
VT_EX/Free 7.30 Manual

2001-05-07
M. Vulis
W. Schmidt

This is the documentation for the free releases of VT_EX. Currently, there are freeware versions for Linux (VT_EX/Linux) and OS/2 (VT_EX/2). If this document refers to both of them, it will use the generic term VT_EX/Free. If the document refers to something which is also true for the commercial Windows version, it will just use the bare VT_EX.

Contents

1	What is VTeX/Free?	3
2	Brief VTeX Documentation	4
2.1	Invocation and usage	4
2.2	The main configuration file	5
2.3	Configuring the font usage	6
2.3.1	The FM file	6
2.3.2	The font map files	7
2.3.3	Base font substitution	8
2.3.4	Re-encoding	9
2.3.5	Encoding files	9
2.4	Path caching	10
3	PostScript and PDF generation	12
3.1	General	12
3.2	Command-line options	12
3.3	Font usage in PDF and PostScript documents	13
3.4	CFF conversion	14
3.5	PCL printing with Acrobat 4	15
3.6	Media size	16
3.7	Links	16
3.8	Outline	18
3.9	Transitions	18
3.10	Thumbs	19
3.11	Annotations	20
3.12	Non-Latin characters in Outlines and Annotations	20
3.13	Color stack issues	21
3.14	Direct PDF/PS writing	21
3.15	PDF information special	22
3.16	Bitmapped graphics inclusion	22
3.16.1	Image size	22
3.16.2	Compression and fine-tuning	23
3.16.3	Imagemasks	25
3.17	PostScript file inclusion	25
3.17.1	Page selection	25
3.18	Page Labels	25
3.19	Unsupported VTeX syntax	26
4	The LaTeX packages color and graphicx	27
4.1	Color support	27
4.2	Text-box scaling and Rotation	27
5	Language extensions	29
6	Support	30
7	License Terms	31

1. What is $\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$?

$\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$ is a *partial* port of the $\text{V}\text{T}_{\text{E}}\text{X}/\text{Windows}$ $\text{T}_{\text{E}}\text{X}$ compiler to Linux and OS/2. It does not include any shell and/or Visual Tools and there is currently no intention to port those.

Even the port of the compiler itself is partial. Out of the four modes of $\text{V}\text{T}_{\text{E}}\text{X}/\text{Win}$, only three are supported (**PDF**, **PS** (PostScript) and **DVI**, but not **HTML**); the DVI mode is essentially useless, since the main advantages of $\text{V}\text{T}_{\text{E}}\text{X}$'s DVI mode under Windows rely on the $\text{V}\text{T}_{\text{E}}\text{X}$ DVI drivers which have not been ported.

Thus, for all practical purposes, $\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$ should be viewed as a PDF and PS mode compiler only.

The PDF and PS modes are very similar in the operation; the implementation code is essentially the same and both the lower-level (`\special's`) and the high-level (configuration files for `graphicx`, `pstricks` etc.) are almost identical.

This documentation, therefore, equally applies to both modes; differences, where exist, are specifically marked.

$\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$ *does* include the full PostScript support ($\text{G}_{\text{E}}\text{X}$) of the Windows version. This includes both inclusion of (encapsulated) PostScript files and inline PostScript, including support for `PStricks`, `PSfrag`, `XYpic` and $\text{G}_{\text{E}}\text{XX}$. $\text{G}_{\text{E}}\text{X}$ can be used in either the PDF or the PS mode; in PDF mode $\text{G}_{\text{E}}\text{X}$ acts as an integrated PDF \rightarrow PS distiller; in the PS mode, it acts as a PS \rightarrow PS distiller.

The Linux port requires Linux version 2.0, and at least 16 MB of physical memory (without X11) or 32 MB (with X11). The OS/2 port needs OS/2 Warp 3 or later, and at least 16 MB of physical memory. Below these limits performance may be unacceptably slow.

2. Brief VT_EX Documentation

2.1 Invocation and usage

The name of the VT_EX executable is `vtex.exe` with OS/2 and `vtexlnx` with Linux. Thus, with OS/2 the syntax of the command line is:

```
vtex -${p|s}[(MODE options) [Other options] [@format] mydoc[.tex]
```

With Linux, the full path of the executable is to be given; even though it should reside in a directory which is listed in the `$PATH`:

```
/usr/local/vtex/bin/vtexlnx -@{p|s}[(MODE options) \
[Other options] [@format] mydoc[.tex]
```

In case the name of the format is not be specified, it defaults to `vtex -` as opposed to `plain` with many other T_EX implementations.

VT_EX switches are entered on the command line preceded by a dash (-) character; multiple switches should be separated by a space. The switches that use alphabetic characters are case-insensitive.



Note, that the distribution comes with two shell scripts named `vlatex` and `vlatexp` that provide all necessary options to run VT_EX in PDF or PostScript mode with the L^AT_EX format. You may want to modify these scripts according to your personal needs, e.g., specify fast static path caching (see section 2.4).

VT_EX supports the following command-line switches and options:

- `$d` Select the DVI mode (default).
- `$p` Select the PDF mode.
- `$s` Select the PostScript mode.
- `C` Compatibility mode: all VT_EX extensions are disabled. DVI files created in compatibility mode are compatible with non-MicroPress drivers. *Not useful for PDF and PS modes.*
- `Ddir` DVI, PDF or PS (output) directory.
- `i` IniT_EX mode.
- `2` Enable bi-directional typesetting (TeX-XeT).
- `n#` where `#` is the initial T_EX running mode:
 - 0 batch
 - 1 nonstop
 - 2 scroll
 - 3 errorstop (default)
- `p` Path caching switches, see section 2.4
- `q` Quick run. This switch disables the output. While you need to compile L^AT_EX documents at least twice (and, often, three times), there is really no need to produce the formatted output on the passes before the last one. Using the `-q` switch will result in approximately 50% time saving on a PDF mode compilation, which would mean 25% overall time saving on a two-pass L^AT_EX compilation with `-q` used on the first pass, and 33% overall time saving on a 3-pass L^AT_EX compilation with two dummy runs.

- ov Resolve `.vf` files. In PDF and PS mode, this option is necessary to use virtual fonts. In DVI mode, this option will result in `.vf-free .dvi` files.
- oc Include complete fonts. This option applies to the PDF and PS modes only and disables font subsetting. In most cases there is no reasons to use this switch, since it leads to large (often, much large) output. However, if you intend to post-process the documents with Adobe Acrobat (Exchange), you may have to use this switch due to the general bugginess of the Adobe handling of subsetted fonts.
- j “Join” fonts. The option applies to PDF and PostScript modes. When a Type 1 font is used both in unmodified and modified (e.g., slanted) form, it will be embedded only once, thus saving space. However, additional PS or PDF code is required for switching between the two variants then. In long documents, this may occur very often, so that they may become actually longer.
- of *file* Specify the font map configuration file, see section 2.3.1.
- ox Enable G_ℙX.
- ox2 Enable G_ℙX with PostScript Level II support.
- ob2 Make G_ℙX recognize all 35 “base” fonts. (Default is to support the “Base 14” only.)
- = Specify alternative main configuration file (See the below section 2.2). This can be useful if V_TℙX is installed in a multiuser environment.
- 83 Disable use of “long” file names with V_TℙX/2, e.g., for running on a FAT drive.

See section 3.2 for further options related with PDF or PS generation only.

2.2 The main configuration file

Search paths and some general settings are determined by a “general” configuration file, whose name is `vtex.ini` (with OS/2) or `vtexlnx.rc` (with Linux), unless a different name is specified through the “-=” option on the command line (see above).

It is a typical Windows-style `.ini` file, which is divided into several sections, each defining its variables. Several sections are not applicable to the V_TℙX/Free distributions and have been omitted from this description.

The section [Directories] defines program paths. The following paths are important:

```
PGMDIR : VTℙX binaries, default vtex/bin/
INCDIR : generic TℙX include path, default vtex/src/
FMTDIR : Format directories, default vtex/fmt/
TMPDIR : Temporary directory, default vtex/tmp/
GRFDIR : Graphics file include directories, default vtex/src/
TFMDIR : .tfm include directory, default vtex/tfm/
VFSDIR : .vf include directory, default vtex/vfs/
ENCDIR : .enc include directory, default vtex/enc/
BIBDIR : Bibtex database files (.bib)
BSTDIR : Bibtex style files (.bst)
MSTDIR : Makeindex style files
```

Multiple directories should be separated with semicolons, and the trailing path separator must be included as well. Automatic subdirectory searches can be turned on on any of these directories by appending a `+` to the directory name.

The second important section is `[FINCLUDE]`. The `.tex` include path in $\text{V}\text{T}_{\text{E}}\text{X}$ is format-dependant. Assuming that you compile with a format named “`latexhv`” (HV-Math based $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2\epsilon}$), $\text{V}\text{T}_{\text{E}}\text{X}$ will check the `[FINCLUDE]` section to see if a private include path for this format is listed; if it is, it will be used rather than the generic `INCDIR` setting. This, in particular, allows to have similarly named styles for $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2\epsilon}$ and $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2.09}$ live side-by-side without conflicts.

The include path defined in the `[FINCLUDE]` section may reference the generic include path (`INCDIR`) with the `*` symbol. By default, we define

```
[FINCLUDE]
VTEX=*
PLAIN=*
LATEX=/vtex/l2e/*
```

which means that Plain-based or $\text{V}\text{T}_{\text{E}}\text{X}$ -based documents search only `/vtex/src`, while $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2\epsilon}$ documents first search `/vtex/l2e`, and then `/vtex/src`.

Automatic subdirectory searches can be turned on on any of these directories by appending a `+` to the directory name

Note that the `[FINCLUDE]` directory match works by prefix comparison. This means that if your format is actually `latexhv`, $\text{V}\text{T}_{\text{E}}\text{X}$ will first check `[FINCLUDE]` for `LATEXHV=`, if this fails, it will try `LATEXH=`, followed by `LATEX=` etc. With the default settings, the match will be achieved at `LATEX=`. The purpose of this logic is to allow compact specifications for a set of related formats.

In the section `[HUGETEX]`, initial values for $\text{T}_{\text{E}}\text{X}$ ’s internal `hyph_size`, `trie_size` and `trie_op_size` memory can be specified. Note that these sizes can grow dynamically, if required. The configuration file supplied with $\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$ sets:

```
[HugeTeX]
trie_size=120000
trie_op_size=20000
```

As to the `A4FIX` variable in the `[THIRDPARTY]` section, see the below section [3.5](#).

2.3 Configuring the font usage

2.3.1 The FM file

Font usage in the PDF and PostScript modes is primarily controlled by a configuration file, whose name is usually given on the command line using the option `-of`. The extension defaults to `.fm`. $\text{V}\text{T}_{\text{E}}\text{X}$ will search for this file along the $\text{T}_{\text{E}}\text{X}$ include path.

The `.fm` file is made up from a number of “sections”. Each section is started by a keyword, followed by the contents of the section included in braces.

The particular sections are:

LOCAL-OS2, LOCAL-LNX These sections define local variables, which can subsequently be referred to in the font map files; `LOCAL-OS2` is evaluated with OS/2 only, whereas `LOCAL-LNX` is for Linux. For instance, the `.fm` files supplied with $\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$ define in the `LOCAL-LNX` section:

```
TEXMF = "/usr/local/vtex/texmf/"
URW = "/usr/share/ghostscript/fonts/"
```

The `LOCAL-...` sections allow to use the same `.fm` file on different platforms in dual-boot environments.


PSRENAME Here the name of the file is to be given, which contains the list of base font substitutions; see section 2.3.3.

A4FIXTABLE The name of the file, which contains the list of fonts to be treated in a special way so as to fix PCL printing with Acrobat4; see section 3.5.

TYPE1 This section lists the aliasing (font map) files for Type 1 font files; for the format of these files, see the next section.

TRUETYPE This section lists the aliasing (font map) files for TrueType font files; for the format of these files, see the next section.

OPENTYPE This section lists the aliasing (font map) files for OpenType/Type 1 font files; for the format of these files, see the next section.

 The files listed here must reside in the \TeX include path – see section 2.2.

When no `.fm` file is given on the command line, a number of default settings come into effect:

LOCAL... No local variables are defined.

PSRENAME The file name defaults to `fontname.fix`

A4FIXTABLE The file name defaults to `a4type1.fix`

TYPE1 Only one single font map file named `aliasing.pst` (Win, OS/2) or `type1.rc` (Linux) will be used.

TRUETYPE Only one single font map file named `aliasing.tt` will be used.

OPENTYPE Only one single font map file named `aliasing.ot` will be used.

2.3.2 The font map files

These files establish the relation between the font names used by \TeX (i.e., the TFM's) and the font files (Type 1, TrueType or OpenType). With $\text{V}\TeX$, they are also named “aliasing” files. The files are ASCII and can be manually edited if needed.

An aliasing file is made up from two sections, separated by an empty line.

The first section lists all the directories that include referenced font files; these are `.pfb` and `.pfm` for Type 1, `.ttf` for TrueType, and `.otf` for OpenType. Each line in this section is in the format

`%<N> = <directory>`

where `<N>` is a unique (non-repeating) integer number and `<directory>` is a full directory path, ending with a slash. For example,

```
%1 = $PFM$
%2 = $TEXMF$fonts/type1/micropress/cm/
%3 = $TEXMF$fonts/type1/micropress/ams/
```

defines three directories that can be referenced below. Note, that the slash is used as a directory separator with OS/2, too! This is not a requirement, but using slash allows to maintain the same `.ali` files on Linux and OS/2. `$TEXMF$` refers to the variable `TEXMF`, that must have been defined in the `.fm` file—see above. The variable name `VTEX` is predefined and points to the $\text{V}\TeX$ root directory. (This is relevant for $\text{V}\TeX$ /Win. only.)

The second section of the file contains the list of fonts. Each line has the format

```
@<Font-name> = <File-name> %<access><pfb-dir>,<pfm-dir>[options]
@<Font-name> = <File-name> %<access><ttf-dir>[options]
@<Font-name> = <File-name> %<access><otf-dir>,<otf-dir>[options]
```

for respectively Type 1, TrueType and OpenType .ali files.

where

FontName: The PostScript FontName of the font in the resulting PDF/PS file, e.g., Times-Roman. This name *must* match the name given in the .pfb file exactly.

File-name: The .tfm file name for the font as referenced by T_EX. It must be the same as the name of the related .pfb file, unless the latter is explicitly specified.

access: Usually, nothing. Fonts that are built-in in the PDF or PostScript interpreter (“base fonts”) should have a ! mark to prevent their (unnecessary) downloading into the document. Access key should not be used in TrueType/OpenType .ali files.

Fonts that should be embedded in full should have a ~ mark. Full embedding is an escape provision in case you encounter a font which V_TE_X cannot subset. (Shall this happen, please let us know). See also g key below.

pfb-dir: The number of the directory that contains the .pfb file for this font (Type 1 .ali files only).

pfm-dir: The number of the directory that contains the .pfm file for this font (Type 1 .ali files only; the presence of .pfm file is not required and this entry is in no way used by the V_TE_X compiler).

ttf-dir: The number of the directory that contains the .ttf file for this font (TrueType .ali files only).

otf-dir: The number of the directory that contains the .otf file for this font (OpenType .ali files only). Notice that in the OpenType .ali files this entry is repeated.

Additional options that can be specified include

b=file-name: The name of the actual font file (basename), in case it differs from the TFM name; the extension (.pfb/.ttf/.otf) may be omitted.

r=map-name: Reencoding map file; the extension (.enc) may be omitted.

g GeX-loading: full processing of the font (Type 1 only) in GeX. This option allows to use Type 1 fonts that are not valid ATM fonts.

m=... MM flags (Type 1 only): See the document [MMsupp.pdf](#), which is supplied with V_TE_X for Windows, for more details.

Examples:

```
@cmr10 = cmr10 %2,1
```

```
@Symbol = psyr %!2,1,b="s0500001"
```

Note that the ‘Symbol’ font is built-in and so marked.



In V_TE_X/Free, the .pfm directories are never actually accessed, but because the syntax for this file is identical to the Windows version, you need to define at least a dummy directory.

2.3.3 Base font substitution

The font substitution table allows to use non-Adobe “Base 35” font files. For copyright reasons, these fonts usually carry incorrect /FontName’s; this may cause the problems as follows:

Assume you have an `.eps` file which uses Palatino, but does not embed it. Loading URW's Palatino clone will not help, since its `/FontName` is hardwired as `/URWPalladioL-Roma`, while the `.eps` program would look for `/Palatino-Roman`.

GeX overcomes the problem as follows: When the font is first loaded, its `/FontName` is replaced. The replacement table is given in the `fontname.fix` file; each line of this file lists two names: the name to be replaced first, and the replacement second. For example, the Palatino substitution is defined as

```
/URWPalladioL-Roma      /Palatino-Roman
```

Note that the Times, Helvetica and Courier fonts are special, as they must *not* be listed here: one may “silently” load (only!) URW's or MicroPress' clones for them.

2.3.4 Re-encoding

Type 1 fonts typically contain more characters than accessible directly. These additional characters, “hidden” inside the font implementation, include ligatures (usually, “fi” and “fl” only), accented characters, and others. The visible Type 1 characters are only a subset of what can be made available. To make things worse, the *default* layout of the third-party Type 1 fonts (usually, `StandardEncoding`) is not very suitable for \TeX processing since it does not support standard ligatures, either text or symbol, and since it comes with non-matching opening and closing quotes.

The layout of the CM fonts which are supplied with $\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$ is, however, already suitable for $\text{T}_{\text{E}}\text{X}$ and no re-encoding need to be applied to these fonts.

$\text{V}\text{T}_{\text{E}}\text{X}$ allows you to configure Type 1 fonts to follow a desired character layout with full ligature support. $\text{V}\text{T}_{\text{E}}\text{X}$ comes with several encodings, including `TeXBase1Encoding` (file `8r.enc`), which is used by the virtual fonts of the PSNFSS system.

Additional encodings can be easily made by the user. See the next section for the details on the format of these files.

Since there may be more than one $\text{T}_{\text{E}}\text{X}$ -font based on a single Type 1 font and multiple encodings, they have to be distinguished by different font names and $\text{T}_{\text{E}}\text{X}$ metric `.tfm` files. Note that the font names can be chosen arbitrarily in this case. With fonts to be reencoded, the entries in the aliasing file contain an additional option to support this:

```
r=<encoding file name>
```

The extension `.enc` of the encoding file may be omitted



For each reencoded font there must also be an entry which references the same Type 1 font without reencoding under its actual font name!

Example: The following lines within the font map file declare two instances of the Times-Roman font: ‘raw’ and reencoded:

```
@Times-Roman = n0210031 %!2,1
@Times-Roman-8r = ptmr8r %!2,1,r="8r",b="n0210031"
```

2.3.5 Encoding files

The available encodings are stored in files with extension `.enc` within the `ENCDIR` subdirectory. Each file should declare a PostScript array of exactly 256 symbol names:

```

/MyEncoding [
    /.notdef /.notdef /.notdef /.notdef
    ....
    /A /B /C /D /E /F
    ....
    /a /b /c /d /e /f
    ....
] def

```

Comment lines are allowed and should start with the % character. These names define the layout of the reencoded font. Note that if you specify a symbol name that actually does not occur in the base font, the entry will be blank (`/.notdef`).

The number of entries in the encoding vector must not exceed 256.

2.4 Path caching

\TeX may spend a considerable amount of time searching for files to be read (font metrics, packages, `.tex` files to be included etc.). This happens because while the CPU speed of the modern computers makes the document compilation fast, the file search performed by the operating system could be very slow. This bottleneck can be eliminated with *path caching*.

The path caching in \TeX comes in two models:

- ▷ Dynamic path caching: at the beginning of each compilation, \TeX builds a full list of the contents of the include directories in memory; when a file needs to be opened, \TeX uses the memory table to get directly to it, without the need for additional searches.
- ▷ Static path caching: the cache is built once and stored as a file; on each compilation, the cache is read and is used to locate files. The path cache files have extension `.ffs` and are stored in the format directory of \TeX , usually `VTEX\FMT`. Since different \TeX formats may have different search paths, you need a separate file for each format you use. For example, `latex.ffs` is the name of the path cache file for \LaTeX .

Static path cache is more efficient than dynamic since it is faster to load a cache from disk than to search the entire include paths (even once). However, static path cache may need to be rebuilt when you modify the include paths or add more files to the include directories, otherwise searching for the files, which have been added or moved, will be unnecessarily slow.

Furthermore, the static path cache is not entirely safe: If the dynamic path cache will find exactly the same files as in the no-cache mode, the static cache may produce different results—if you have neglected to rebuild the cache file. Below we describe a hypothetical situation which may lead to different results with the dynamic (or no-) cache and the static cache. (The example uses the Plain \TeX format; path names are for OS/2).

1. Assume that a document contains the command `\input test`.
2. Assume that the search path for the Plain \TeX files includes `C:\MYFILES\+` (the `+`-notation indicates a recursive search on `C:\MYFILES` and all its sub-directories.) Assume that `C:\MYFILES\+` contains a subdirectory `OLD`.
3. Make a one-line file, `test.tex`, containing

```

\message{Old file!}

```

and place it into `C:\MYFILES\OLD`.
4. Build the static path cache for the Plain format.

5. Make a one-line file, again called `test.tex`, but containing

```
\message{New file!}
```

and place it into `C:\MYFILES`.

6. Now, compiling with the dynamic cache or without caching will result in the message `New file!` from the `C:\MYFILES\test.tex` file; but compiling with the static cache will yield `Old file!` from the `C:\MYFILES\OLD\test.tex` file.

Naturally, this type of situation will arise only if your cache file has not been updated, and only if you have duplicate files (which is always error-prone).



Note, however, that the current directory (and the directory of the main document, which may be different) are always searched *before* the path cache. Thus, a file in one of these directories is guaranteed to have precedence over a duplicate one, whose location has been memorized in the path cache. As a result, a static cache needs *not* to be rebuilt when include files or special macro packages are stored together with a document in one directory.

Path caching acts on macros and include files, which are read by \TeX , as well as on the \TeX font metrics files (`.tfm`) and the virtual fonts (`.vf`).

The command-line switches that govern cache operations are:

- pu Use dynamic cache.
- pw Write static cache file.
- pr Read static cache file.

Example:

```
vtex <further options> @latex sample2e
```

compiles the standard \LaTeX sample file without path cache. This is the slowest.

```
vtex -pu <further options> @latex sample2e
```

compiles the same file using dynamic cache (faster).

```
vtex -pw @latex
```

writes out the path cache for the \LaTeX format. Note that this operation does not compile any document; it just writes the cache and exits.

```
vtex -pr <further options> @latex sample2e
```

compiles the file using static cache (fastest).

There are two modifiers allowed after the path cache switches:

- `t` produces a console trace. You can use it to detect duplicate file names and see which of the duplicates will actually be used by \VTeX . This modifier can be used with `-pw` and `-pu` (`-pwt`, `-put`).
- `0` specifies cache-only search. Usually, if the required file is not found in the cache, \VTeX will attempt to find it on disk anyway. With this modifier, \VTeX will not spend time looking for the files on disk. This modifier is used with both `-pu` and `-pr`.

The additional time savings from the use of this switch come mostly from non-searching for `.vf` files (which applies only if you specified the `-ov` switch).



Using `-pu0` is recommended under normal circumstances.



`-pr0` saves additional time over `-pr`, but may cause unexpected errors if the static cache file is out of date. For example, if you copied a new style into a directory which is on the include path, with `-pr0` it would not be seen by \VTeX until you rebuild the cache. Use this switch only, if you are aware of the consequences!

3. PostScript and PDF generation

3.1 General

The \TeX document compiler supports generating PDF and PostScript files in addition to the traditional DVI files.

This has been designed to be totally transparent to use. No changes are required to your document source. All additional features, such as the hyperlinks and outline entries supported by the Portable Document Format, are supported via `\special` commands, rather than syntax changes.

The PDF and PS backends also support *most* of the \TeX extensions. This chapter explains most of the PDF/PS mode specifics. Omitted are the three more interesting parts, which are explained in separate manuals:

- ▷ \GEX , the integrated PostScript→PDF compiler, see [gex.pdf](#).
- ▷ Animated GIFs inclusion, see [animgif.pdf](#).
- ▷ Form and JavaScript support, see [forms.pdf](#).

Note that most features shown in the latter document require Acrobat 4 or better!

3.2 Command-line options

To create a `.pdf` file, you should supply the `-$p` switch to the \TeX document compiler. For example (OS/2)

```
vtex -$p[(PDF options) [Other options] [@format] mydoc[.tex]
```

or (Linux)

```
.../vtexlnx -\ $p[(PDF options) [Other options] [@format] mydoc[.tex]
```

The counterpart for the PS mode is the `-$s` switch (same syntax).



Under most Unix shells you will need to precede the `$` with the `\` escape. Alternatively, you can use `@` instead of `$`. Furthermore, some Linux shells are unhappy about `(` as well; you can use `{` instead.

The `[(MODE options)]` part includes the options specific for PS and PDF mode, described below. Note that this part may not contain embedded spaces; all options are separated by commas.

The mode-specific options are

- c** `c=<number>` specifies the compression level (0 through 9). This option affects only Flate-compressed parts of the document. Higher number indicates stronger compression (meaning: smaller `.pdf` file and slower compilation).
- d** `d` specifies to end lines within the `.pdf` files with the DOS `[CR][LF]` sequence, rather than the default `[CR]`.
- !** Do not load the “base” Type 1 fonts into \GEX unless they are actually required, thus speeding up \GEX .

- f **f=<output spec>** specifies how fonts should be embedded into the document.
- g **g=<output spec>** specifies how graphic images should be embedded into the document.
- h **h=<dimen>** specifies the document MediaBox height (see below).
- r **r=<number>** specifies the build resolution for .if4 fonts.
- t **t=<output spec>** specifies how text should be embedded into the document.
- v make V_T_EX embed a font one time only, even if it is used with multiple encodings.
- w **w=<dimen>** specifies the document MediaBox width (see below).
- **_=<char>** replaces underscores in internally generated font names with the indicated character, for compatibility with Acrobat Reader 3; this is currently relevant in conjunction with MM fonts only.

The <output spec> may include these letters:

- a ASCII85 output
- h Hex output
- b Binary output (default)
- c CFF conversion of Type 1 fonts (with f= key only), see section 3.4.
- n No compression of the output (default)
- f Flate compression of the output

In the case of the conflicting options (for example, n and f), the last specification is taken. Thus, f=nb is the same f=b. The default options do not need to be specified.

The following specifications are particularly useful:

- ▷ **f=a,g=a,t=b** : when used in PDF mode, this will create an ASCII-editable .pdf file. Note that t=b is not a misprint: the text portion of the generated .pdf file always contains only printable characters and does not need to be further encoded. Note further, that here “ASCII-editable” .pdf files means that such files *usually* can be loaded into a text editor, changed in a minor way, saved, and then loaded into the Acrobat Reader, which should be able to repair them. *usually* does not mean *always*, of course. Naturally, to make manual corrections to the .pdf format you should know the format.
- ▷ **f=cf,g=f,t=f**: this will create the smallest .pdf file.



Use of Flate compression in the PS mode is supported, but should be restricted to the cases when the PS output is to be processed by a PS Level III processor. These include GhostScript, Acrobat Distiller 4, but not the majority of the printers.

3.3 Font usage in PDF and PostScript documents

V_T_EX-built PDF and PostScript files currently support these font formats:

Type 1 This is the recommended format.

IF4 V_T_EX native fonts; currently not supplied with V_T_EX/Free.

TrueType In general, we strongly recommend *not using* TrueType fonts. Some reasons for this are:

- ▷ Type 1 fonts almost always deliver higher quality.
- ▷ TrueType fonts cannot be used in/with G_ℙX.
- ▷ Currently, V_TℙX cannot reencode TrueType fonts.
- ▷ TrueType fonts often cause problems with Acrobat Reader on non-Windows platform.

OpenType For the time being, use of OpenType fonts with V_TℙX is very much theory, because there are no tools to create the required T_ℙX metrics for these fonts. Furthermore, they can be used with PDF mode only, and cannot be loaded into G_ℙX. Future improvements are to be expected in this area.



V_TℙX/Free does *not* support the METAFONT format.

To avoid generating bad output files, V_TℙX checks for the presense of an acceptable font file whenever the font is referenced in the document. Further, before starting the compilation of a document, V_TℙX ensures that all fonts referenced in the format file correspond to valid (i.e. in a supported file format) fonts.

If a valid font is not available, V_TℙX will generate an error and substitute the `nullfont`. Note that this behavior may produce documents that are formatted differently as `.dvi` and as `.pdf`.

At the time of the font availability check V_TℙX also decides which format and font file to use. The check is done as follows:

- ▷ If the font is listed in one of the the font map files, it will be used. Otherwise,
- ▷ if the font file with extension `.if4` is present in the `VTℙX\IF4` subdirectory, it will be used.

3.4 CFF conversion


One of the major enhancements in version 7.3 is the ability of the V_TℙX compiler to convert Type 1 fonts to the new CFF format. The Compact Font Format is much more efficient than Type 1; it cuts down the size of the font files by a factor of 2 or more. Since in a typical T_ℙX documents the fonts contribute most to the size of the `.pdf` output, such documents, when recompiled with CFF compression enabled, often would become much smaller.

The operation of the CFF converter is transparent to the end-user; the CFF converter is enabled with the `-$p(f=c` command-line key, or, for the most possible compression, with `-$p(f=cf`: this will perform Flate compression on top of the CFF conversion.



CFF conversion can be performed in either PDF or PS modes; however, while in the PDF mode, the PDF readers (both Adobe Reader and GhostScript) understand embedded CFF fonts, in the PS mode, GhostScript is the only application which is able to process `.ps` files with embedded CFF fonts. Adobe applications are mostly unable, even if it was Adobe that invented the CFF format and published the specs for CFF embedding into the `.ps` files. CFF embedding in the PS mode generally should be avoided; however, the `-$s(f=c` switch is available for special-purpose processing.

3.5 PCL printing with Acrobat 4

 This section applies to PDF mode only.

Acrobat 4, while a genuine improvement over Acrobat 3, introduced some severe bugs. Of primary importance is the mishandling of fonts, particularly the \TeX math extension fonts such as `cmex10`. On many non-PostScript devices (including all HP printers in the PCL mode), the characters from the extension fonts will be printed only partially. This will occur—at least—with Acrobat 4 prior to version 4.05.

Even if Adobe has released version 4.05 which does not suffer from the problems described here, a great number of people still have version 4 installed.

This problem affects not just \TeX , but other pdf-making software (like the Distiller); it also seems to affect all math extension fonts which exist today, regardless of the specific tools used to construct the fonts; it equally applies to MicroPress fonts and to AMS/BSR fonts. There is nothing wrong with the fonts themselves, of course; the problem resides in the Adobe's rasterization code which makes bad PCL softfonts.

While we do not have full information about the causes of the bug, \TeX (starting with v6.56) will try to cope with the problem in various ways. This is controlled by the variable `A4FIX` in the `[THIRDPARTY]` section of the configuration file `vtex.ini`, see above (2.2). Valid settings are:

`A4FIX=0` No correction for the Acrobat 4 printing problem. This is identical to the pdf \TeX approach or the default approach of dvips.

`A4FIX=1` Correction for the Acrobat 4 problem via reencoding the font out of 0...31 range. This, however, *does not always help* and it *is ignored for CFFs* (see section 3.4). This is the default behaviour since \TeX version 6.4. The correction applies to fonts specifically listed in the `a4type1.fix` file (see below).

`A4FIX=-1` Same, but apply the correction to all fonts. Note that the correction will have no effect on fonts that are already “full”, like the EC fonts.


`A4FIX=2` Correction for the Acrobat 4 problem via FTM changes. This works always for Acrobat 4, but fails with Acrobat 3, i.e., *the resulting PDF file cannot be displayed with Acrobat 3 any more*. The correction applies to fonts specifically listed in the `a4type1.fix` file.

`A4FIX=-2` Same, but apply the correction to all fonts.

A file named `a4type1.fix` contains (in free format) the list of fonts to be relocated with `A4FIX=1` or 2. A different name for this file may be specified in the FM configuration file; see section 2.3.1.

The current list of fonts is:

```
cmex10 cmex7 cmex8 cmex9 euex10 euex8 euex9 ifex10 hvex10 hvexb10
tmex10 tmexb10 mvex10 mvex7 mvex8 mvex9 baex10 paex10
xcex10 xcex7 xcex8
```

 The shell scripts supplied with the \TeX /Free distribution call the \TeX program—in PDF mode—with the option `f=cf`, i.e., fonts are embedded as CFFs. Thus, no correction for the above bug is provided by default. If you want to assure that your PDF documents can be printed flawlessly from Acrobat 4.0, too, this must be changed to `f=f`. Thus, the setting `A4FIX=1`, as described above, comes into effect, at the cost of larger output file size.

3.6 Media size

The default MediaBox generated by $\text{V}\text{T}_{\text{E}}\text{X}$ is based on the settings of `\hsize` and `\vsize`. If your format is Plain, $\text{V}\text{T}_{\text{E}}\text{X}$, or $\text{A}\text{M}\text{S}\text{T}_{\text{E}}\text{X}$, you usually will not have to specify the MediaBox sizes manually, using the `w` and `h` command line options.

With $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, the MediaBox sizes are set to `\paperwidth` and `\paperheight`, provided that the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ format has been created from the file `texmf/vtex/config/latex.tex` which is part of the $\text{V}\text{T}_{\text{E}}\text{X}$ /Free distribution. The command line options `w` and `h` should specify a fallback MediaBox, which is to be used when a $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ class does not define the paper size. (`\hsize` and `\vsize` are not correct with $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.) For example, the `latex` shell script issues `-p(w=21cm,h=29.7cm)` in order to make A4 the default MediaBox.

Setting the MediaBox from within the document is accomplished through the non-standard `\dimen`'s `\mediawidth` and `\mediaheight`. Thus, the desired values can safely be stored within the $\text{T}_{\text{E}}\text{X}$ source. Furthermore, using these registers explicitly would be helpful, if the MediaBox should be set differently for different pages. This would happen with, for example, a document that is to contain portrait as well as landscape pages. Note, however, that this will make the document source incompatible with other $\text{T}_{\text{E}}\text{X}$ implementations!

To summarize this, the PDF media size is determined, in descending order of precedence, by

1. `\mediawidth` and `\mediaheight` (if specified explicitly),
2. `\paperwidth` and `\paperheight` (with $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ only),
3. the command line options `w` and `h`,
4. `\hsize` and `\vsize`.

3.7 Links

One of the advantages of `.pdf` files is the ability to produce hyperlinks. On the low level, this is accomplished by $\text{V}\text{T}_{\text{E}}\text{X}$ `\special` commands:

`!ref...` Start a hyperlink

`!endref` End a hyperlink. This `\special` has no arguments.

`!aname` Define a target.

Both `\special{!aref...}` and `\special{!aname...}` have additional arguments. The most important one is the label specification which should be the first and has one of three formats:

`*<Number>` This denotes a local (non-exportable) label.

`Name` This denotes a global (exportable) label. `Name` generally should contain only alphanumeric characters. Name resolution is case-sensitive.

`!<Number>` This denotes a page number label.

Note that global labels are accessible from other `.pdf` documents while local labels are not; on the other hand, local labels result in somewhat smaller document files. Note also $\text{V}\text{T}_{\text{E}}\text{X}$ always generates a label `PageNNN` on each document page, which can be used for external references.

Example of a simple hyperlink follows:

```
See \special{!aref Gnues}Gnues\special{!endaref}
for more details.
...
```



```
...
...
\special{!aname Gnues}\section{Gnues}
```

To produce a link to another .pdf file, use the global (or page) label format and precede the name with <f=filename> specification. For example,

```
See documentation on
\special{!aref <f=pdfspec.pdf>!8}pdf\special{!endaref}
%% Refer file pdfspec.pdf, Page 8.
file format for more details.
```

Note that if you do not provide the target file for an external link given in this form you will a warning

```
Invalid file specification object.
```

from the Acrobat Reader. This error does not mean a structural problem in the .pdf file, but merely an absense of the target file.

The appearance of the link itself can be controlled in two ways:

- ▷ Via T_EX itself, you can, for example, underline the link, or color it, or produce a colorbox over the link.
- ▷ Via .pdf syntax you can specify the attributes of the link. The attribute specification can appear in the !aref command after a semicolon and a=; everything that follows a= through the end of the special is passed over to Acrobat as the link attribute. For example,

```
\special{!aref Name;a=</Border [0 0 0]>}
```

will produce a link without the default border. Specifying

```
\special{c"FF0000}\special{!aref Name;a=</Border [0 0 0]>}
.....
\special{!endaref}\special{c"000000}
```

will produce red-colored links without border. For more details on link attributes, see the PDF documentation, as published by Adobe. (The pdfspec.pdf file can be found on the www.adobe.com Web site.)

Note. The \special{!aref} and \special{!endaref} commands should come in pairs: for each \special{!aref} there should be a matching \special{!endaref}. Furthermore, a matching pair should be given on the same T_EX nesting level. For example,

```
...
\smallskip
\special{!aref Label}See section 5
\special{!endaref}
for more details.
...
```

is incorrect and will cause a backend error, since \special{!aref} appears in the vertical mode, while \special{!endaref} appears in the horizontal. One way to avoid (or minimize) such nesting errors is to always place \leavevmode before a \special{!aref}; the other is to place everything (\special{!aref}, \special{!endaref} and the text between them) into an \hbox.



In the PS mode, hyperlinks (as well as the outline entries, described in the

next section, are emitted as `pdfmarks`. `pdfmarks` are ignored by most of PS processors but processed by PS→PDF converters (distiller, GhostScript in PS2PDF mode). Emission of `pdfmarks` in V_T_EX's PS mode makes sense only if you intend to post-process the `.ps` output and then feed it to a PS→PDF converter.


3.8 Outline

You can supplement the `.pdf` file with an outline. The low-level interface for this is the V_T_EX `\special{!outline}` command. The general syntax of this command is


```
\special{!outline Label;i=Id,p=Parent,s=Status,t=Text}
```

where


- ▷ Label is as described in the previous section (local, global, or page). For local or global tags you should have a defining `\special{!aname ...}`.
- ▷ The Id (`i=`) is a unique integer number identifying this outline record. You must not have two outline entries with the same id.
- ▷ The Parent (`p=`) is the id number of the parent entry, or 0 for the topmost entry.
- ▷ The Status (`s=`) is the letter 'o' (open) or 'c' (closed) identifying the initial state of the outline at this node. It is used only for the non-leaf nodes of the outline (i.e. the nodes that have subchildren). Due to bugs in many version of Acrobat, we strongly recommend keeping all entries closed.
- ▷ The Text (`t=`) is the text for the outline. Since this text is not formatted by T_EX, you should use only “normal” characters within it. For example, putting `\TeX` within the outline entry text is a very bad idea.

 V_T_EX version 6.12 and later processes the `\char` commands within the outline title specifications. You can use this to place accented letters into the outline entries. An example L^AT_EX file, `outlchrs.tex`, is supplied in the `texmf/doc/vtex/examples` subdirectory.

The `\char` command should be followed by a decimal, hex (with a leading `#`), or an octal number (with a leading `o`). Use of decimal is discouraged, since it may lead to errors due to varying size of a decimal number. Hex and octal are safe since they always require a fix number of digits (2 and 3).

 If you are a L^AT_EX user, please consider using a predefined high-level style rather than programming the low-level commands. Two such styles are the generic `hyperref` package by S. Rahtz and much simpler and smaller `pdf.sty`. Even if you decide to develop your own macros (for Plain you will have to), looking at `pdf.sty` will save you time. See the files `texmf/doc/latex/hyperref/manual.pdf` and `texmf/doc/latex/misc/pdf.pdf`.

3.9 Transitions

 The information in section applies to the PDF mode only.

Starting with version 6.4, V_T_EX supports Acrobat pdf transition effects. Transitions can be used to spice up a pdf-coded presentation. Since there is no high-level support at this time, the transition effects should be coded using the `\special{!trans ...}`.

The special takes the form:

```
\special{!trans <CODE>[,<TIME>]+}
```

The <TIME> part is optional; if present, it indicated the transition time in milliseconds. The possible transition codes are:

- ▷ W0, Wipe at 0 degree angle.
- ▷ W90, Wipe at 90 degree angle.
- ▷ W180, Wipe at 180 degree angle.
- ▷ W270, Wipe at 270 degree angle.
- ▷ D, Dissolve.
- ▷ BH, Blinds, horizontal.
- ▷ BV, Blinds, vertical.
- ▷ G0, Glitter at 0 degree angle.
- ▷ G270, Glitter at 270 degree angle.
- ▷ G315, Glitter at 315 degree angle.
- ▷ XI, Box, In.
- ▷ XO, Box, Out.
- ▷ SHI, Split, Horizontal, In.
- ▷ SVI, Split, Vertical, In.
- ▷ SHO, Split, Horizontal, Out.
- ▷ SVO, Split, Vertical, Out.

These are the only transition values supported by the Acrobat. An example Plain TeX file, `trans.tex`, is supplied in the `texmf/doc/vtex/examples` sub-directory. This file shows all available transitions.

3.10 Thumbs



The information in section applies to the PDF mode only.

Starting with version 6.5, VTeX supports page thumbs generation. Page thumbs are shown by the Adobe Acrobat reader when the “Thumbnails” sub-Window is open.

VTeX supports two `\special{...}` commands for thumbs generation:

- ▷ `\special{!thumb ...}`
- ▷ `\special{!dthumb ...}`

The first command affects only the current page; the second sets the default to be used on the rest of the document. Both `\special`’s have identical syntax and parameters.

There are also two ways the thumbs can be generated. You can specify the thumb image as an external file, or ask VTeX to generate a low-res page approximation dynamically. The first option is selected with the `f=` parameter in the `\special{...}` syntax; all the other parameters select the second option.

The valid parameters are:

- `f=` Select an image file. The image file must be a bitmapped graphics file (not `.eps`). This option is mutually exclusive with the rest.
- `w=` Define (in pixels) the width of the thumb to generate.
- `h=` Define (in pixels) the height of the thumb to generate.
- `c=` Define (hex, rrggbb) the color of the thumb.
- `b=` Define (hex, rrggbb) the background color of the thumb.

Multiple parameters are separated with a comma. Examples:

- ▷ `\special{!thumb f=pic.gif}` loads the thumb for the current page from the `pic.gif` file.
- ▷ `\special{!thumb h=110,w=85}` creates 110×85 black and white thumb for the current page.
- ▷ `\special{!dthumb h=110,w=85,c=ff0000,b=00ff00}` creates 110×85 red-on-green thumbs for all subsequent pages.

Notes:

- ▷ specifying no parameters disables thumbs either for the current page only (`\special{!thumb}`) or for all subsequent pages (`\special{!dthumb}`).
- ▷ The global default is no thumbs.
- ▷ After a (`\special{!thumb...}`) the thumb settings revert to those given by the preceding (`\special{!dthumb...}`), or, if there were not one, to the no-thumbs default.
- ▷ Acrobat rescales the thumbs in all cases. Thus, only the proportion between the `w=` and `h=` settings matters; doubling the values of `w=` and `h=` will not change the thumbs appearance except for a small improvement in quality.

3.11 Annotations

VT_EX allows to place text annotations into the PDF file. This could be quite useful if you are working on a document with a colleague: text annotations could be used as “internal notes” to be deleted later.

This note has been produced with:

```
\verb+\special{!annotate w=5cm, h=4cm, t=For exa...}+.
```



The parameters understood by this `\special` are:

`t=` Annotation text, should always be the last parameter.

`w=` Open annotation width (dimen).

`h=` Open annotation height (dimen).

`a=` Annotation attributes (enclose in `<...>`, see PDF reference for additional details).

For example, use `a=</C [1 0 0]>` to have  annotation appear in red, rather than the default yellow, for example .

```
\special{!annotate a=</C [1 0 0]>,t=like this}
```

The text within annotation may contain embedded `\char` commands which are resolved as explained in section 3.8.

3.12 Non-Latin characters in Outlines and Annotations

Acrobat 4 allows to put non-Latin characters into outlines and annotations. These characters are coded in Unicode. To simplify the task of entering such text, VT_EX 6.5+ adds a couple of new primitives:

- ▷ `\unicode` is a 256-character table which defines the Unicode values for all characters. This table behaves in a way similar to the `\catcode` or `\lccode` tables.
- ▷ `\unithe` expands tokens into Unicode double-tokens, similarly to the usual `\the` operator.

Equipped with these operators, we can easily enter non-Latin text into annotations.

Similar technique works with outlines. Notice that `\char254\char255` is the usual Unicode text prefix marker; it must be supplied to inform the Adobe Reader that the following text is indeed in Unicode.



Unicode text in annotations/outlines will be visible only on computers that have the required system fonts. In Win95/98/NT this specifically means installing the multilingual support options of the operating system.

3.13 Color stack issues

\TeX color support is a later addition to \TeX ; Knuth's \TeX kernel does not support color directly. As a result, the color commands do not obey either \TeX syntax grouping or \TeX boxes. In most cases, this is not a problem; but in some cases, the color may *leak* beyond what was intended, sometimes to the next page. To allow a macro designer to handle colors properly, $\text{V}\TeX$ now adds two more `\special{...}` commands:

```
▷ \special{G{}
```

```
▷ \special{G}}
```

The first of the commands is essentially equivalent to $\text{G}\TeX$'s `\special{pS: gsave}`; the second is `\special{pS: grestore}`. However, these commands function even when $\text{G}\TeX$ has not been enabled.

For $\text{V}\TeX$ 'nically minded: in $\text{V}\TeX$, $\text{G}\TeX$'s kernel is active even if $\text{G}\TeX$ interpreter has not been enabled (i.e. no `-ox` switch given). It handles commands which deal with the colors and page transformations; thus some $\text{V}\TeX$ specials merely call $\text{G}\TeX$, whether it is enabled in full or not. Besides the two `\special`'s listed above, this applies to the rotation specials `\special{r{...}}` and `\special{r...}` as well as the color specials `\special{c...}`, both explained elsewhere. When the `-ox` switch given, some of these commands (`\special{r...}`, for example) trigger full $\text{G}\TeX$ initialization.

3.14 Direct PDF/PS writing

The `\special{!direct}` (or `\special{!=}`) is used to copy the text into the .pdf file directly.

Generally you should avoid using this `\special` unless you are both very familiar with the PDF syntax *and* the understand how $\text{V}\TeX$ generates the output code. For example,

```
\special{!= 1 0 0 rg}
```

will correctly turn the color to **red** but may cause color errors later (since the code generator would not be aware of the color change.) Even safe-looking

```
\special{!= q 1 0 0 rg}
.....
\special{!= Q}
```

could lead to errors if a color, font, or CTM change occurred in the bracketed code.

3.15 PDF information special



The `\special{!pdfinfo...}` special command is used to set the general document settings. The tail of the command can contain one or more of the following tags:

- ▷ `a=<...>` Define the Author of the document.
- ▷ `t=<...>` Define the Title of the document.
- ▷ `s=<...>` Define the Subject of the document.
- ▷ `k=<...>` Define the Keywords of the document.
- ▷ `p=<...>` Define the Page Mode of the document.

The Author, Title, Subject and Keywords fields can be viewed in the Acrobat Reader by going into the **[File]/[Document Info]** selection. The Page Mode field defines how the document should be opened by the Acrobat. For example,

- ▷ `/UseOutlines` means that the document should be opened with the outline visible.
- ▷ `/UseThumbs` means that the document should be opened with the thumbnails visible.
- ▷ `/FullScreen` means that the document should be opened in the full screen mode.

This document, for example, was compiled with the `p=</UseOutlines>` option. Notice that specifying illegal keys for Page Mode may lead to Acrobat crashes.

-  If you need to set more than one information parameter, use multiple `\special`'s.
-  In the PS mode this information is emitted as `pdfmark`'s.

3.16 Bitmapped graphics inclusion

The $\text{V}\text{T}_{\text{E}}\text{X}/\text{Free}$ compiler features all of the bitmapped graphics filters also found in the windows version: **PNG, JPEG, BMP** (Windows BMP!), **GIF, PCX, TARGA** and **TIFF** images can all be included directly. (TIFF support is not yet available with the OS/2 version, however.)

As to JPEG, more modern types (i.e. progressive) will be included correctly, but Acrobat Reader 3 will not be able to show them right (gray rectangle only); Acrobat 4+ *will* show them.

Bitmapped graphics inclusion should be done through the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ macro package **graphicx**; see the document `texmf/doc/latex/graphics/grfguide.pdf` for a general description of the package. This section explains how $\text{V}\text{T}_{\text{E}}\text{X}$ processes bitmapped files in the PDF and PS modes and the additional `\includegraphics` keys that can be of use in fine-tuning the performance.

3.16.1 Image size

The physical size of bitmapped graphics on the printed page can be specified in two ways:

- ▷ The desired height and/or width can be specified through the `height=` and `width=` keywords of the `\includegraphics` command. Upon rendering to an output device, the original bitmap will be scaled to these dimensions.

- ▷ Use the keyword `atres=<resolution>`. $\text{V}\text{T}_{\text{E}}\text{X}$ will then determine the size of the graphics from its size – in pixels – and the given resolution. When printed at this very resolution, the bitmap needs not to be scaled; when the resolution of the output device is different, the bitmap will be scaled appropriately.

3.16.2 Compression and fine-tuning

Few ground rules to start with:

1. $\text{V}\text{T}_{\text{E}}\text{X}$ supplies two distinct techniques for loading the bitmapped files: direct processing and loading into the integrated PostScript interpreter, $\text{G}_{\text{F}}\text{X}$. In many cases the results are identical, but there are several differences, explained below. The default behaviour is to process the pictures directly: this is always faster. However, there are some advantages in going through $\text{G}_{\text{F}}\text{X}$: for example, you can post-process the images with the $\text{G}_{\text{F}}\text{X}$ imaging plugins (see the [G_FX](#) documentation).

To load a bitmapped image for $\text{G}_{\text{F}}\text{X}$ inclusion processing, use the `viagex` key for the `\includegraphics` command.



Note, however, that the default shell script for running $\text{V}\text{T}_{\text{E}}\text{X}$ in PostScript mode does *not* turn on $\text{G}_{\text{F}}\text{X}$. You may change this by adding the `-ox` option, if you want to use $\text{G}_{\text{F}}\text{X}$'s facilities in PS mode.

2. The default processing logic for the direct picture inclusion is to try to preserve the structure of the image as it comes from the original file. For JPEG images, it means graphics inclusion of DCT-compressed data, exactly as it was stored in the source file; for other images it means copying the color space as specified in the original file. Assuming that the original file was optimally built, this logic produces optimal image in the PDF file as well as minimizes the operations on the image.

Use the `repackjpeg` key for the `\includegraphics` command to force a JPEG image to be DCT-decompressed.

3. The default processing logic for the `viagex` graphics inclusion is to fully unpack the image. This means applying DCT-decompression to the JPEG files as well as de-indexing indexed graphics files. The rationale is make the image data easily accessible to the imaging plugins; the drawback is that in many cases this will cause larger output.

You can force the image data to be re-indexed with the `reindex` key for the `\includegraphics` command ($\text{G}_{\text{F}}\text{X}$ mode only).

4. In order to force minimal size of directly included images, you can force $\text{V}\text{T}_{\text{E}}\text{X}$ to apply CCITT or DCT compressions. For monochrome images, CCITT often produces much more compact results than the default Flate compression. For the color images, especially the photographs, DCT often improves on Flate, *even when DCT is lossless*. Note that CCITT and DCT compression are *not* controlled by the command line option `-$p(g=f`.

The keys for the `\includegraphics` command are `ccitt` and `dct`.

5. CCITT compression applies to monochrome images only, including image-masks, and has no effect on color images. It may produce worse results on some pictures which are drawn rather than scanned (specifically, if they contain gray areas emulated with bit patterns).
6. The DCT compression by default is lossless; but often you do not mind losing some data, if this does not result in clear loss of quality and reduces the PDF size. The `dctquality=###` key for the `\includegraphics` command

specifies the quality loss you are willing to sustain; with the default value of 0, there is no loss, the largest allowed value is 1000000 (loss of most of the data).

Case study: monochrome GIF

The `herzjesu.gif` sample file is a monochrome .gif image; the original file size is 18396 bytes. This is one case when the default loading procedure will produce non-optimal results: GIF's are always built around a 256-color palette, even when only two colors (monochrome!) are needed. The default loading of this file, done with this code

```
\documentclass{article}
\pagestyle{empty}
\usepackage{graphicx}
\begin{document}
\includegraphics[width=1in]{herzjesu.gif}
\end{document}
```

will result in a 618k PDF file (uncompressed), or a 36k PDF file (flate-compressed). If the image were loaded via `GEX`, with

```
\includegraphics[width=1in,viagex]{herzjesu.gif}
```

the results will be even worse: 1815k (uncompressed), or 48k (Flate-compressed). However, repacking the image, accomplished with

```
\includegraphics[width=1in,viagex,reindex]{herzjesu.gif}
```

will bring the size down to 77k (uncompressed), or 9148 bytes (flate compressed).

👉 Reindexing can substantially reduce the size of monochrome GIF images, or any image that has been incorrectly packed.

Case study: Photo Bitmap

The `mac400.bmp` file is a photo bitmap. The default inclusion of this image file, done with

```
\pagestyle{empty}
\documentclass{article}
\usepackage{graphicx}
\begin{document}
\includegraphics{mac400.bmp}
\end{document}
```

will result in a 402k PDF file (Flate compression). However, the use of lossless DCT compression, as in

```
\includegraphics[dct]{mac400.bmp}
```

will result in only 369k file. With

```
\includegraphics[dct,dctquality=10]{mac400.bmp}
```

the size of the file will decrease to 221k, without much quality loss. Specifying `dctquality=100` makes the size fall to 70k, with some loss for printing, but not for the screen display; at `dctquality=1000` the file size will be down to 20k (with a very visible quality degradation).

👉 Use of DCT compression often decreases the size of the PDF files, even when the compression is lossless.

👉 The `dct` key will have no effect on JPEG images which are already DCT com-

pressed. If you would like to recompress/degrade JPEG images, use the key `repackjpeg` together with `dct` and `dctquality`.

3.16.3 Imagemasks

The `imagemask` keyword for the `\includegraphics` command applies to monochrome bitmapped images only. By default, images are not subjected to text color changes. With `imagemask` on, the images are treated as imagemasks: they assume the text color.

`imagemask` has a second application besides the ability to color images: when this keyword is specified, the image becomes *transparent*; this allows to overlap text and images without the risk of the image erasing the text.

3.17 PostScript file inclusion

Both the PDF and PostScript backend fully support inclusion of (encapsulated) PostScript files.

In PDF mode, PostScript→PDF translation is performed by the `GεX` converter.

With the PostScript backend, PostScript files can be included without using `GεX`: they are simply passed unchanged to the `.ps` output. *With* `GεX`, however, the PS code is reinterpreted and optimized. One of the benefits of this is the combining of the fonts and other resources that are often repeated in included PostScript files.

You can use the L^AT_EX packages `graphicx`, `graphics` or `psfig` to include PostScript images. Of the three choices, `graphicx` is the one we recommend; see the document `texmf/doc/latex/graphics/grfguide.pdf` for the documentation.

3.17.1 Page selection

Using `graphicx.sty`, a single page of a multi-page PostScript file can be selected for inclusion. With V_TE_X, the `\includegraphics` command provides an additional key `page=` to specify the *absolute* page number within the file.

3.18 Page Labels



The information in section applies to the PDF mode only.

Acrobat 4+ supports the new *Page Label* feature. Page Labels affect the way pages are numbered in Acrobat Menus and Controls; they have no effect on the “printed page” portion of the document.

Page labels are useful if you are creating online documentation which uses a page sequence other than the default 1..N; for example, if some pages are numbered with the roman numerals, and some with the italics. To have the Acrobat number the pages the same way in its menus, use the `\special{!pdfpagelabels...}` command:

```
\special{!pdfpagelabels
  /Nums [ 0 << /S /r >> % From page 0, roman (/r)
         3 << /S /D >> % From page 3, arabic digits (/D)
        ]
}
```

Notice that unlike most other `\special`'s in $\text{V}\text{T}_{\text{E}}\text{X}$, here the syntax is taken from the PDF 1.3 specifications verbatim. The five numbering styles supported by PDF 1.3 are

- ▷ `/r` lower case roman numbers
- ▷ `/R` upper case roman numbers
- ▷ `/d` digits
- ▷ `/a` lower case latin letters
- ▷ `/A` upper case latin letters

Two additional optional keys that can be used within the page label specifications are the prefix (`/P`, to be followed by a string), and starting value for the range (`/St`, to be followed by an integer, default 1). The prefix specifies additional text to be appended to the left of the labels; the numbering starts with the starting value. For example:

```
\special{!pdfpagelabels
  /Nums [ 0 << /S /D /P (Intro-) >>
        5 << /S /D /St 6>>
        200 << /S /D /P (App-) /St 201 >>
  ]
}
```

Here, all the numbering is done with digits, but the Introduction and the Appendix pages are marked so.

The `\special{!pdfpagelabels...}` command may appear anywhere in the document; if more than one is specified, the last specification is used. $\text{V}\text{T}_{\text{E}}\text{X}$ does not check the validity of the arguments.

This `special` has no effect on previewing under Acrobat 3 or GhostScript.

If you are using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}2_{\text{e}}$, the task of constructing the Page Labels specifications can be given to `Hyperref`; with other formats, you need to build the specification manually.

3.19 Unsupported $\text{V}\text{T}_{\text{E}}\text{X}$ syntax

Due to the limitations of the `.pdf` format, some of the $\text{V}\text{T}_{\text{E}}\text{X}$ language extensions will not work:

- ▷ Gray rules are not supported (but colored rules are, so this is not really a problem).
- ▷ Font effects, except for `slant`, `aspect`, and the simplest form of `outline` are not supported on Type 1 fonts. All font effects are supported on `.if4` fonts.

4. Implementation of the \LaTeX packages `color` and `graphicx`

4.1 Color support

The `color` package allows to specify the color in either the RGB space or the CMYK space. On the low level this corresponds to \TeX color `\special's`

- ▷ `\special{c"rrggbb}`
- ▷ `\special{c:ccmmyykk}`

All letter pairs in the syntax notation above correspond to two hex digits that define a hex number in the range of 0 through 255. Dividing this number over 255 gives the appropriate color component. For example, `\special{c"FF0000}` defines the red color. This color notation is patterned over the syntax used in HTML.

The macro that converts the `color.sty` notation of the color components to \TeX 's notation has been contributed by David Carlisle.

4.2 Text-box scaling and Rotation

These features are implemented via the new `\special{r...}` command. This command modifies the PDF Current Transformation Matrix (CTM) in the PDF backend or its `.dvi` analog in the DVI backend. There are two forms of the command:

- ▷ Save and modify the CTM:

`\special{r(c11,c12,c21,c22,0,0)}`

This will push the original CTM matrix to the stack and multiply it with the

$$\begin{pmatrix} c11 & c12 \\ c21 & c22 \end{pmatrix}$$

- ▷ Restore the CTM:

`\special{r})}`

This will pop the previous CTM from the stack.

Notice that both the \TeX drivers and the `.pdf` backend assume that CTM transformations are properly nested. The Graphics state stack overflows and underflows are treated as fatal errors.

Examples:

Scaling To scale a box by the factor of 2, use

`\special{r(2,0,0,2,0,0)} ... box ... \special{r})}`

Non-uniform scaling To scale the y -coordinate only by the factor of three, use

`\special{r(1,0,0,3,0,0)} ... box ... \special{r})}`

Flip To flip a box along the x -axis, use

`\special{r(-1,0,0,1,0,0)} ... box ... \special{r})}`

Rotation To rotate a box 30° , use

```
\special{r(0.866,0.5,-0.5,0.866,0,0} ... box ... \special{r)}
```

Notice here that $0.866 \approx \cos(30/180 \times \pi)$ and $0.5 = \sin(30/180 \times \pi)$.

The responsibility of allocating sufficient space for the transformed box lies with the macro package designer. This allocation is automatically handled by the `Graphicx \scalebox{..}` and `\rotatebox{..}` commands.

The current implementation will transform text and rules but not graphics images within the text box.

5. Language extensions

This chapter very briefly summarizes a few important syntax extensions in $\text{V}\text{T}_{\text{E}}\text{X}$. A comprehensive description can be found in the $\text{V}\text{T}_{\text{E}}\text{X}$ Language Guide, which is currently provided with the commercial Windows version only.

- ▷ `\font` command: additional tags like `slant`, `outline`, `charmap`, and others allow to alter font shape and encoding dynamically.
- ▷ `\aliasfont` primitive allows to define an alias to the another font (the latter being possibly modified using the extended `\font` syntax). See the file `texmf/vtex/config/preload.cfg`, which uses `\aliasfont` and the extended `\font` syntax extensively.
- ▷ `\parshape` primitive extension is introduced to allow paragraphs with “holes”. See the document `parshape.tex` for an example.
- ▷ The `\exec` and `\command` primitives provide interface to launching external programs:

The `\exec` command takes two arguments; the first is the name (and the path, if required) of an application to invoke; the second is the argument string.

`\command` takes a single argument which is passed to the `command.com` or `cmd.exe` shell. Since commands like `dir` or `pause` are not programs under Windows and OS/2 but are handled by the command processor, they must be invoked via `\command`. On Linux, however, `ls` is a program, so it can be used in `\exec{}`.

- ▷ Gray rules. Rules can be filled with gray color, or with the user-defined pattern. The `\special{M...}` is used to define a gray pattern. For an example, see the `grule.tex` file.
- ▷ Almost unlimited number of fonts and registers: The number of fonts and TeX registers (`\count`'s, `\dimen`'s, `\box`'es) in $\text{V}\text{T}_{\text{E}}\text{X}$ is limited by 65536 (256 in standard TeX). Thus, it is entirely legal to write `\count1000=10`. These registers are allocated “on use”. You can trace register reallocation if you set `\tracingallocs` to a positive value. Note, however, that $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ register allocation does not use these extra registers, so under $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ you may still run out of registers, unless you use your own allocation scheme.

6. Support

The NTG hosts a mailing list that can be used both to ask questions and to report bugs in the software and/or installation. New versions of V_TE_X/Free will also be announced there. The mailing list is run by *Majordomo*, so to subscribe send a message to `majordomo@ntg.nl` with in the body

`subscribe ntg-vtex`

The mailing list itself can be reached as `ntg-vtex@ntg.nl`.

7. License Terms

MicroPress and V_TE_X are trademarks of MicroPress, Inc.

The MicroPress' components of the V_TE_X/Free distribution are

Copyright © 1998-2001 by MicroPress Inc.

The MicroPress' components of the V_TE_X/Free distribution are free for personal use, subject to the following restriction:

You must obtain a commercial license from MicroPress if you intend to redistribute V_TE_X/Free on a permanent media (CD).

The V_TE_X/Free distribution includes components that are covered by different licenses, as follows:

The Adobe Utopia fonts, see `texmf/fonts/type1/adobe/utopia`, are subject to the following conditions of use and distribution:

Permission to use, reproduce, display and distribute the listed typefaces is hereby granted, provided that the Adobe Copyright notice appears in all whole and partial copies of the software and that the following trademark symbol and attribution appear in all unmodified copies of the software:

Copyright © 1989 Adobe Systems Incorporated.

Utopia[®] is a registered trademark of Adobe Systems Incorporated.

The Bitstream Charter fonts are subject to the following conditions of use and distribution:

© Copyright 1989-1992, Bitstream Inc., Cambridge, MA.

You are hereby granted permission under all Bitstream propriety rights to use, copy, modify, sublicense, sell, and redistribute the 4 Bitstream Charter[®] Type 1 outline fonts for any purpose and without restriction; provided, that this notice is left intact on all copies of such fonts and that Bitstream's trademark is acknowledged as shown below on all unmodified copies of the 4 Charter Type 1 fonts:

Bitstream Charter is a registered trademark of Bitstream Inc.

Notice also the license conditions in the files of the L^AT_EX system!