## The package ps4pdf: from PostScript to PDF

Rolf Niepraschk and Herbert Voß

**Abstract**

The only graphic object which TEX can handle internally is the `picture` environment, which is on the one hand very easy to use, but on the other hand very restrictive. All other graphical material must be encapsulated in `\special` commands and later extracted by the DVI processor, for example, `dvips` into PostScript code. Packages like `pstricks` (and its extensions `pst-xxxx`) and `psfrag` can create such `\special` commands. Unfortunately, `pdflatex` cannot work when one of these packages is part of the document file. The new package `ps4pdf` makes it possible to collect all PostScript-related parts and convert them to PDF in a single run.

### 1 Introduction

PDF output can be created in several different ways:

- traditional: `dvi`→`ps`→`pdf` using the commands `latex` to create the DVI file, `dvips` for the `ps` file and `ps2pdf` for the PDF file.
- using `dvipdfm` to skip the `ps` step—but have a look at the manual page of `dvipdfm` for some restrictions.
- using pdfLATEX to skip the `dvi` step and generate PDF directly—but this has the problem stated in the abstract.
- using VTEX as an alternative to pdfLATEX—but this is available without charge only for Linux and OS/2 [2].
- using the package `pdftricks` [5, 6]—but in some PostScript environments, a bounding box is difficult to determine.
- using the package introduced here, `ps4pdf` [4].

This new package `ps4pdf` works very differently from `pdftricks`. It uses the package `preview`, which is part of the `latex-preview` [1, 3] bundle, available at any CTAN server. `preview` extracts all 'marked' parts of a complete LATEX document to a DVI file, in which each such part is saved on a separate page. This makes it easy to convert this DVI file into PDF format and then include these parts in a last pdfLATEX run.

### 2 Package options

Table 1 shows the available package options. Specifying `inactive` causes all the `ps4pdf` macros disables everything except the trimming functionality, so that `latex` runs in the usual way. This makes direct PostScript output possible, so that the other methods listed above can be used.

Table 1: `ps4pdf` package options

| name | meaning |
|---|---|
| `active` | enables the `ps4pdf` macros (default) |
| `inactive` | disables the `ps4pdf` macros, making direct PostScript output possible; this is the default if VTEX is detected |
| `trim` | modify the internal bounding box (similar to the `trim` option from `\includegraphics`) |
| `draft` | suppresses the content of the `\PSforPDF` macros (do not influence the content of the graphics container) |
| `final` | shows the content of the `\PSforPDF` macros (default) |

### 3 Usage

#### 3.1 Usage in preamble

Assume that we have the small LATEX file shown in listing 1; the output is shown in figure 1. To use the `ps4pdf` package, we must pass **all** PostScript-related parts through the `\PSforPDF` macro, beginning with the preamble. And of course we must use the `ps4pdf` package itself.

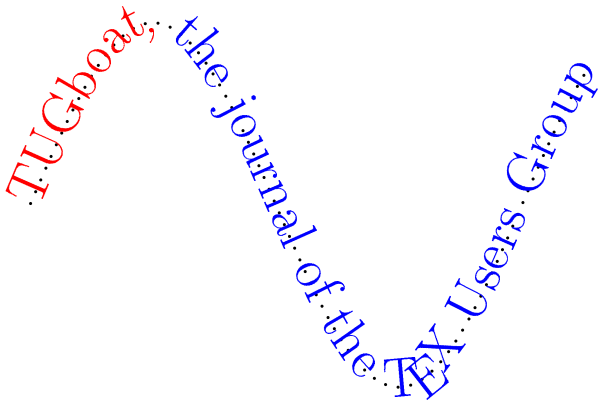


**Figure 1**: Output of listing 1

`ps4pdf` must know every package that the PostScript images depend upon, otherwise it cannot create the images and convert them to PDF. For our example here, listing 2 shows how this is done.

Listing 1: Demonstration of a `pstricks` object

Table 2: The command sequence from PostScript to PDF

| | |
|---|---|
| ▷ **graphics container creation:** | |
| `latex file.tex` | LaTeX run |
| `dvips -Ppdf -o file-pics.ps file.dvi` | `dvips` run to convert the DVI file to PostScript |
| `ps2pdf file-pics.ps file-pics.pdf` | `ps2pdf` run to convert the PostScript file to PDF |
| ▷ **document creation:** | |
| `pdflatex file.tex` | first pdfLaTeX run |
| `bibtex file` | BibTeX run |
| [ ... ] | any other additional runs (for example: glossary) |
| `pdflatex file.tex` | second (and last) pdfLaTeX run |

```
\documentclass{article}
\usepackage{pst-plot}
\usepackage{pst-text}
\begin{document}
  \psset{unit=1cm}
  \begin{pspicture}(-0.25,-2.25)(6.25,2.25)
    \pstextpath[linestyle=none]%
       {\psplot[linewidth=1pt,%
           linestyle=dotted,%
           plotpoints=300,%
           xunit=0.015,%
           yunit=2]{0}{400}{x sin}}
       {\LARGE TUGboat, the journal
          of the \TeX{} Users Group}
  \end{pspicture}%
\end{document}
```

Listing 2: Using \PSforPDF in the preamble

```
\documentclass{article}
\usepackage{ps4pdf}
\PSforPDF{%--- BEGIN PSforPDF
  \usepackage{pst-plot}%
  \usepackage{pst-text}%
}%--- END PSforPDF
```

### 3.2 Usage in document body

For the user, there is no difference between using ps4pdf in the preamble or in the text part of the document: any PostScript-related material must be passed to the \PSforPDF macro. Only internally are these separate parts of the document handled in different ways. Thus, listing 1 is changed to what we have in listing 3.

Listing 3: Using \PSforPDF in the body

```
\begin{document}
\PSforPDF{%--- BEGIN PSforPDF
  \psset{unit=1cm}
  \begin{pspicture}(-0.25,-2.25)(6.25,2.25)
     [ ... ]
  \end{pspicture}%
}%--- END PSforPDF
\end{document}
```

## 4 Implementation

ps4pdf is part of the process shown in table 2, which can also be encapsulated as a shell script (listing 4).

Listing 4: Shell script implementing table 2

```
#!/bin/sh
# build a pdf file with PostScript code
# Herbert Voss 2003-03-10
# usage: ps4pdf.sh file (without suffix tex)
latex $1.tex
dvips -Ppdf -o $1-pics.ps $1.dvi
ps2pdf $1-pics.ps $1-pics.pdf
pdflatex $1.tex
bibtex $1
pdflatex $1.tex
```

- In the first LaTeX run, `preview-latex` extracts all objects which are included as a argument to \PSforPDF, and saves them into `<file>.dvi`. Each object is on its own page.
- This object file is then converted into a PostScript file with `dvips`. The `-Ppdf` option tells `dvips` to load the config file for PDF-related output. `dvips` creates the new file `<file>-pics.ps`.
- This PostScript file `<file>-pics.ps` is then converted into the corresponding PDF file with `ps2pdf` (a front end to Ghostscript).

- At this point the important work of `ps4pdf` is done, and the usual `pdflatex` runs can be done, as well as additional runs for BiBTEX or other post-processors.
- If all worked well, then the final PDF file includes all PostScript-related code as PDF images!

The meaning of the macro `\PSforPDF` changes for the pdfLATEX runs. It now becomes:

```
\includegraphics[page=<n>]%
     {<file>-pics.pdf}
```

which inserts the $n$th page of the object file. An internal counter is used to get the right object at the right place. This implies that the whole command sequence in table 2 has to be repeated if the sequence of the objects changes.

## 5 Saving the images

Saving all graphical objects from the PDF graphic container as single files is very easy and can be done with the small script shown in listing 5. After running this script with `<file>-pics.pdf` as parameter, the images are saved as `picture<n>.eps`. This may be useful for other purposes.

Listing 5: Script to convert PDF images to PostScript

```
1 #!/bin/sh
2 File=$1
3 n=‘pdfinfo $File | awk ’($1 ~ /Pages:/) {
      print $2}’‘
4 for i in ‘seq $n‘ ; do
5   pdftops -f $i -l $i -eps $File picture$i.
        eps
6 done
```

## References


[1] David Kastrup. *preview-latex*. CTAN:/support/preview-latex/, 2003.

[2] Micropress. *VTEX/Lnx*. http://www.micropress-inc.com/linux/, 2003.

[3] Rolf Niepraschk. Anwendungen des LATEX-pakets preview. *Die TEXnische Komödie*, 1/2003:60–65, February 2003.

[4] Rolf Niepraschk. *ps4pdf*. CTAN:/macros/latex/contrib/ps4pdf/, 2003.

[5] Chambert-Loir Radhakrishnan, Rajagopal. *pdftricks*. CTAN:/macros/latex/contrib/supported/pdftricks/pdftricks.sty, 2002.

[6] Herbert Voß. *PSTricks Support for pdf*. http://www.pstricks.de/pdf/pdftricks.phtml, 2002.



◇ Rolf Niepraschk
Persiusstr. 12
10245 Berlin GERMANY
niepraschk@ptb.de

◇ Herbert Voß
Wasgenstr. 21
14129 Berlin GERMANY
voss@perce.de
http://www.perce.de