Features

XVI32 is a free hex-editor with the following main features. Especially note the **highlighted** advantages.

- **Data Inspector** to show permanently decoding of numbers
- **Built-in XVI32script interpreter** allows to automate editing or share patches. More...
- Computing of **CRC16** (standard) and **CRC32** (PKZIP compatible) checksums for complete file and selected block (only if block is currently selected)
- **Easily works with huge files.** Try to open a 60 MB sized text file with some other hex editors (not to speak about Wordpad), then use XVI32...
- XVI32 allows to edit files **up to 2 GB** (enough virtual memory provided, of course)
- For your convenience, **XVI32 stores settings** and last used search strings etc. in XVI32.INI file
- **Progress indication in percent** for most operations
- You can **abort nearly all operations** (reading/writing files, search, replace, print...)
- Display of both text (ASCII/ANSI) and hexadecimal representation
- **Two synchronous cursors** in text and hex area
- **Wheel mouse support**
- Fully resizeable window (change number of rows and columns)
- Font and font size adjustable
- Overwrite or **insert** characters
- Insert text or hex string **n times**
- Switch byte offset (address) of first byte between **0 or 1** to examine also record structure of plain text files
- **Search text or hex string**, e.g. find "this text" or find "0D 0A"
- **Search optionally with joker (wildcard) char** that will match any character, e.g. find "A.C" or "00 2E 2E 00" where "." = "2E" (user-defined) stands for any character
- Fast searching algorithm **(Quicksearch)** for **both** search directions (down/up)
- **Count** occurences of text or hex string
- **Replace** text or hex string, e.g. replace "0D 0A" by "0A" or replace "0D 0A" by text "EOL"
- **Simplified search for Unicode Latin (UTF-16) strings**
- **Extremely fast "replace all"** mode (if needed, additional memory is allocated beforehead, not at every single replacing operation)
- **Auto-fill** feature to copy bytes from current address into input field for hex string using right arrow key
- **Character conversion using self-defined character table**
- Easy **converting of text to hex string** in dialogs (e.g. "abc" -> "61 62 63")
- **Decoding and encoding of 1, 2, 4, and 8 byte integers and floats** in 2 possible byte orders (optionally shown permanently by Data Inspector)
- **Bit manipulation** (view or set bits)
- Open file in **Read Only mode** (e.g. if opened by another application or to avoid unintentional modifications)
- **Insert file contents** into file

4 **Write block to file**
4 Copy, move or delete block
4 Clipboard support
4 Goto address **(absolute or relative up/down)**
4 **Bookmarks**
4 Enter jump width and jump up/down **(useful for files with fixed record length)**
4 **Patch BORLAND PASCAL 7.0 EXE files** for execution on processors > 200 MHz
4 **Printing with preview** or print to file
4 Easily access most recently used files
4 **No setup programm needed, doesn't write any data to registry**
4 And last, but not least: **XVI32 is free!**

chmaas@handshake.de
www.chmaas.handshake.de

XVI32 was built using Borland Delphi.

History

At the begin of the 90s, my first hex editor XVI was developed under TURBO PASCAL (XVI stands for the roman notation of the number 16). This was the ancestor of XVI32, limited to file sizes of 64 KB. In 1994, I made the step towards object oriented programming with BORLAND PASCAL 6.0 using TURBO VISION as user interface. At this time, I developed TVXVI, still a real mode program, but now able to edit files up to around 400 KB by using an array of 64 KB sized arrays. Both XVI and TVXVI were never released to the public.

From 1995 until now, I developed with DELPHI 1.0, 2.0 and 4.0. Of course there was immediately the idea to develop a 32 bit version of XVI, but it was not before mid 1998 when I began this project.

Licence Agreement and Copyright

IMPORTANT - READ CAREFULLY

This license and disclaimer statement constitutes a legal agreement ("License Agreement") between you (either as an individual or a single entity) and Christian Maas (the "Author") for this software product ("Software"), including any software, media, and accompanying on-line or printed documentation.

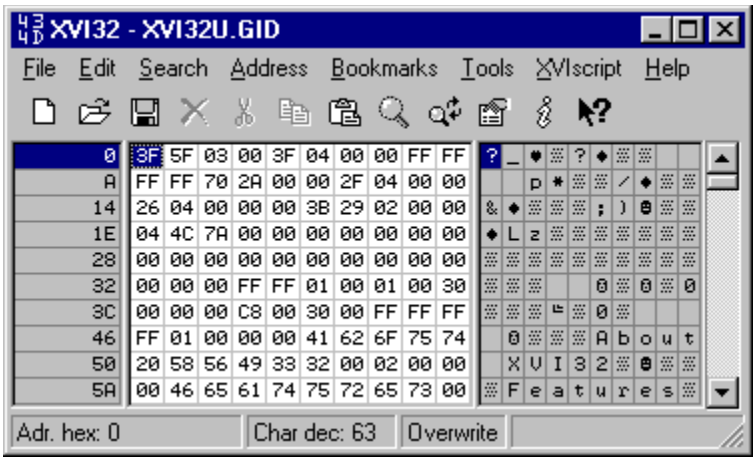
BY DOWNLOADING, INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE AND DISCLAIMER AGREEMENT.

1. This software is freeware. You can use this software royalty-free for private and commercial purposes.
2. You can freely distribute copies of the main archive as long as no alterations are made to the contents and no charge is raised except a reasonable fee for distributing costs.
3. You may not modify, reverse engineer, decompile, or disassemble the object code portions of this software.
4. This Software is owned by Christian Maas and is protected by copyright law and international copyright treaty. Therefore, you must treat this Software like any other copyrighted material (e.g. a book).
5. This software is provided "as is" and without any warranties expressed or implied, including, but not limited to, implied warranties of fitness for a particular purpose.
6. In no event shall the author be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use this software or documentation, even if the author has been advised of the possibility of such damages.
7. Any feedback given to the author will be treated as non-confidential. The author may use any feedback free of charge without limitation.

chmaas@handshake.de
www.chmaas.handshake.de

XVI32 Window

This is XVI32 windows after a file has been opened. Click on the picture to get more information.



Menus

There are the following main menu items:

File	The common file commands to open, save etc.
Edit	Block commands, deleting and inserting
Search	Search, count and replace
Address	Address related commands (e.g. goto...)
Bookmarks	Manage named bookmarks for up to 9 addresses
Tools	Switch text and overwrite mode, tool functions (encode/decode number, patch BP 7.0 executables) and options
XVIScript	Edit or execute XVIscript
Help	The common help menu

This is the toolbar providing quick access to some often used functions. You can turn displaying on and off using the menu `Tools|Options`

Menu "File"

New	Create a new file (EOF char generated depending on options settings)
Open	Open file If you check the appropriate box in the open dialog, the file will be opened in Read Only mode (e.g. to avoid unintentional modifications). This mode is also suggested if a file is already opened by another application. In Read Only mode, the `Save` menu and tool button are disabled, but you can still edit the file and use `Save As`
Reload	Open current file again
Reopen	Select a previously opened file to reopen it (history count depending on options settings)
Save	Save file
Save As	Save file under new name
Close	Close file
Insert	Insert contents of other file before current address
Write Block	Write the currently marked block (refer to menu edit) to file
Print	Print (with preview)
Exit	Exit XVI32

Note: Since version 2.30, only the Windows standard dialogs to open or save files are available. The XVI32 proprietary dialogs are no longer supported.

Menu "Edit"

Insert string	Enter text or hex string to be inserted <n> times
Overwrite string	Overwrite existing characters with a text or hex string
Block mark	Mark begin or end of block
Block <n> chars	Mark block consisting of <n> chars beginning at current address
Block unmark	Unmark current block
Block delete	Delete current block
Block copy	Copy block before current address
Block copy/overwrite	Copy block at current address overwriting existing chars
Block move	Move block before current address
Delete from cursor	Delete all chars from current address (including) until EOF
Delete to cursor	Delete all chars from begin of file until current address (including)
Clipboard	Access to the clipboard functions; special menu items are 4 "Copy as hex string" to copy selected block as hex string (e.g. "ABC" as "41 42 43") 4 "Paste from hex string" to paste a hex string like "41 42 43" or "414243"

Read how to mark a block.

Menu "Search"

Find	Find text or hex string
Find again down	Repeat last search operation downwards
Find again up	Repeat last search operation upwards
Count	Count occurrences of text or hex string
Replace	Replace text or hex string
Replace again down	Repeat last replace operation downwards
Replace undo	Undo last replace operation
Character conversion	Convert characters from one character set into another, e.g. from DOS to Windows, by using a user-defined character conversion table

Menu "Address"

Goto	Go to an absolute or relative address
Jump width	Enter a number of bytes to jump down or up using one of the following menu items
Jump down	Go previously entered number of bytes downwards
Jump up	Go previously entered number of bytes upwards
Remember address	Remember current address. You can also use bookmarks to remember up to 9 addresses.
Goto remembered	Go to previously remembered address
Display difference	Display difference <current address> - <remembered address>

Menu "Tools"

Text mode	Toggle between text mode (text area is active, address is displayed in decimal format) and hex mode (hex area is active, address is displayed in hexadecimal format)
Overwrite	Turn overwrite mode on (input is overwriting existing characters) or off (input is inserted before current address)
Data Inspector	Show or hide Data Inspector to view permanently decoded numbers
Decode number	Decode from current address 1 byte as shortint, 2 bytes as integer and word, 4 bytes as longint, 4 and 8 bytes as float (provided bytes are existing); result can be copied into clipboard
Encode number	Encode number as shortint, byte, word, integer, longint, or float; result can overwrite existing characters, be inserted before current address, or only be displayed in dialog
Bit manipulation	Allows to view or set the bits of the byte at the current address
CRC	Computing of CRC16 (standard) and CRC32 (PKZIP compatible) checksums for complete file and selected block (if block is currently selected).
Patch BP 7.0 EXE	Due to a bug in unit CRT, BORLAND PASCAL 7.0 executables can't be executed on CPUs over 200 MHz. Simply open the file, select this command and save the file. Note: The bug is fixed by changing hex string `F7 D0 F7 D2 B9 37` to `F7 D0 F7 D2 B9 7E`
XVIscript	Invoke XVIscript interpreter to edit, load/save script, perform syntax check on script, execute script.
Options	Change general options, appearance, Data Inspector settings, and manage shortcut link

Menu "XVIScript"

Editor...	Invoke the XVIscript editor window to edit, load, save, syntax check or execute a script by built-in interpreter
Execute active	Immediate execution of the script currently loaded within the script editor window

More about XVIscript

Menu "Help"

Contents	Invokes online help
About	Display information about the author and memory state (current file size, current allocated memory)

Installation

No special installation procedure is required. Just unzip the file XVI32.ZIP which contains the following files:

XVI32.EXE	Executable
XVI32U.HLP	Help file
XVI32U.CNT	Help contents file
*.XCT	Included files for character conversion
README.TXT	Information about installation

Copy these files into a directory of your choice. No data will be written to registry, no DLLs are copied to your hard disk. The user defined settings are stored in a file XVI32.INI which is located in the same directory as the executable.

After Installation, you should go to the options page and create a shortcut link in Windows' SendTo folder. This enables you to right-click on any file in Windows Explorer and choose *Send It There -> XVI32* to open it with XVI32.

Using Keys

Pos1	Go to first character of line
End	Go to last character of line
CTRL-Pos1	Go to first character of page
CTRL-End	Go to last character of page
PgUp	Go one page up
PgDown	Go one page down
CTRL-PgUp	Go to first page of file
CTRL-PgDown	Go to last page of file
Ins	Switch between insert/overwrite mode
Tab	Switch between text and hex mode (enabling either input as text or hex)

The shortcuts of common menu commands are indicated in the menu.

If this radio button is selected, the cursor automatically goes to the corresponding edit field where you can enter a text string.

If this radio button is selected, the cursor automatically goes to the corresponding edit field where you can enter a hex string (e.g. "0D 0A" for carriage return/line feed characters).

After entering text in the appropriate text edit field (e.g. "abc"), you can convert the text to the corresponding hex string ("61 62 63") using this button.

If this radio button is selected, the number in the corresponding edit field can be entered in decimal system. If the edit fields contains a hexadecimal value, it will automatically be converted.

If this radio button is selected, the number in the corresponding edit field can be entered in hexadecimal system. If the edit fields contains a decimal value, it will automatically be converted.

Check this box if search should be case sensitive according to the currently installed language driver.
Note: Checking this option will increase search speed.

A search string can contain joker (wildcard) chars if the hexadecimal value of the desired joker char is entered in the appropriate edit field (default is 2E for ".") and the corresponding box is checked. A joker (wildcard) character will match any character. Note: Searching with joker chars will decrease search speed.

Search can be carried out downwards or upwards.

Replace all occurrences from current address until end of file without confirmation.

The target address to go to is
4 the indicated address **(absolute)**
4 the current address plus the indicated address **(relative down)**
4 the current address minus the indicated address **(relative up)**

Determines the data type for the integer to be encoded. The byte order (little endian or big-endian) can be selected via `Tools | Options | Data Inspector`

The encoding is done in the indicated byte order. The output can be
4 inserted before the current address
4 overwriting existing characters beginning with the current address
4 displayed in the corresponding edit field (file will not be changed)
The byte order can be selected via `Tools | Options | Data Inspector`.

Here you can enter text (e.g. "abc")

Here you can enter a hex string (e.g. "61 62 63")

Title bar indicating the name of the edited file

Current address in decimal (when in text mode) or hexadecimal format (when in hex mode).
Note: This status panel is blank when no file is opened or file is empty.

Decimal code of character at current address.
Note: This status panel is blank when no file is opened or file is empty.

Indication for mode "Insert" (input is inserted before current address) or "Overwrite" (input is overwriting existing characters); switching is possible using <Ins> key or menu `Tools | Overwrite`

Hint for controls or currently performed operation. Here you can also see an information after a file was opened in Read Only mode.

This is the text area where the file contents is displayed in ASCII (if "Terminal" is chosen as font) or ANSI representation. If the text area is not grayed, it is enabled, and the file is edited in text mode, i.e. text characters are accepted as input.

In text mode, the current address is displayed in the status bar in decimal format.

Click on the disabled area, use the <TAB> key or the `Tools | Text mode` menu to toggle between text and hex mode.

You can change the number of rows and columns used for both areas. Simply resize the window or use the menu `Tools|Options` (where you can change font and font size, too). Note that the minimum number of rows and columns is 10, whereas the maximum is depending on your screen resolution.

This is the hex area where the file contents is displayed in hexadecimal representation. If the hex area is not grayed, it is enabled, and the file is edited in hex mode, i.e. hex characters (0-9, A-F, a-f) are accepted as input.

In hex mode, the current address is displayed in the status bar in hexadecimal format.

Click on the disabled area, use the <TAB> key or the `Tools | Text mode` menu to toggle between text and hex mode.

You can change the number of rows and columns used for both areas. Simply resize the window or use the menu `Tools|Options` (where you can change font and font size, too). Note that the minimum number of rows and columns is 10, whereas the maximum is depending on your screen resolution.

Memory usage

XVI32 is loading the complete file into (virtual) memory. Memory is allocated dynamically, i.e. only as much memory as needed for the actual file size will be used. To achieve better performance,

- a minimum of 512 KB is always allocated
- if the file size is increasing, additional memory is allocated in chunks of 128 KB
- if file size is decreasing, memory is released in chunks of 128 KB, if possible

4 Address of first byte can be displayed as 0 or 1
4 If "Save file with old date/time" is checked, the time stamp of the original file is not modified when using the `Save` or `Save as` commands
4 The item count of `File | Reopen` history can be adjusted between 1 and 9
4 If "Create new file with EOF char" is checked, `File | New` automatically generates the EOF char $1A.

4 Select the desired font using the font combo box
4 Adjust font size
4 Select color of chars in block
4 Adjust number of rows and columns (or simply resize XVI32 window)
4 Check "Hide toolbar" to have more space for text and hex representation available
4 Check "Hide Address of Rows" to turn off displaying of the address of the first byte of each row in hexadecimal representation
4 Check "Text grid on the left" to revert default position of text and hex area
4 Use the "Defaults" button to restore the original settings.

Using help in dialogs

All dialogs should be self-explaining. In addition, most of them provide context sensitive help. Simply use the "What' This?" button [S] to get help on the controls of each dialog.

Miscellaneous

- Go to the options page to create a shortcut link in Windows' SendTo folder. This enables you to right-click on any file in Windows Explorer and choose *Send It There -> XVI32* to open it with XVI32.
- When working with large files, inserting of single characters is relatively slow. Therefore, the function `Edit | Insert string` is recommended for inserting complete strings.
- When you copy data to clipboard using `Edit | Clipboard | Copy` menu item, they have a XVI32 binary format that can only be pasted from XVI32, not by other programs. But you can paste this binary format as well as text format copied within other applications.
- Use `Edit | Clipboard | Copy as hex string` to copy selected block as hex string (e.g. "ABC" as "41 42 43") to clipboard.
- For searching and replacing with confirmation, use F3 for locating the next occurrence, decide whether to replace or not and press F4 if needed. Proceed pressing F3 - F4 until end of file is reached.
- You can start XVI32 from command line with parameter, e.g.: `XVI32 TEST.TXT` starts XVI32 and opens the file Test.txt (enclose file name that contains space(s) within quotes, e.g. `XVI32 "WITH SPACE.TXT"`)
- You can open a file using drag and drop from Windows Explorer
- When scrolling through a file, screen repainting may be slow

Insert string

When working with large files, inserting of chars is relatively slow. Therefore, this function is recommended for inserting consecutive chars (strings).

4 Enter a text (e.g. "abc") or hex string (e.g. "61 62 63")

4 Enter how often the string has to be inserted (as decimal or hexadecimal value)

Inserting Unicode strings

If you check the box ”as Unicode Latin”, the string will be internally converted into Unicode Latin (UTF-16) format as follows:

4 Depending on the options set for the data inspector, little or big endian (UTF-16LE or UTF-16BE) is used for coding (as displayed by the label of the check box).

4 For little-endian encoding, a zero byte is inserted **after** each character

4 For big-endian encoding, a zero byte is inserted **before** each character

Please use the similar command `Overwrite string...` if you want to overwrite existing characters.

Find

This function will find the next occurrence of a string beginning at the current address (or at beginning or end of file) downwards or upwards.

- Enter the string to be found as text (e.g. "abc") or hex string (e.g. "61 62 63")
- For converting text to the corresponding hex string (that you want to change or extend), use the button "Text -> Hex"
- Check the appropriate box if search should be case sensitive according to the currently installed language driver (Note: Checking this option will increase search speed)
- A search string can contain joker (wildcard) chars if the hexadecimal value of the desired joker char is entered in the appropriate edit field (default is 2E for ".") and the corresponding box is checked. A joker (wildcard) character will match any character. (Note: Searching with joker chars will decrease search speed)
- Select if search is to be carried out downwards or upwards
- Search can begin at current address ("Cursor") respectively at beginning ("Begin") or end ("End") of file
- Click on the "OK" button

Subsequently, you can find the next occurrence using `Search | Find again down` (F3) or `Search | Find again up` (SHIFT-F3).

Searching for Unicode strings

If you check the box ”as Unicode Latin”, the string will be internally converted into Unicode Latin (UTF-16) format as follows:

- Depending on the options set for the data inspector, little or big endian (UTF-16LE or UTF-16BE) is used for coding (as displayed by the label of the check box).
- For little-endian encoding, a zero byte is inserted **after** each character
- For big-endian encoding, a zero byte is inserted **before** each character

Count

This function will count the occurrence of a string beginning at the current address (or at beginning or end of file) downwards or upwards.

- Enter the string to be counted as text (e.g. "abc") or hex string (e.g. "61 62 63")
- For converting text to the corresponding hex string (that you want to change or extend), use the button "Text -> Hex"
- Check the appropriate box if search should be case sensitive according to the currently installed language driver (Note: Checking this option will increase search speed)
- A search string can contain joker (wildcard) chars if the hexadecimal value of the desired joker char is entered in the appropriate edit field (default is 2E for ".") and the corresponding box is checked. A joker (wildcard) character will match any character. (Note: Searching with joker chars will decrease search speed)
- Select if counting is to be carried out downwards or upwards
- Counting can begin at current address ("Cursor") respectively at beginning ("Begin") or end ("End") of file
- Click on the "OK" button

Counting Unicode strings

If you check the box ”as Unicode Latin”, the string will be internally converted into Unicode Latin (UTF-16) format as follows:

- Depending on the options set for the data inspector, little or big endian (UTF-16LE or UTF-16BE) is used for coding (as displayed by the label of the check box).
- For little-endian encoding, a zero byte is inserted **after** each character
- For big-endian encoding, a zero byte is inserted **before** each character

Replace

You can replace the next or all occurrences of a string beginning at current address (or at beginning of file) until end of file.

- Enter the string to be replaced as text (e.g. "abc") or hex string (e.g. "61 62 63")
- For converting text to the corresponding hex string (that you want to change or extend), use the button "Text -> Hex"
- Enter the string to replace with in the same manner
- Check the appropriate box if search should be case sensitive according to the currently installed language driver (Note: Checking this option will increase search speed)
- A search string can contain joker (wildcard) chars if the hexadecimal value of the desired joker char is entered in the appropriate edit field (default is 2E for ".") and the corresponding box is checked. A joker (wildcard) character will match any character. (Note: Searching with joker chars will decrease search speed)
- Replacing can begin at current address ("Cursor") respectively at beginning ("Begin") of file
- Click on the "OK" button to perform one replacement
- Click on the "Replace all" button to replace all occurrences until end of file without confirmation

Subsequently, you can replace the next occurrence using `Search | Replace again down` (F4). Note that you can undo only the last replacement by `Search | Replace undo`.

In addition, you can find the next occurrence using `Search | Find again down` (F3) or `Search | Find again up` (SHIFT-F3).

Replacing with confirmation:

- Use F3 to locate the next occurrence
- Decide whether to replace or not and press F4 to perform replacing
- Proceed pressing F3 - F4 until end of file is reached

Searching/Replacing Unicode strings

If you check the box ”as Unicode Latin”, the string will be internally converted into Unicode Latin (UTF-16) format as follows:

- Depending on the options set for the data inspector, little or big endian (UTF-16LE or UTF-16BE) is used for coding (as displayed by the label of the check box).
- For little-endian encoding, a zero byte is inserted **after** each character
- For big-endian encoding, a zero byte is inserted **before** each character

Go to

You can go to an absolute or relative (to the current) address. Enter a decimal or hexadecimal value and select the appropriate go mode. The target address to go to is

- the indicated address **(absolute)**
- the current address plus the indicated address **(relative down)**
- the current address minus the indicated address **(relative up)**

Encode number

You can encode integers as shortint, byte, word, integer or longint; floats in IEEE format (4 or 8 bytes). The result can

- overwrite existing characters
- be inserted before current address
- only be displayed in dialog (file will not be changed)

At first you should select the desired byte order via options. Then proceed as follows:

- Enter the value of the desired number
- Select an integer or float type for encoding
- Select where output will be done

Note to floats

- You must enter the decimal point according to settings in control panel
- Valid are both fixed (e.g. -1024.25 or 0.125) and scientific format (e.g. -1.02425E3 or 1.25E-1)

Mark a block

To mark a block of characters, use one of the following three methods:

1. Go to the first character of the desired block and select `Edit | Block mark` or press CTRL-B. Then go to the last character of the desired block and select `Edit | Block mark` or press CTRL-B.
2. Go to the first or last character of the desired block, press and hold down SHIFT and use the cursor keys to extend the block.
3. Go to the first character of the desired block and select `Edit | Block <n> chars` to enter the number of chars the block should consist of.

A block can be deleted, moved or copied to clipboard.

Search begins at cursor position (if "Cursor" is checked) respectively at beginning ("Begin") or end ("End") of file.

Character Conversion

This command is used to convert characters from one character set into another, e.g. from DOS to Windows, by using a user-defined character conversion table. Converting can start at the first character or at the current address.

When selected, a window with a grid for the conversion table appears. This table displays in the first column all characters from $00 to $FF, whereas the second and the third column contain for the printable characters the representation of the character in ANSI respectively ASCII code.

Initially, the right column of the grid is empty. This means that no character will be converted, i.e. no replacements will be accomplished.

Supposed you want to convert the character $84 to $E4 (a German "Umlaut" from DOS/ASCII to WIN/ANSI). Use one of the following methods:

4 Enter E4 in the right column of the line for character 84 ("char 84, convert to E4")

OR

4 Use drag and drop: drag the field containing 84 ("convert 84...") and drop it to the last column of the line for character E4 ("...to E4"). During dragging, move the mouse cursor to the upper or lower border of the grid if scrolling is needed. Note that after drag and drop, the line where dragging **started** is filled automatically with the character code of the line where dropping took place.

Repeat this for all characters you want to convert. When finished, save your conversion table to file using the "Save Table" button. By default, the file will be stored in the directory of the application with the extension ".xct" for **X**VI32 **c**onversion **t**able. Likewise, you can load a table from file.

Use the button "Convert" to perform the conversion starting at the first character or at the current address (depending on the setting of radio group "Scope from").

Note: XCT files are plain text, so you can create them also using a text editor. Just write **for each character to convert** a line with the hexadecimal character codes separated by semicolon, e.g. **80;C7** to replace character $80 by $C7.

Several XCT files are included.

For any character to convert, enter the new hexadecimal char code in this table. You can also use drag and drop, e.g. drag the first column containing 84 ("convert 84...") and drop it to the last column of the line for character E4 ("...to E4"). An empty column "Convert to" means that no conversion for the appropriate character will be performed.

The term comes from 'Gulliver's Travels' by Jonathan Swift. The Lilliputians, being very small, had correspondingly small political problems. The Big-Endian and Little-Endian parties debated over whether soft-boiled eggs should be opened at the big end or the little end.

Included XCT files

file name	used to convert from character set	to character set
DOSWIN.XCT	DOS codepage 850	Windows codepage 1252
WINDOS.XCT	Windows codepage 1252	DOS codepage 850
WINEBCUS.XCT	Windows codepage 1252	EBCDIC US
EBCUSWIN.XCT	EBCDIC US	Windows codepage 1252
WINEBCDE.XCT	Windows codepage 1252	EBCDIC DE (Germany/Austria)
EBCDEWIN.XCT	EBCDIC DE (Germany/Austria)	Windows codepage 1252

Auto-fill feature

This feature facilitates searching or replacing strings.

When the cursor is positioned in an empty input field for a hex string and you press the right arrow key, the field will be filled automatically with hex values taken one after another from the current position within the file:

- Clear the input field
- Hold down the right arrow key or press it several times
- Auto-fill begins with the hex code of the character at the current file position
- The hex codes of the following characters are filled in, respectively
- This is feature is available in the *Search*, *Count*, *Replace* and *Insert* dialogs

Printing

You can select a printer, the font size and various options. Before printing, you will always preview the output with the option to print single pages or all pages. During printing it is possible to abort. Optionally, you can direct the output to a file.

Printing range

4 only current page
4 current selection (marked block)
4 all pages

Margins

4 in millimeters or inches
4 adjust left, right, top, and bottom margin
4 values below the physical printable area margins will be automatically corrected
4 you can't set the paper size, because the margins are applied to the printer's default paper size

Output format

4 portrait or landscape
4 select or enter number of bytes per row
4 offset in hexadecimal or decimal format (always zero-based, independent from the options settings)
4 print text below (in 2nd line) or on the right side

Output with text on the right side

```
Offset hex. 00 01 02 03 04 05 06 07
000000000:  41 42 43 44 45 46 47 48 ABCDEFGH
000000008:  49 4A 4B 4C 4D 4E 4F 50 IJKLMNOP
```

Note: In this case, any non-printable char is printed in text as dot.

Output with text below (in 2nd line)

```
Offset hex. 00 01 02 03 04 05 06 07
000000000:  41 42 43 44 45 46 47 48
             A  B  C  D  E  F  G  H
000000008:  49 4A 4B 4C 4D 4E 4F 50
             I  J  K  L  M  N  O  P
```

Note: In this case, any non-printable char is printed in text as blank.

Print Preview

This windows shows all pages to print. Use the toolbar to

- scroll through the pages back and forth
- go to the first or last page
- quick zoom between full page height or full page width
- print only currently previewed page
- print all pages

Left-click with the mouse within the previewed page to magnify, right-click to reduce the visible area. The PageUp and PageDown keys can be used to go to the previous and next page, respectively.

these margins are applied to the printer's default paper size

if checked, the output looks like follows:

```
Offset hex. 00 01 02 03 04 05 06 07
000000000:  41 42 43 44 45 46 47 48
             A  B  C  D  E  F  G  H
000000008:  49 4A 4B 4C 4D 4E 4F 50
             I  J  K  L  M  N  O  P
```

otherwise, the output looks like follows:

```
Offset hex. 00 01 02 03 04 05 06 07
000000000:  41 42 43 44 45 46 47 48 ABCDEFGH
000000008:  49 4A 4B 4C 4D 4E 4F 50 IJKLMNOP
```

please use this option, **not** a installed text file printer driver which may produce unsuitable output depending on the installed text driver

for redirecting output to file, don't use a installed text file printer driver, but check option "redirect output to file"

4 Easily create or remove a shortcut link in Windows' SendTo folder.
4 Under Windows 9x, the SendTo folder will usually be C:\Windows\SendTo.
4 Under Windows NT, this is a user specific folder like C:\Windows\Profiles\<USER>\SendTo, i.e. every user must create his own shortcut link.
4 Under Windows XP, the folder is located user-specific at C:\Documents and Settings\<USER>\SendTo

This is the menu bar.

Using Clipboard

XVI32 handles two clipboard formats: text and and a particular binary format.

- The clipboard functions cut and copy are available after a block has been marked. They transfer data in a XVI32 binary format to clipboard, i.e. these data can only be pasted from XVI32, not by other applications.
- Use `Edit | Clipboard | Copy as hex string` to copy selected block as hex string (e.g. "ABC" as "41 42 43") to clipboard. This is a text format and can be pasted within other applications.
- `Edit | Clipboard | Paste from hex string` is providing the inverse operation, i.e. paste a hex string from clipboard into the file (e.g. "41 42 43" or "414243"). This can be used e.g. to apply binary patches. An error message appears, if the clipboard does not contain a valid hex string.
- You can paste text copied to clipboard with other applications.
- The block functions delete, copy and move don't use the clipboard and don't allocate additional memory. They are therefore recommended for operations within a file.

Bit manipulation

This dialog allows to view or set the status of each bit at the current address. The bits 0..7 are represented by the correspondent check boxes. In addition, the decimal, hexadecimal and binary representation of the current bit status is indicated.

Bit	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1

Buttons are provided for the following commands:

- Set all bits to one ("Check all")
- Set all bits to zero ("Uncheck all")
- Toggle all bits
- Set bit status for current address
- Reread bit status for current address

View or set the status of each bit at the current address. The bits 0..7 are represented by the correspondent check boxes. Use the buttons to

- Set all bits to one ("Check all")
- Set all bits to zero ("Uncheck all")
- Toggle all bits
- Set bit status for current address
- Reread bit status for current address

Here, the address of the first byte of the corresponding row is indicated in hexadecimal representation.
You can hide this area by checking "Hide Address of Rows" in the `Tools | Options` menu.
Note: The address of the first byte of the file can be set to 0 or 1 using the `Tools | Options` menu.

What's a hex editor?

Some time ago, I received the following e-mail:

Hello,

first of all thank you for your free hex editor.

I have never used one nor do I know exactly what it is for, but 2 people at alt.comp.freeware told me to get one so I could "see" the watermark/invisible imprint that a jpg compressor puts in the jpg's I compress! Now that I have your program (thank you again)....

What is a hex editor? Why do I need one (beside the above problem)? How do I change the imprint I found....thanks to your program?

Here's my attempt to answer these questions.

A hex editor is mainly a tool to examine the structure of files. For every programmer (like me) this is an almost daily duty, but for "normal" users this may sound strange. But consider so called "plain text" files like the ubiquitous README.TXT files, created e.g. using notepad (without any layout like bold formatted words etc). They **don't contain only the plain text**, but also end-of-line marks (and often tabstops). Unfortunately, these end-of-line marks (not visible itself in notepad, you'll see only the breaking line) are coded differently under Windows, Unix, and Macintosh. If you happen to get a Unix or Mac text file over the net, it may cause problems under Windows (e.g. it will not show properly in notepad). Using XVI32 (or another hex editor), you can look into the file, examine the end-of-line marks and replace if necessary (using `Search | Replace | Replace all`). Just open the README.TXT file that comes with XVI32. The first 30 characters are looking in the hex area as follows:

```
48 65 78 2D 65 64 69 74 6F 72 20 58 56 49 33 32
20 76 65 72 73 69 6F 6E 20 31 2E 35 0D 0A
```

These are the hexadecimal codes for each byte. The hexadecimal system is similar to our common decimal system (which has the digits 0, 1, ... 9), but with the additional digits A, B, C, D, E, F - and the meaning of those digits 10, 11, 12, 13, 14, 15. To write the number 10, you need **two** digits in the decimal system, but in the hexadecimal system **one** digit A is sufficient.

In the decimal and hexadecimal system, the weight of a digit depends from its position within a number. With each position, the weight of a digit is incremented by factor 10 (or 16). The decimal number 72 e.g. consists of the digit 7 (with weight 7 * 10) and digit 2 (with weight 2 * 1), i.e. the value of 72 is 7 * 10 + 2 * 1. This may seem trivial to you, but consider now the hexadecimal system: the hex number 48 stands for 4 * 16 + 8 * 1 = 72.

When you've openend README.TXT in XVI32, the decimal value of hex 48 is displayed in the status line behind "Char dec:". The hex code 48 represents the capital letter H. There are 256 possible values for each byte ranging from decimal 0 to 255 (or hex 0 to FF). These 256 "characters" comprise the 26 capital letters, 26 small letters, 10 digits, special characters e.g. like the German Umlaute, but also non-printable control codes.

The last two bytes above (0D 0A = decimal 13 10) represent a line break under Windows. If this README.TXT were created under Unix, you would see only 0A. Try to download several HTML files (which are plain text in opposite to binary files like JPG, GIF etc. which have non human-readable contents) from the web and look for the end-of-line mark. Mostly you'll find 0D 0A, but sometimes 0A, depending from their origin (and the transfer mode to your PC). To convert Unix text files to Windows format, you need to change all occurrences of 0A by 0D 0A.

Use XVI32 to open a document created with a word processor: Besides the text itself, you'll see additional information. Most of them "unreadable" for you, but maybe there is also previously deleted text, your name, the complete path to your document and so on. BTW: This is a severe security problem! **But be careful:** Changing the contents of binary files may make them completely useless. Don't blame me if improper use of XVI32 will damage a valuable file...

So you need hex editors for mainly two reasons:

4 Examining the structure of a file, because you don't see everything which is stored in a file using the application to open it (this is true also for so called "plain text" files as explained above). You need this knowledge e.g. to write by yourself an application that will interpret the contents of a file properly.

4 Changing the contents of a file deliberately. This also requires to know the exact structure of the file. If you don't know how watermarks are stored in JPGs, don't tamper with them - aside from the fact that it would be illegal to delete Copyright information.

I hope this helps and will be a starting point for further investigations done by yourself. But always remember: be careful to change file contents!

Overwrite string

This function is similar to `Insert string...` and allows to overwrite existing characters with a text or hex string. You can use this e.g. to apply binary patches: copy the hex string to the clipboard and paste it to the appropriate edit field.

4 Enter a text (e.g. "abc") or hex string (e.g. "61 62 63")

Menu "Bookmarks"

You can define up to 9 named bookmarks for important addresses of a file. The menu Bookmarks is providing the following commands:

4 Use **Add** to add a bookmark for the current address. Giving a name in the following dialog is optional.

4 **Remove** allows to remove a single bookmark from the current list.

4 **Remove all** clears the complete list of currently defined bookmarks.

4 **Save to file** will save the currently defined bookmarks to a XBM (XVI Bookmarks) file.

4 To reload bookmarks later, e.g. when you examine the same file or a file with similar structure, use **Load from file**.

Just click on the desired bookmark to return to the appropriate address. These bookmarks appear at the end of the bookmark menu with the following information:

4 Accelerator char.

4 Address in decimal (when editing the file in text mode) or hexadecimal (when in hex mode) format. Depending on the current options setting for offset, the displayed address is zero-based or one-based.

4 Name of the bookmark (if entered).

Example

1 $0: BOF

2 $36: EOF

XVIscript: Introduction

The built-in XVIscript interpreter is a powerful feature to automate tasks (e.g. apply binary patches). You can

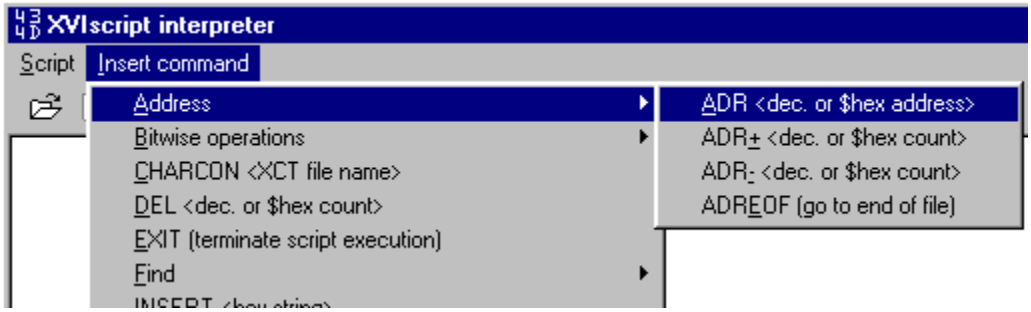
- create or edit scripts
- load a script from a XSC (XVI Script) file; default directory is the XVI32 directory
- save a script to file
- perform a syntax check
- execute a script

Suppose you have often to convert Unix text files into Windows format by replacing all EOF marks $0A by $0D 0A. Although this is very easy to accomplish using the `Search | Replace` dialog, you could use the following almost self-explaining script:

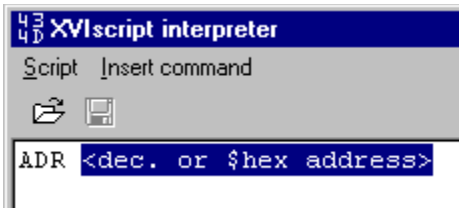
```
REM goto begin of file (always zero-based)
ADR 0
REM replace all $0A by $0D 0A
REPLACEALL 0A BY 0D 0A
```

The interpreter is expecting hex strings to have spaces between each byte as shown in the above `REPLACEALL` command. These spaces must be entered manually, i.e. they are not filled automatically in contrast to other places in the program (e.g. search, replace).

All available script commands are explained in detail later. For now, just open a text file and select `XVIscript | Editor...` from the main menu. The script editor window appears where you can enter (or paste) the above script. But for the beginning, it's easier to select from the `Insert command` menu from a list of all available commands as shown below (similar commands are grouped into submenus).



When you select a command, it will be pasted at the current position within the editor. If a command needs additional argument(s), the pasted text does contain an appropriate explanation within `<...>`. This explanation will be highlighted by default, so the next key press will overwrite it. Text within `(...)` is only an additional hint and will not be pasted into the editor; refer to `ADREOF (go to end of file)` in the above picture. Text within `[...]` is referring to optional parameters.



XVIscript: Syntax check and execution

Syntax check

If you press this button, only a syntax check of the script is performed. **A Syntax check will never do apply changes to the file.** The first syntax error will be reported, and the faulty line of the script will be marked. Note that a syntax check will not detect typically run-time errors (e.g. an ADR command with an invalid address or a VERIFY command that fails during execution).

Arguments of script commands containing run-time parameter(s) will be excluded from syntax check, e.g.

```
ADR %1
REPLACEALL %2 BY %3
```

The command itself must not be a parameter, so the following line would be illegal:

```
%4
```

Execution

This will automatically perform a syntax check beforehand.

4 If a syntax error does occur, an error message will appear. In this case, **no changes** are made to the file.

4 Only if the syntax check is passed successfully, the script will now be executed line by line, as commented in the previous topic example by the REM (remark) lines.

4 Errors during execution are reported, and the corresponding line will be highlighted automatically. In this case, **all previously made changes remain in effect**.

4 A message informs you that the script was executed without errors. You must close the script editor window to return to the XVI32 main window. You're asked if the script should be saved to file if it was modified or not yet saved at all.

When you execute a script by pressing the appropriate button, XVI32 doesn't save the changes made automatically, so you must close the script editor window and save the edited file manually. But you have the possibility to invoke XVI32 from the command line so that the complete script will run automatically and save the file after successful execution.

You can also save or reload scripts from file using the appropriate menu or toolbar button. XVIscripts are saved as text files with the extension XSC (XVI script file). I'm sure you'll imagine already much more sophisticated things you can do with XVIscript...

Direct script execution

When a script is loaded by the script editor window, you can execute this script directly from the XVI32 main window by menu `XVIscript | Execute active` or by pressing <Ctrl> <F9>. In this case, the editor window will only be displayed on syntax or run-time errors.

XVIscript: Error messages

The syntax check does recognize syntax errors within a script. To see what happens, just enter an invalid command, e.g.

```
CONVERTUNIXTOWINDOWS
ADR BOF
REPLACEALL CR BY LF
```

These invalid commands will result in

- an error message
- highlighting of the faulty line

Note that passing the syntax check does not always guarantee error-free execution of a script. During execution, **run-time errors** may occur (e.g. jump to an invalid address because the file is not large enough, e.g. ADR 100 when the file has only 10 bytes). Because script execution will normally modify the file contents (but syntax check does NO modifications), it would not make sense to look for invalid addresses during syntax check.

Run-time errors are also reported. But in this case, all previously made changes will remain in effect, so you should be extremely cautious before saving the file!

XVIScript: Command reference

The XVIscript editor window allows to select easily from the following available commands. Note that

- each line can only contain one command
- commands are not case sensitive, although in the documentation they are capitalized
- trailing and leading spaces are ignored
- empty lines are ignored
- any line beginning with a semicolon is treated as comment (i.e. ignored)
- the interpreter is expecting hex strings to have spaces between each byte which must be entered manually, i.e. they are not filled automatically in contrast to other places in the program (e.g. search, replace)
- parameters enclosed within <> are necessary, parameters within [] are optional.

ADR <addr>

Go to address <addr>; <addr> is always zero-based and can be indicated either as decimal or hexadecimal (with prefix $) value. ADR should always be the first command in a script; if omitted, ADR 0 will be used as default. If the indicated address is invalid, script execution will be aborted with an error message. Examples:

```
ADR 32
ADR $20
```

ADR+ <n>

Increment current address by <n> bytes; <n> can be indicated either as decimal or hexadecimal (with prefix $) value. If the computed new address is invalid, the script will be aborted with an error message. Examples:

```
ADR+ 16
ADR+ $10
```

ADR- <n>

Decrement current address by <n> bytes; <n> can be indicated either as decimal or hexadecimal (with prefix $) value. If the computed new address is invalid, the script will be aborted with an error message. Examples:

```
ADR- 20
ADR- $14
```

ADREOF

Go to end of file, i.e. last byte.

BITAND <hex byte> [byte count]

For [byte count] bytes beginning at the current address, perform a bitwise AND operation with <hex byte>. Default for [byte count] is 1. Examples:

This will set the byte at the current address to zero:

```
BITAND 00
```

Set 20 bytes beginning at the current address to zero:

```
BITAND 00 20
```

Set $A = 10 bytes beginning at the current address to zero:

```
BITAND 00 $A
```

BITNOT [byte count]

For [byte count] bytes beginning at the current address, perform a bitwise NOT operation (i.e. each bit is

inverted). Default for [byte count] is 1. Examples:

```
BITNOT
BITNOT 20
BITNOT $A
```

BITOR <hex byte> [byte count]

For [byte count] bytes beginning at the current address, perform a bitwise OR operation with <hex byte>. Default for [byte count] is 1. Examples:

This will set bit 1 of the byte at the current address to 1:

```
BITOR 02
```

Set bit 1 of 20 bytes beginning at the current address to 1:

```
BITOR 02 20
```

Set bit 1 of $A = 10 bytes beginning at the current address to 1:

```
BITOR 02 $A
```

BITSHL <bit count> [byte count]

For [byte count] bytes beginning at the current address, perform a logical left shift by <bit count> bits. Default for [byte count] is 1. Examples:

This will left shift the byte at the current address by 1 bit (= multiply byte by 2):

```
BITSHL 1
```

This will left shift 20 bytes beginning at the current address by 1 bit:

```
BITSHL 1 20
```

This will left shift $A = 10 bytes beginning at the current address by 1 bit:

```
BITSHL 1 $A
```

BITSHR <bit count> [byte count]

For [byte count] bytes beginning at the current address, perform a logical right shift by <bit count> bits. Default for [byte count] is 1. Examples:

This will right shift the byte at the current address by 1 bit (= divide byte by 2):

```
BITSHR 1
```

This will right shift 20 bytes beginning at the current address by 1 bit:

```
BITSHR 1 20
```

This will right shift $A = 10 bytes beginning at the current address by 1 bit:

```
BITSHR 1 $A
```

BITXOR <hex byte> [byte count]

For [byte count] bytes beginning at the current address, perform a bitwise XOR operation with <hex byte>. Default for [byte count] is 1. Examples:

```
BITXOR 02
BITXOR 02 20
BITXOR 02 $A
```

CHARCON <XCT file name>

Load indicated character conversion file and perform a character conversion beginning at current address. The XCT file must exist at run-time, otherwise the script will be aborted. Performing a character conversion will not change the current address. Example:

```
ADR 0
CHARCON C:\ProgramFiles\XVI32\DOSWIN.xct
```

DEL <n>

Delete <n> bytes; <n> can be indicated either as decimal or hexadecimal (with prefix $) value. The current address will not be changed as long at it remains valid; otherwise, the current address will be set to end of file. If n is less or equal 0, a syntax error occurs. If there are not enough bytes to delete, this will cause a run-time error. Examples:

```
DEL 64
DEL $40
```

EXIT

Terminate execution of script (without error). Useful e.g. to execute a script only partially. The EXIT command does not terminate syntax checking, i.e. the syntax check is always performed for the complete script.

FIND <hex string>

Find next occurrence of <hex string>. If a joker (wildcard) byte was previously defined using the JOKERON command, the appropriate byte will match any byte within the file. If no match of <hex string> is found, this will cause a **run-time error**. Example:

```
FIND 0D 0A
```

FINDASC <string>

Find (case sensitive) next occurrence of <string>. If a joker (wildcard) byte was previously defined using the JOKERON command, the appropriate byte will match any byte within the file. If no match of <string> is found, this will cause a **run-time error**. Note: any usage of a % character within <string> will be considered as reference of a command line parameter. If a % character should be taken literally, you must use the FIND command instead. Leading and trailing blanks within <string> are allowed. Example:

```
FIND Hello
```

INSERT <hex string>

Insert <hex string> before current address. Afterwards, the current address will be right **after** the last inserted byte. Example:

```
INSERT FF FF
```

JOKERON <hex char>

Define a joker (wildcard) byte that is used by all following FIND, REPLACE, and REPLACEALL commands. A joker (wildcard) byte will match any byte. If <hex char> is invalid (or more than one byte), this will cause a syntax error. Example:

```
JOKERON FF
```

JOKEROFF

Disable previously defined joker (wildcard) byte, i.e. all following FIND, REPLACE, and REPLACEALL commands will not use a joker byte.

MSG <message>

Display <message> on screen. **This command is ignored when executing a script from the command line using the /S parameter.** The message is shown within a memo, so you can copy it to clipboard.

The following placeholders are expanded (case sensitive; numbers are decoded in decimal format):

\n	line break
$BYTE	decode current address as byte
$WORD	decode from current address 2 bytes as word (provided 2 bytes are existing)
$INT	ditto 2 bytes as integer

$LONG ditto 4 bytes as long integer
$IEEE32 ditto 4 bytes as float
$IEEE64 ditto 8 bytes as float

Examples:

```
MSG File successfully patched!\nPlease save.
ADR 1023
MSG Decoded at address 1023\nWord: $WORD\nLong: $LONG
```

OVERWRITE <hex string>

Overwrite bytes beginning at current address with <hex string>. Afterwards, the current address will be right **after** the last byte that was overwritten. If there are not enough bytes to overwrite, this will cause a run-time error. Example:

```
OVERWRITE 00 00 00
```

REM

Line contains remark ignored by XVI32script. Lines beginning with a semicolon are also treated as comments. Examples:

```
REM This is a comment
;this line is ignored too
```

REPLACE <from hex string> BY <to hex string>

Find (case sensitive) next occurrence of <from hex string> and replace by <to hex string>. If a joker (wildcard) byte was previously defined using the JOKERON command, the appropriate byte will match any byte within the file. Afterwards, the current address will be at the **last** replaced byte. If no match of <from hex string> is found, this will cause **no run-time error**. Empty <to hex string> is allowed. Example:

```
REPLACE 0D 0A BY 0A
```

REPLACEALL <from hex string> BY <to hex string>

Same as above, but all occurrences are replaced.

REPLACEASC <from string> BY <to string>

Find next occurrence of ASCII-string <from string> and replace by <to string>. If a joker (wildcard) character was previously defined using the JOKERON command, the appropriate byte will match any character within the file. Afterwards, the current address will be at the **last** replaced byte. If no match of <from string> is found, this will cause **no run-time error**. Empty <to string> is allowed. Note: any usage of a % character within <from string> or <to string> will be considered as reference of a command line parameter. If a % character should be taken literally, you must use the REPLACE command instead. Leading and trailing blanks within <from string> and <to string> are allowed, as shown in the following example where <from string> has both leading and trailing blank:

```
REPLACE  xvi32  BY xvi32
REM     ^-----^ leading and trailing blank
```

REPLACEALLASC <from string> BY <to string>

Same as above, but all occurrences are replaced.

VERIFY <hex string>

Verify that <hex string> is found at current address, otherwise execution of script will be aborted with an error message. Use ?? within the hex string as joker (wildcard) byte that matches any byte. It is also possible to indicate several alternative hex strings separated by OR. Examples:

```
VERIFY 0D 0A
```

```
VERIFY FF ?? FF
VERIFY FF 00 FF OR FF 01 FF
VERIFY FF 00 FF OR 00 ?? FF
```

XVIscript: Command line parameters

First parameter

When you start XVI32 from the command line with a file name as parameter, XVI32 will open this file automatically. Example:

```
XVI32.exe readme.txt
```

Second parameter

There is a second optional parameter: /S (or typed as /S=, -S, -S=). In this case, you must specify a script file name right after /S, e.g.

```
XVI32.exe readme.txt /S=win2unix.xsc
```

This will start XVI32, open readme.txt and then execute the script win2unix.xsc automatically.

4 If no errors do occur and changes were made, the file will be automatically saved, and XVI32 will terminate afterwards.

4 In case of any syntax or run-time error, XVI32 will report this error. It's up to you to handle this situation, i.e. the file **will not be saved automatically and XVI32 will not terminate**.

Additional parameters

The 3rd, 4th, ..., 11th command line parameters of XVI32 will be passed to the script. Please reference these parameters within the script as %1, %2, ..., %9. Example (note the quotation marks neccessary to include spaces):

```
XVI32.exe readme.txt /S=c:\ProgramFiles\xvi32\universalrepl.xsc 0A "0D 0A"
```

In this example, `0A` can be referenced as %1 and `"0D 0A"` as %2 within universalrepl.xsc as follows:

```
REM goto begin of file (always zero-based)
ADR 0
REM in above example %1 is 0A
REM and %2 is 0D 0A
REPLACEALL %1 BY %2
```

Arguments of script commands containing run-time parameter(s) will be excluded from syntax check, e.g.

```
REPLACEALL %1 BY %2
```

The command itself must not be a parameter, so the following line would be illegal:

```
%4
```

Note: Scripts containing run-time parameter(s) can't be executed in interactive mode, i.e. you must execute them from the command line.

Usage of batch file to process multiple files

Using the batch commands FOR and START allows to process multiple files by the same script. Example for a batch file that will convert all text files in the current directory to Unix:

```
FOR %%f IN (*.txt) DO START /W c:\ProgramFiles\xvi32\xvi32.exe %%f /S=c:\
ProgramFiles\xvi32\win2unix.xsc
```

The parameter /W of the START batch commands is ensuring that XVI32 will be relaunched not until the previous script execution has terminated. These examples are also very simple, but you probably can imagine much more sophisticated ones.

XVIscript: Example

On http://www.jps.net/thamiter/anitest/icon2cur.htm you'll find an explanation how to convert a Windows icon into a cursor file. To avoid mistakes while editing the file, the author of the above site could now publish simply the following XVIscript:

```
REM script to convert an icon to cursor
REM http://www.jps.net/thamiter/anitest/icon2cur.htm
ADR 0
REM it must be a single-icon with 16 colors
REM 32x32 pixels not required
REM 16 colors ------------------+
REM pixels    ------------+--+  |
REM 1 icon    ------+     |  |  |
REM                 |     |  |  |
VERIFY 00 00 01 00 01 00 ?? ?? 10 00
ADR 2
REM replace 01 by 02
OVERWRITE 02
REM this is decimal ($A is hex equivalent)
ADR 10
MSG Please enter coordinates in XVI32 as XX 00 XX and save file as .CUR
```

Unfortunately, you must manually enter meaningful coordinates depending on the shape of the icon. In the example on the above website, you should overwrite the three bytes at the current address with `10 00 1D` in XVI32.

Uninstall XVI32

To uninstall XVI32 completely, proceed as follows:

4 If you have created a shortcut link in the SendTo folder, start XVI32, go to the options dialog (Tools | Options... | Shortcut link) and click on the remove button. Under Windows NT, 2000, and XP, every user must log in and perform this task. It is also possible to remove the link(s) manually. The Send To folder is located at C:\Windows\SendTo (Windows 9x), C:\Windows\Profiles\<USER>\SendTo (Windows NT), or C:\Documents and Settings\<USER>\SendTo (Windows XP).

4 Exit XVI32 and delete the complete folder where XVI32 was installed.

Options: General

- Offset (address) of first byte can be set to 0 or 1.
- If "Save file with old date/time" is checked, the time stamp of the original file is not modified when using the `Save` or `Save as` commands.
- The item count of `File | Reopen` history can be adjusted between 1 and 9.
- If "Create new file with EOF char" is checked, `File | New` automatically generates the EOF char $1A.

Options: Appearance

4 Select the desired font using the font combo box.

4 Adjust font size.

4 Select color of chars in block.

4 Adjust number of rows and columns (or simply resize XVI32 window).

4 Check "Hide toolbar" to have more space for text and hex representation available.

4 Check "Hide Address of Rows" to turn off displaying of the address of the first byte of each row in hexadecimal representation.

4 Check "Text grid on the left" to revert default position of text and hex area.

4 Uncheck "Show Grid Lines" to have a more conventional look

4 Uncheck "Use Blank to Display Control Characters" if you prefer to see characters below blank within the text area

4 Use the "Defaults" button to restore the original settings.

Options: Data Inspector

4 Select byte order for coding integers and floats: little-endian (Intel: least significant byte leftmost, most significant byte rightmost) or big-endian (Motorola: vice versa). Note that this byte order is also important when searching, counting, replacing, or inserting Unicode strings.

4 Select data types shown in Data Inspector. If "block size" is checked, you will also see the number of currently selected bytes in a marked block.

To view the Data Inspector, check the `Options | Data Inspector` menu item.

Options: Shortcut Link

To open any file with XVI32 from the Windows Explorer, you can create a shortcut link in the SendTo folder. Thus, you can select a file and choose `Send To` 4 `XVI32`.

4 Use the appropriate button to create or remove the shortcut link.

4 Under Windows 9x, the SendTo folder will usually be C:\Windows\SendTo.

4 Under Windows NT, this is a user specific folder like C:\Windows\Profiles\<USER>\SendTo, i.e. every user must create his own shortcut link.

4 Under Windows XP, the folder is located user-specific at C:\Documents and Settings\<USER>\SendTo

Data Inspector

The Data Inspector is a tool window (always staying on top) to view permanently decoding of numbers. The `Tools | Options` dialog (also selectable by the Data Inspector's context menu item `Configure...`) allows to select which of the following data types are shown:

4 short (one byte signed)

4 byte

4 word (two bytes unsigned)

4 integer (two bytes signed)

4 longint (four bytes signed)

4 IEEE single (four bytes)

4 IEEE double (eight bytes)

4 binary

If the additional item "block size" is checked, you will also see the number of currently selected bytes in a marked block.

To edit a number, simply double-click on the decoded value shown in the Data Inspector (or select from the context menu the item `Edit...`).

4 Select byte order for coding integers and floats
4 Select data types shown in Data Inspector
4 If "block size" is checked, you will also see the number of currently selected bytes in a marked block
4 To view the Data Inspector, check the `Options | Data Inspector` menu item

4 If checked, the string is internally converted into Unicode Latin (UTF-16) format
4 Depending on the options set for the data inspector, little or big endian (UTF-16LE or UTF-16BE) is used for coding
4 For little-endian encoding, a zero byte is inserted **after** each character
4 For big-endian encoding, a zero byte is inserted **before** each character