VTEX/Free Manual

M. Vulis
T. Hoekwater
W. Schmidt

***Preliminary** edition for VTEX 7.00*
2000-05-11

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

1. What is VTEX/Free?

VTEX/Free is a *partial* port of the VTEX/Windows TEX 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 VTEX/Win, only three are supported (**PDF**, **PS** (PostScript) and **DVI**, but not **HTML**); the DVI mode is essentially useless, since the main advantages of VTEX's DVI mode under Windows rely on the VTEX DVI drivers which have not been ported.

Thus, for all practical purposes, VTEX/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`,...) are almost identical.

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

VTEX/Free *does* include the full PostScript support (GEX) of the Windows version. This includes both the **EPS** inclusion and inline PostScript, including support for PStricks, PSfrag, XYpic and GEXX. GEX can be used in either the PDF or the PS mode; in PDF mode GEX acts as an integrated PDF→PS distiller; in the PS mode, it acts as a PS→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 VTEX Documentation

2.1 Invocation and usage

The name of the VTEX 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 so 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 TEX implementations.

VTEX 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.

☞ Notice, that the distribution comes with two shell scripts named `vlatex` and `vlatexp` that provide all necessary options to run VTEX in PDF or PostScript mode with the LATEX format. You may want to modify these scripts according to your personal needs, e.g,, specify fast Static path caching (see section 2.3).

VTEX 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 VTEX extensions are disabled. DVI files created in compatibility mode are compatible with non-MicroPress drivers. *Not useful for PDF and PS modes.*

`-D...` DVI, PDF or PS (output) directory.

`-i` IniTEX mode.

`-2` Enable bi-directional typesetting (TeX-XeT).

`-n#` where `#` is the initial TEX running mode:

0	batch
1	nonstop
2	scroll
3	errorstop (default)

`-p` Path caching switches, see section 2.3

`-q` Quick run. This switch disables the output. While you need to compile LATEX 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 LATEX compilation with `-q` used on the first pass, and 33% overall time saving on a 3-pass LaTeX 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.

`-ox` Enable GeX.

`-=` Specify alternative configuration file. This could be useful if VTeX is installed in a multiuser environment.

`-83` Disable use of “long” file names with VTeX/2, e.g., for running on a FAT drive.

See section 3.2 for additional options related with PDF or PS generation only!

2.2 General layout and configuration files

Unlike some other versions of TeX, VTeX does not rely on the environment variables, but rather on a configuration file, `vtex.ini` (`vtexlnx.rc` under Linux). This file is a typical Windows-style `.ini` file; it is divided onto several sections, each defining its variables. The majority of sections are not applicable to the VTeX/Free distributions and have been omitted from this description. What remains is described in the next section.

The only other configuration file is `aliasing.pst` (`type1.rc` under Linux). This file functions as a font mapping file for the PDF mode compiler.

2.2.1 vtex.ini – vtexlnx.rc

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

- `PGMDIR`: VTeX binaries, default `/vtex/bin/`
- `INCDIR`: `.tex` 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/`

Multiple directories should be separated with semicolons, and the trailing path separator should 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 VTeX is format-dependant. Assuming that you compile with a format named “`latexhv`” (HV-math based LaTeX2E), VTeX 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 LaTeX2E and LaTeX2.09 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 VTEX-based documents search only /vtex/src, while LATEX2E 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 (this is completely independant of the -: switch).

Notice that the [FINCLUDE] directory match works by prefix comparison. This means that if your format is actually latexhv, VTEX 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.

See below (3.5) for an explanation of the A4FIX variable in the [THIRDPARTY] section!

2.2.2 aliasing.pst – type1.rc

This file lists all the available Type 1 fonts and the font file locations. The file is ASCII and can be manually edited if needed.

The aliasing.pst consists of two sections, separated by an empty line. The first section lists all the directories that include referenced .pfb and .pfm files. 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. For example (OS/2),

```
%1 = c:\acrobat3\reados2\fonts\pfm\
%2 = e:\texmf\fonts\type1\micropress\cm\
%3 = e:\texmf\fonts\type1\micropress\ams\
```

defines three directories that can be referenced below.

☞ In VTEX/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 one dummy directory.

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

```
@<Full-name> = <File-name> %<access><pfb-dir>,<pfm-dir>[,<other-flags>]
```

where

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

File-name: The .tfm file name for the font as referenced by TEX, which must be the same as the name of the related .pfb file. For example, tir_____ may be the file name for Times-Roman.

access: Usually, nothing. Fonts that are built-in in the Acrobat Reader should have a ! mark to prevent their (unnecessary) downloading into the document.

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

pfb-dir: The number of the directory that contains the `.pfb` file for this font.

pfm-dir: The number of the directory that contains the `.pfm` file for this font.

other-flags: Used for reencoding.

Example:

```
@cmr10 = cmr10 %2,1
@Times-Bold = tib_____ %!4,1
@Times-BoldItalic = tibi____ %!4,1
@Times-Italic = tii_____ %!4,1
@Times-Roman = tir_____ %!4,1
```

Notice that the Times fonts are built-in and so marked.

2.2.3 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 default layout of the CM fonts which are supplied with VTEX/Free is suitable for TEX and no re-encoding need to be applied to these fonts.

VTEX allows you to configure Type 1 fonts to follow a desired character layout with full ligature support. VTEX comes with several encodings, including

- TEX T1-encoding (the suggested default for LATEX2e)
- VTEX Standard encoding (the layout that matches the other fonts supplied in the VTEX implementation.)

Additional encodings can be easily made by the user.

The available encodings are stored in files with extension `.enc` within the ENC subdirectory. See the next section for the details on the format of these files.

Since there may be more than one TEX-font based on the a single Type 1 font and multiple encodings, they have to be distinguished by different font names and TEX metric `.tfm` files. Notice that the font names can be chosen arbitrarily in this case. The format of the `aliasing.pst` entries for reencoded fonts contains two additional entries to support this:

- `r="<encoding file name">`
- `b="<base Type 1 font">`, i.e., the actual name of the `.pfb` file.

This information appears at the end of the `aliasing.pst` entries. 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 `aliasing.pst` declare two instances of the Times-Roman font: default and reencoded.

```
@Times-Roman = tir_____ %!4,1
@Times-Roman-8r = ptmr8r %!4,1,r="8r.enc",b="tir_____"
```

Notice that if the reencoding map is not specified, no reencoding is assumed. Thus you can assign a TFM name, which differs from the name of the `.pfb` file, e.g.:

```
@Symbol = psyr %4,1,b="sy______"
```

2.2.4 Encoding files

The available encodings are stored in files with extension `.enc` within the ENC 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. Notice 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.

2.3 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 VTEX comes in two models:

- Dynamic path caching: at the beginning of each compilation, VTEX builds a full list of the contents of the include directories in memory; when a file needs to be opened, VTEX 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 extention `.ffs` and are stored in the format directory of VTEX, 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; `vtex.ffs` is the name of the path cache file for the VTEX/Plain format.]

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 descibe 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).

- Assume that a document contains the command `\input test`.

- 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 subdirectories.) Assume that `C:\MYFILES\+` contains a subdirectory `OLD`.
- Make a one-line file, `test.tex`, containing

  ```
  \message{Old file!}
  ```

  and place it into `C:\MYFILES\OLD`.
- Build the static path cache for the Plain format.
- Make a one-line file, again called `test.tex`, but containing

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

  and place it into `C:\MYFILES`.
- 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).

☞ Notice, 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. Notice 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 `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.

☞ Use of the `-pr0` command 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. We recommend using this switch in batch production environments only.

3. PostScript and PDF generation

3.1 General

The VTEX document compiler supports generating `.pdf` files in addition to the traditional `.dvi` files.

The PDF mode has been designed to be totally transparent to use. No changes are required to your document source. All additional features supported by the Portable document format (like hyperlinks and outline entries) are supported via `\special` commands, rather than syntax changes.

The PDF backend also supports *most* of the VTEX extensions. This chapter explains most of the PDF 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 (preliminary release in VTEX 6.5), see forms.pdf.

Notice that most features shown in the latter document require Acrobat 4 !

3.2 Command-line options

To create a `.pdf` file, you should supply the `-$p` switch to the VTEX 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 `$` with the `\` escape. Alternatively, you can use `@` instead of `$`. Further, 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 13” Type 1 font into GEX unless they are actually required, thus speeding up GEX. (Notice that this is to become the default behaviour with the next version of VTEX.)
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.
w	`w=<dimen>` specifies the document MediaBox width (see below).

The `<output spec>` may include these letters:

a	ASCII85 output
h	Hex output
b	Binary output (default)
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. Notice 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. Notice 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=f,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 Media size

The default MediaBox generated by VTEX is based on the settings of `\hsize` and `\vsize`. If your format is Plain, VTEX, or AmSTEX, you usually will not have to specify the MediaBox sizes manually, using the `w` and `h` command line options.

With LATEX, the MediaBox sizes are set to `\paperwidth` and `\paperheight`, provided that the LATEX format has been created from the file `texmf/vtex/config/latex.tex` which is part of the VTEX/Free distribution. The command line options `w` and `h` should specify a fallback MediaBox, which is to be used when a LATEX class does not define the paper size. (`\hsize` and `vsize` are not correct with LATEX.) For example, the `vlatex` 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 TEX 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. Notice, however, that this will make the document source incompatible with other TEX 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 LaTeX only),
3. the command line options `w` and `h`,
4. `\hsize` and `\vsize`.

3.4 Font usage

VTeX-built `.pdf` files currently support only these two formats:

- `.pfb` (Type 1 PostScript binary files)
- `.if4` (VTeX native fonts, not supplied with VTeX/Free)

Other formats (`.ttf` and bitmapped fonts) are not supported in this version.

To avoid generating bad `.pdf` files, VTeX checks for the presense of an acceptible font file whenever the font is referenced in the document. Further, before starting the compilation of a document, VTeX 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, VTeX will generate an error and substitute the `nullfont`. Notice that this behavior may produce documents that are formatted differently as `.dvi` and as `.pdf`.

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

- If the font is listed in the `aliasing.pst` configuration file, it will be used. Otherwise,
- if the font file with extension `.if4` is present in the `VTEX\IF4` subdirectory, it will be used.

Notice that the Type 1 format is substantially better for the `.pdf` documents than the `.if4`. This is because Type 1 fonts are saved within the `.pdf` files as outlines; `.if4` fonts, while internally scalable, are converted to bitmapped fonts when saved into the `.pdf` files (and lose their scalability as a result with serious degradation in quality). Use `.if4` fonts only if there is no way to obtain an equivalent Type 1 font!

☞ In PDF mode the `.log` file lists the fonts used; if the log lists IF4 fonts, you may want to reexamine your configuration.

3.5 Acrobat 4 issues

3.5.1 PCL printing

☞ This section applies to PDF mode only.

Acrobat 4, while a genuine improvement over Acrobat 3, introduces some severe bugs. Of primary importance is the mishandling of fonts, particularly the TeX math extension fonts (like `cmex10` or `mtex10`): 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. Thus, if you intend to distribute your `.pdf`'s, you should assume that some of your readers will have version 4 with all the dire consequences.

This problem affects not just VTEX, 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, VTEX (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 (2.2.1). Valid settings are:

`A4FIX=0` No correction for the Acrobat 4 printing problem. This is identical to the PdfTeX 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*. This was the default behaviour since VTEX 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. Notice 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`. This file must reside in the include path, see 2.2.1. The current list of fonts is:

```
cmex10 cmex7 cmex8 cmex9 euex10 ifex10 mhex10 mhexb10
mtex10 mtexb10 mvex10 mvex7 mvex8 mvex9
```

3.5.2 Substituting Times and Helvetica

In contrast to Acrobat 3, the Acrobat 4 distribution no longer includes the Type 1 font families Adobe Times and Adobe Helvetica. Since these are *required* for running VTEX, they need either to be purchased or taken from an Acrobat 3 package. Besides, VTEX/Free is also providing a workaround:

In contrast to what has been said above (section 2.2.2), the `aliasing.pst` or `type1.rc` file may also list the URW clones of Times and Helvetica under their Adobe names:

```
@Helvetica =             n019003l %!5,1
@Helvetica-Bold =        n019004l %!5,1
@Helvetica-Oblique =     n019023l %!5,1
...
```

And, of course, all reencoded entries to these fonts must also refer to `.pfb` files of URW clones. Notice, however, that this holds for Times and Helvetica (not Helvetica-Narrow) only.

3.6 Links

One of the advantages of `.pdf` files is the ability to produce hyperlinks. On the low level, this is accompished by VTEX `\special` commands:

!aref... Start a hyperlink

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

!aname Define a target.

Both `\special{!aref...` and `\special{!name...` 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.

Notice 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. Notice also VTEX 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 TEX 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 TEX 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 VTEX's PS mode makes sense only if you intend to post-process the `.ps` output and then feed it to a PS→PDF converter.

3.7 Outline

You can supplement the `.pdf` file with an outline. The low-level interface for this is the VTEX `\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 TEX, you should use only “normal” characters within it. For example, putting `\TeX` within the outline entry text is a very bad idea.

☞ VTEX 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 LATEX 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 `'`). 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 LaTeX 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.txt`.

3.8 Transitions

☞ The information in section applies to the PDF mode only.

Starting with version 6.4, VTEX 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` subdirectory. This file shows all available transitions.

3.9 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 bacground 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.

3.10 Annotations

VTEX allows to place text annotations into the PDF file. This could be quite useful if you are working on a document with a collegue: 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]>` t[illegible]e the 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 3.7.

3.11 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, VTEX 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.

3.12 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, VTEX now adds two more `\special{...}` commands:

- `\special{G(}`
- `\special{G)}`

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

For TEX'nically minded: in VTEX, GEX's kernel is active even if GEX interpreter has not been enabled (i.e. no `-ox` *switch given). It handles commands which deal with the colors and page transformations; thus some VTEX specials merely call GEX, 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 GEX initialization.*

3.13 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 VTEX 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.

3.14 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.

3.15 Bitmapped graphics inclusion

The Linux port of the compiler features all of the bitmapped graphics filters also found in the windows version: **PCX**, **TARGA**, **BMP**, **GIF**, **JPEG**, **PNG** and **TIFF** images can all be included directly.

For the time being, the OS/2 port of the compiler features the bitmapped graphics filter for **BMP** (Windows BMP !), **GIF** and **JPEG** format only.

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 LATEX macro package `graphicx`; see the document `texmf/doc/latex/graphics/grfguide.pdf` for a general description of the package. This section explains how VTEX processes bitmapped files in the PDF and PS modes and the additional `\includegraphics` keys that can be of use in fine-tuning the performance.

3.15.1 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 `\includgraphics` command. Upon rendering to an output device, the original bitmap will be scaled to these dimensions.
- Use the keyword `atres=`⟨*resolution*⟩. VTEX 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 resolutiom of the output device is different, the bitmap is scaled appropriately.

3.15.2 Compression and fine-tuning

Few ground rules to start with:

1. VTEX supplies two distinct techniques for loading the bitmapped files: direct processing and loading into the integrated PostScript interpreter, GEX. 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 GEX: for example, you can post-process the images with the GEX imaging plugins (see the GEX documentation).

   To load a bitmapped image for GEX inclusion processing, use the `viagex` key for the `\includegraphics` command.

   ☞ Notice, however, that the default shell script for running VTEX in PostScript mode does *not* turn on GEX. You may change this by adding the `-ox` option, if you want to use GEX's facilities in PS mode.

2. The default processing logic for the direct picture inclusion is to 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 (GEX mode only).

4. In order to force minimal size of directly included images, you can force VTEX 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*. Notice 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 compressed. If you would like to recompress/degrade JPEG images, use the key `repackjpeg` together with `dct` and `dctquality`.

3.15.3 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.

3.16 EPS file inclusion

Both the PDF and PostScript backend fully support EPS file inclusion.

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

With the PostScript backend, `.eps` files can be included without using GEX: they are simply passed unchanged to the `.ps` output. *With* GEX, however, the EPS 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 `.eps` files.

You can use the LaTeX packages `graphicx`, `vpsfig` or `epsf` to include `.eps` images. Of the three choices, `graphicx` is the one we recommend; see the document `texmf/doc/latex/graphics/grfguide.pdf` for a description of the package.

3.17 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 VTEX, 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 in integer, default 1). The prefix specifies additonal 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 **Intro-**duction and the **App-**endix 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. VTEX does not check the validity of the arguments.

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

If you are using LATEX2e, the task of constructing the Page Labels specifications can be given to `Hyperref`; with other formats, you need to build the specification manually.

3.18 Unsupported VTEX syntax

Due to the limitations of the `.pdf` format, some of the VTEX language extensions will not work. Specifically

- 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 VTeX 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 VTeX'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 VTeX 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 the important syntax extensions in VTEX. A detailed description can be found in the VTEX 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
- `\charmap` primitive allows to define a new font encoding table
- `\aliasfont` primitive allows to define an alias to the another font (the latter being possibly modified using the extended `\font` syntax)
- `\parshape` primitive extension is introduced to allow paragraphs with "holes"
- `\exec` and `\command` primitives provide interface to launching external programms
- 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 VTEX 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. Notice, however, that LATEX register allocation does not use these extra registers, so under LATEX 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 VTEX/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