBCS** characters, the disk is accessed at random. So when you look at a sector data, if these data represent a **DBCS** string, there is no way to predict if the buffer first byte represents a valid **DBCS** character.

If the first byte is not a valid character, the second byte may be a valid **DBCS** leading byte.

The shift menu option tells the program to:

- Use the **second** byte as valid **DBCS** character when ***checked***.
- Use the **first** byte as valid **DBCS** character when ***unchecked***.

Saving sector edition

Content

Saving disk sector edition:

When you have done your required changes in the sector edition window, you can save back the data to the disk.

Before saving your data, keep in mind that this is dangerous work, and that other processes can change this sector. You are working in a multi-processing environment.

Writing data to a hard disk with a program like WINDISK can be very dangerous for your system integrity, if you do not understand exactly what you are doing. Do not change any sector used by your operating system or any running application.

Element properties

Content

Display element properties:

This shows you the different **properties** attached to the **element** being edited.

This can be done for a **disk**, a **volume** or a **file**.

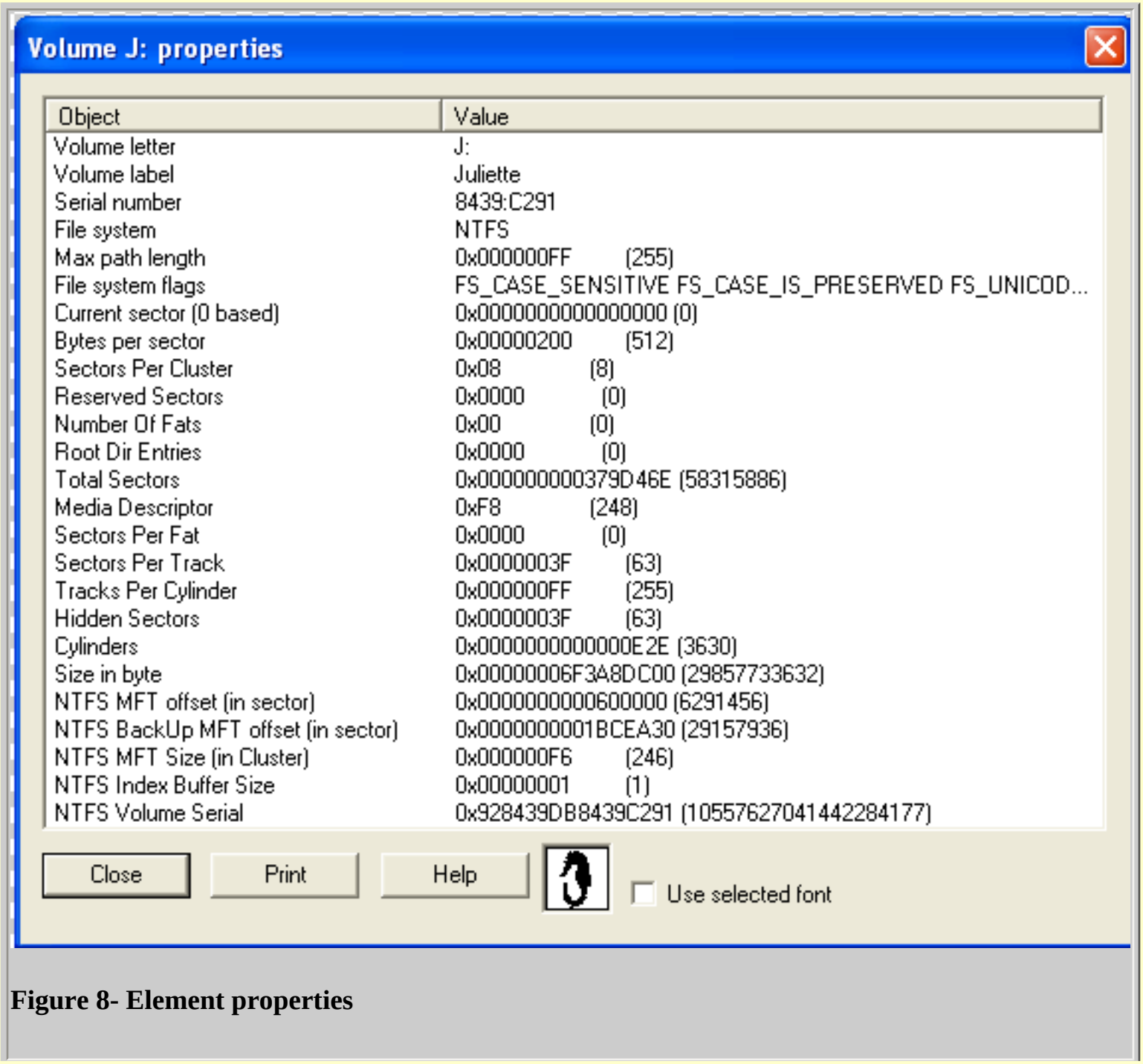


Figure 8- Element properties

Printing properties:

Printing is done using the font you selected to view data in the program window. The font size also is used for printing the properties.

You can also use this font to view the properties by checking the check-box button.

Viewing element properties:

The program includes three programming interface families:

- One for Windows 95 before OSR/2
- One for Windows 95 with OSR/2 and above (including Windows 98)
- One for Windows NT 4.00

This impacts volume properties.

Volume properties can show some of these values (depending on the system and on the volume file system).

Volume file system can be:

- FAT12
- FAT16
- FAT32
- NTFS

Volumes formatted with HPFS or any of the UNIX file systems are not known to Windows interface.

Volume properties:

- *Volume letter: C:*

  The name given by the system to access it.
- *Volume label: My Label*

  The volume label.

- *Serial Number: 211B:1AD1*

  A double word hexadecimal number related to the time the volume was created.
- *File system: FAT32*

  One of the above supported file system.
- *Max path length: 255*

  Maximum length of the path name element supported by a call to the system.
- *File system flags: FS_CASE_IS_PRESERVED*

  Some volume file system properties.
- *Current sector:12345*

  Current edition current sector.
- *Bytes per sector: 512*

  Sector size in byte.
- *Sector per cluster: 8*

  Minimum disk allocation size in sector.
- *Reserved sectors: 1*

  Sectors reserved by the file system.
- *Number of FATs: 2*

  Number of File Allocation Tables (for FATxx file systems).
- *Root Dir Entries:512*

  Maximum number of entries in the root directory of FAT12 and FAT16 file systems.
- *Total sector:2056257*

  Volume size in sector.
- *Media descriptor: F8*

  Mainly used in the DOS 1.10 time to specify diskette size. Maintained for compatibility.
- *Sector per FAT: 251*

  Size of the File Allocation Table (FAT).
- *Sectors per track: 63*

  Emulated disk geometry for the partition.
- *Track per cylinder: 255*

Emulated disk geometry for the partition

- *Hidden sectors: 63*

Space used by the partition table, multi-boot manager, program protection and viruses. Could be edited as a Disk element.

- *Cylinders: 128*

Emulated disk geometry for the partition

- *Size in byte: 1052803584*

Total sector * Bytes per sector.

- *Flags: PROT_MODE_LOGICAL_DRIVE*

Volume device driver flags (running in real mode DOS or protected mode).

- *Int13h address: 0x80*

Drive BIOS address hosting the logical drive (volume).0x80 is the first drive, 0x81 the second one etc…

Diskette number are 0x00 and up.

- *Associated drive map: 0x0000*

Drive mapping flags (if any).

- *Drive RBA offset (in sector):4128768*

Volume starting sector within the drive (Relative Block Address).

FAT32 data:

- *dpb_drive*

The drive number (0 = A, 1 = B, and so on).

- *dpb_unit*

Specifies the unit number. The device driver uses the unit number to distinguish the specified drive from the other drives it supports.

- *dpb_sector_size*

The size of each sector, in bytes.

- *dpb_cluster_mask*

The number of sectors per cluster minus 1.

- *dpb_cluster_shift*

The number of sectors per cluster, expressed as a power of 2.

- *dpb_first_fat*

The sector number of the first sector containing the file allocation table (FAT).

- *dpb_fat_count*

The number of FATs on the drive.

- *dpb_root_entries*

The number of entries in the root directory.

- *dpb_first_sector*

The sector number of the first sector in the first cluster.

- *dpb_max_cluster*

The number of clusters on the drive plus 1. This field is undefined for FAT32 drives.

- *dpb_fat_size*

The number of sectors occupied by each FAT. The value of zero indicates a FAT32 drive. Use the value in extdpb_fat_size instead.

- *dpb_dir_sector*

The sector number of the first sector containing the root directory. This field is undefined for FAT32 drives.

- *dpb_media*

Specifies the media descriptor for the medium in the specified drive.

- *dpb_next_free*

The cluster number of the most recently allocated cluster.

- *dpb_free_cnt*

The number of free clusters on the medium. This field is 0FFFFh if the number is unknown.

- *extdpb_free_cnt_hi*

The high word of free count.

- *extdpb_flags*

Flags describing the drive. The low 4 bits of this field contain the 0-based FAT number of the Active FAT. This field can contain a combination of the following values.

Value Description

BGBPB_F_ActiveFATMsk (000Fh) Mask for low four bits.

BGBPB_F_NoFATMirror (0080h) Do not Mirror active FAT to inactive FATs.

Bits 4-6 and 8-15 are reserved.

- *extdpb_FSInfoSec*

  The sector number of the file system information sector. This field is set to 0FFFFh if there is no FSINFO sector. Otherwise, this value must be non-zero and less than the reserved sector count.
- *extdpb_BkUpBootSec*

  The sector number of the backup boot sector. This field is set to 0FFFFh if there is no backup boot sector. Otherwise, this value must be non-zero and less than the reserved sector count.
- *extdpb_first_sector*

  The first sector of the first cluster.
- *extdpb_max_cluster*

  The number of clusters on the drive plus 1.
- *extdpb_fat_size*

  The number of sectors occupied by the FAT.
- *extdpb_root_clus*

  The cluster number of the first cluster in the root directory.
- *extdpb_next_free*

  TheSeaHorse number of the cluster that was most recently allocated.

BPB (FAT32 & FAT):

The BPB for FAT32 drives is an extended version of the FAT16/FAT12 BPB. It contains identical information to a standard BPB, but also includes several extra fields for FAT32 specific information.

- *BPB_BytesPerSector*

  The number of bytes per sector.
- *BPB_SectorsPerCluster*

  The number of sectors per cluster.
- *BPB_ReservedSectors*

  The number of reserved sectors, beginning with sector 0.

- *BPB_NumberOfFATs*

  The number of File Allocation Tables.
- *BPB_RootEntries*

  This field is ignored on FAT32 drives.
- *BPB_TotalSectors*

  The size of the partition, in sectors.
- *BPB_MediaDescriptor*

  The media descriptor. Values in this field are identical to standard BPB.
- *BPB_SectorsPerFAT*

  The number of sectors per FAT.

  Note: This field will always be zero in a FAT32 BPB. Use the values from ***BPB_BigSectorsPerFat*** for FAT32 media.
- *BPB_SectorsPerTrack*

  The number of sectors per track.
- *BPB_Heads*

  The number of read/write heads on the drive.
- *BPB_HiddenSectors*

  The number of hidden sectors on the drive.
- *BPB_BigTotalSectors*

  The total number of sectors on the FAT32 drive.
- *BPB_BigSectorsPerFat*

  The number of sectors per FAT on the FAT32 drive.
- *BPB_RootDirStrtClus*

  The cluster number of the first cluster in the FAT32 drive's root directory.
- *BPB_FSInfoSec*

  The sector number of the file system information sector. The file system info sector contains a BIGFATBOOTFSINFO structure. This field is set to 0FFFFh if there is no FSINFO sector. Otherwise, this value must be non-zero and less than the reserved sector count.
- *BPB_BkUpBootSec*

The sector number of the backup boot sector. This field is set to 0FFFFh if there is no backup boot

sector. Otherwise, this value must be non-zero and less than the reserved sector count.

Edit Root directory sectors

Content

Going to the root directory first sector:

This will display the ROOT directory first sector.

FAT12 & FAT16:

On this file system, the root directory is placed at a fixed place after the two FATs.

Use the assist tool to edit the directory entries

FAT32:

On this file system, the root directory is just an ordinary file and can be placed anywhere on the disk.

Use the assist tool to edit the directory entries

NTFS:

This file system place the directory in a file table ($MFT) and a backup file table.

HPFS:

This file system is no more supported under Windows NT 4. You have to convert it to FAT or NTFS with the Windows NT **"convert"** utility.

The **HPFS** file system is built on a linked list data structure pointed to by a **"super block"**. Data block are written in 2 mb blocks separated by bitmaps. Bitmaps show used and unused sectors.

Working with FATxx Directories:

When the **root directory** is loaded in the editor, a **red vertical bar** is displayed to show you where you can **click** to select an entry.

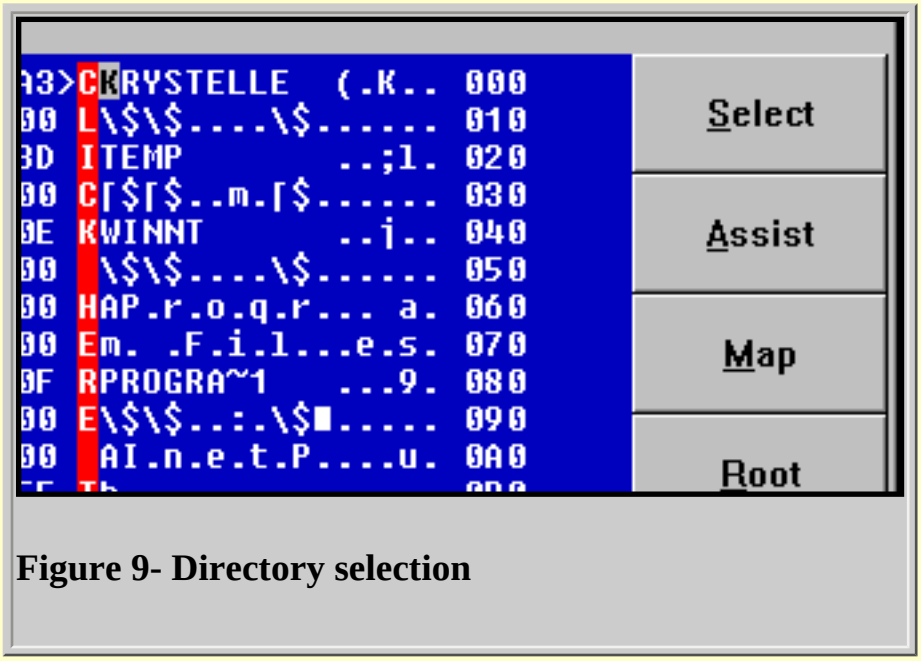


Figure 9- Directory selection

The selected entry is indicated by the **">"** sign in front on it.

If you press the **"Assist"** button, the directory entry is expanded in the directory dialog.

If you **double-click** on the selection bar, the sector editor displays the data the selected entry point to.

You can use the **"Next Sector"** and **"Previous Sector"** buttons to navigate within directory or file.

In directory there is a **"Go to previous sector"** option in the **Edit menu**. So you have a one step backward option.

FAT directory assist

Content

Editing directory entry using the assist tool:

This tool is available for FAT file system.

Displayed data:

- **Type** edit field is computed from the attribute bits and the first directory entry byte.
- **NAME** field is copied from the byte 0 to 10 of the directory entry when applicable.
- **Attribute** bits explode the attribute byte.
- **Creation** date, **modification** date and **last access** date are directly copied from the directory entry. Sometimes, the creation date may be after the modification date, because only the system handle these fields, and DOS programs only know the creation date.
- **B12** (Byte 12) is used by Windows NT to store names in lower case:
  - **0x00** means all upper case
  - **0x10** means filename in lower case and extension in upper case.
  - **0x08** means filename in upper case and extension in lower case
  - **0x18** means filename and extension in lower case.
- **B13** is a sequence number
- **Cluster** is the element first cluster.
- **OS/2 EA ordinal** is the OS/2 filename attached ordinal to point to the EA DATA. SF file.

  This entry is over-written by the **FAT32** file system to handle 32 bit cluster entry.
- **Lfn part** is a part of the long file name stored in the directory entry Long File Name extension (VFAT and FAT32).

Size in byte is the element size in byte.

Available functions:

- **[Ok]** Quit the dialog without saving change to data
- **[Cancel]** Quit the dialog without saving change to data
- **[Print]** Print the displayed data.
- **[Help]** This Assist help page.
- **[Previous entry]** Go to directory previous entry (same as main window button).
- **[Next entry]** Go to directory next entry (same as main window button).
- **[Previous Sector]** Go to directory previous sector (same as main window button).
- **[Next entry]** Go to directory next entry (same as main window button).
- **[Root]** Go to top of root directory (same as main window button).
- **[Parent entry]** A one shot button allowing return to home directory for the current entry when this is possible.
- **[Map]** Map the current entry (cluster map) The map can be printed from the buffer menu. See Map function
- **[Save changes to disk]** When changes have been made to edit field or check box, save the change to disk (WARNING you must know what you are doing…)

Note:

Some functions are not provided in this assist:

- **Erase an entry:** This is somewhat complex because not only the entry require a 0xE5 byte in the first position, but the corresponding FAT entries should be freed by putting 0xFFFF, 0xFFF or 0x0FFFFFFF, in any FAT pointer in the FAT chain. So it is much simpler and sure to use the file system for this function.
- **Change an entry from file name to vFAT extension:** Documentation on the B13 entry would be required. Without a valid value in bye 13, the system will not recognize the entry.
- **Setting first byte in vFAT extension:** The values are 0x01, 0x02 … up to 0x4n for the nth and last entry. You can use the direct buffer edition for this setting.

MAP drive cluster allocation

Content

Showing drive occupation:

This option is used for FAT file system.

This map shows occupied clusters and gives an idea of drive fragmentation.

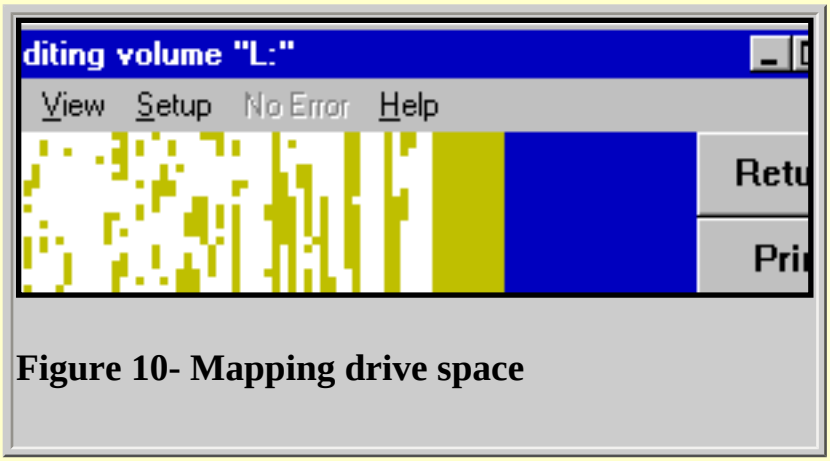


Figure 10- Mapping drive space

How to return from drive mapping:

There are three method to quit drive mapping:

- Click the **“Return”** button
- **Right click** on the map area
- Press the **ESC**ape key**.**

Technical data:

Display map:

This tool shows disk occupation according to the **FAT**. So, this is only available for **FAT** formatted drives.

As there is a lot of clusters on modern drives any box element on the picture usually shows several

clusters. The number of clusters per box is computed from the window size and the number of clusters. The smallest screen box size if three per three pixels. So, if you **maximize** the application, the picture will be more detailed. If your media has defective (or marked defective by a virus) clusters, they will be shown in red.

The screen map is filled from the begin of the disk to the end in **vertical strips** from up to bottom and from left to right.

You can MAP the entire volume by default, but if you are in directory entry mode with the red vertical bar, and select an entry from the “ROOT”, you can MAP the selected object (even erased entry).

Print map:

If you print the MAP, it will be computed with a different number of clusters per box according to the printer definition. The smallest box is one 150^{th} of the paper width. So, **portrait printing** shows more details.

On printed map, defective clusters are shown as black boxes and a crossed box shows used clusters.

Printed map is filled in **horizontal strips** from left to right and from top to bottom.

FAT

Content

Going to FAT sectors:

This option is used for **FAT** file system.

FAT (**File Allocation Table**) represent a table of pointer to **DATA** or **DIRECTORY** cluster.

The size of the pointer depends on the maximum disk size.

- **FAT12** is used on diskettes and was used on the IBM PC/XT.

  Values 000, FF0 to FFF are special markers

  The maximum entry number is 4085

  The maximum cluster size is 4096 byte
- **FAT16** was introduced for more than 32 MB disk.

  Values 0000,FFF0 to FFFF are special markers

  The maximum entry number is 65519

  The maximum cluster size is 65536 byte

  This can support 4 GB partitions. Unfortunately, only Windows NT supports **4 GB** partitions.

  Other systems (**OS/2**, **DOS**, **Windows 3.x**, **95,98**) only support **2 GB** partitions in **FAT16**
- **FAT32** was introduced by Microsoft with the second release of **Windows 95 for OEM**. **Windows 2000** supports **FAT32** file system.

  FAT32 also includes **VFAT** long file name system kluge introduced by Microsoft for Windows 95.

  The two main drawbacks of **VFAT** are :

  - The limit of the number of directory entries in the root directory for non **FAT32**.

- The non permanent association of short file names with long file names, making backup restore impossible.

Table of Figures