

The pst-pdf package*

Rolf Niepraschk[†] Hubert Gäßlein

2019/11/15

1 Introduction

The package `pst-pdf` simplifies the use of graphics from PSTricks and other PostScript code in PDF documents. As in building a bibliography with `BIBTEX` additional external programmes are being invoked. In this case they are used to create a PDF file (`\PDFcontainer`) that will contain all this graphics material. In the final document this contents will be inserted instead of the original PostScript code.

2 Usage

2.1 Package options

active Activates the extraction mode (DVI output). An explicit declaration usually is not necessary (default in `LATEX` mode).

inactive No special actions; only the packages `pstricks` and `graphicx` are loaded (default in `VTEX`). Can be used to just convert the document with `LATEX` into a DVI file while avoiding the automatic extraction mode.

pstricks The package `pstricks` is loaded (default).

nopstricks The package `pstricks` does not get loaded. Once it is detected that `pstricks` was loaded however in some other way, the `pspicture` environment is treated as if the option “`pstricks`” was given.

draft From the `\PDFcontainer` file included graphics is displayed as frame in `pdfLATEX` mode.

final From the `\PDFcontainer` file included graphics is correctly displayed in `pdfLATEX` mode (default).

tightpage The graphics’ dimensions in the `\PDFcontainer` file match exactly those of the corresponding `TEX` boxes (default).

notightpage The dimensions of the `TEX` box corresponding to its graphics is not always correct, since a PostScript statement can draw outside its box. The option “`notightpage`” makes the graphics in the `\PDFcontainer` file to be at

*This document corresponds to `pst-pdf` v1.2e, dated 2019/11/15. Thanks to Peter Dybala for the translation.

[†]`Rolf.Niepraschk@gmx.de`

least the size of the whole page. To be able to make use of the graphics' in a later pdfL^AT_EX run, the \PDFcontainer file needs to be finished in a way that each graphics gets reduced in size to its visible part. For this an external programme like pdfcrop¹ can be useful. Its use can save declaring the option “trim” (see also section ??).

displaymath In PDF mode the mathematical environments displaymath, eqnarray, and \$\$ get also extracted and included as graphics. This way additional PSTricks extensions can easily be added to the contents of these environments. (Question: how do AMSL^AT_EX environments behave?)

⟨*other*⟩ All other options are passed to pstricks package.

2.2 Program calls

The following table shows the course necessary to create a PDF document containing PostScript graphics². As comparison the analogous course for a bibliography is shown.

PostScript graphics	bibliography
pdf _l atex document.tex	pdf _l atex document.tex
<i>auxiliary calls</i>	
latex document.tex	
dvips -o document-pics.ps document.dvi	
ps2pdf document-pics.ps	bibtex document.aux
pdf _l atex document.tex	pdf _l atex document.tex

While creating the output only code from inside a pspicture or postscript environment is considered. PostScript graphics files, which are passed as parameter of an \includegraphics statement, too are included into the \PDFcontainer file. This file's name is by default ⟨\jobname⟩-pics.pdf. It can be changed by re-defining the macro \PDFcontainer.

2.3 User commands

pspicture \begin{pspicture}[⟨keys⟩] (⟨x0,x1⟩) (⟨y0,y1⟩) ... \end{pspicture}
The pspicture environment is not available when the option “nopstricks” was given. It is to be used the same way as if in PSTricks. In pdfL^AT_EX mode this environment's contents is only displayed when the \PDFcontainer file was created before.

postscript \begin{postscript}[⟨keys⟩] ... \end{postscript}
The postscript environment can contain any code except floats. In pdfL^AT_EX mode its contents is take too off the \PDFcontainer file. Other as in the pspicture environment the necessary space is not always preserved when the \PDFcontainer file does not exist yet.

\includegraphics \includegraphics[⟨keys⟩]{⟨filename⟩}

¹CTAN: support/pdfcrop/

²The T_EX distribution “teT_EX” contains a UNIX shell script ps4pdf which executes all the necessary steps. See: CTAN: macros/latex/contrib/ps4pdf/

To be used as in `graphics/graphicx` defined. In pdfL^AT_EX mode it is now additionally feasible to pass the name of an EPS file. Its visible contents too is taken from the `\PDFcontainer` file.

<code>\includegraphics</code>	<code>\includegraphics[<i><keys></i>](<i><pfxadd></i>)<<i><ovpfgd></i>>[<i><ovpbgd></i>]{<i><filename></i>}</code> To be used like defined in <code>packagepsfragx</code> .
<code>\savepicture</code>	<code>\savepicture{<i><name></i>}</code> The last output graphics (result of the <code>pspicture</code> or <code>postscript</code> environments or the <code>\includegraphics</code> statement with an PostScript file as argument) is being saved in a file under the name as given by the parameter.
<code>\usepicture</code>	<code>\usepicture[<i><keys></i>]{<i><name></i>}</code> The graphic previously stored with <code>\savepicture</code> is outputted. The optional parameter corresponds to <code>\includegraphics</code> .
<code>pst-pdf-defs</code>	<code>\begin{pst-pdf-defs} ... \end{pst-pdf-defs}</code> For defining macros or environments, which contain character <code>&</code> (others?) in the output, these definitions have to be wrapped with environment <code>pst-pdf-defs</code> .

2.4 Command options

The behaviour of the `\includegraphics` and `\usepicture` statements and the `postscript` environment can be modified with any of the following parameters (key value syntax):

frame=*<true|false>* As with the `\fbox` statement a frame is drawn around the graphics. Any change of size due to rotation is taken into account. Drawing happens in pdfL^AT_EX mode; before, in creating the `\PDFcontainer` file, it is ignored. Default: `false`.

innerframe=*<true|false>* As in “**frame**”, but the frame is drawn around the graphics, not its box.

ignore=*<true|false>* If set to “**true**” no graphics are outputted. With macro `\savepicture{<name>}` the graphics can be used later in a different place via `\usepicture`. Default: `false`.

showname=*<true|false>* A caption of minimal font size records the used file’s name. Default: `false`.

namefont=** Controls the font used when “**showname**=**true**” is set. Default: `\ttfamily\tiny`

All parameters can be set globally as in `\setkeys{Gin}{<key=value>}`.

3 Implementation

1 *<*package>*

3.1 Package options

2 `\newcommand*\ppf@TeX@mode{-1}`
3 `\newcommand*\ppf@draft{false}`

```

4 \newif\if@ppf@PST@used\@ppf@PST@usedtrue
5 \newif\if@ppf@tightpage \@ppf@tightpagetrue
6 \DeclareOption{active}{\OptionNotUsed}
7 \DeclareOption{inactive}{\def\ppf@TeX@mode{9}}
8 \DeclareOption{ignore}{\def\ppf@TeX@mode{999}}
9 \DeclareOption{pstricks}{\@ppf@PST@usedtrue}
10 \DeclareOption{nopstricks}{\@ppf@PST@usedfalse}
11 \DeclareOption{displaymath}{%
12   \PassOptionsToPackage\CurrentOption{preview}}
13 \DeclareOption{draft}{\def\ppf@draft{true}}
14 \DeclareOption{final}{\def\ppf@draft{false}}%
15   \PassOptionsToPackage\CurrentOption{graphicx}}
16 \DeclareOption{notightpage}{\@ppf@tightpagefalse}%
17 \DeclareOption{tightpage}{\@ppf@tightpagetrue}%
18 \DeclareOption*{%
19   \PassOptionsToPackage\CurrentOption{pstricks}}
20 \ProcessOptions\relax
21 \ifnum\ppf@TeX@mode=999\relax\expandafter\endinput\fi

```

3.2 Compiler tests

It is tested which \TeX compiler in which mode of operation is actually used (see ‘graphics.cfg’ in te\TeX / \TeX Live). Accordingly the environments `pspicture` and `postscript` gain each a different range of functions. This test is only executed when the options `active` or `inactive` were not given.

```

22 \RequirePackage{ifpdf,ifxetex,ifvtex}
23 \ifnum\ppf@TeX@mode=-1\relax
24   \ifpdf
25     ⇒ pdf $\text{\TeX}$  or Lua $\text{\TeX}$  are running in PDF mode
26     \def\ppf@TeX@mode{1}%
27     \RequirePackage{luatex85}%
28   \else
29     \ifvtex
30     ⇒ V $\text{\TeX}$ 
31     \def\ppf@TeX@mode{9}%
32   \else
33     \ifxetex
34     ⇒ Xe $\text{\TeX}$ 
35     \def\ppf@TeX@mode{9}%
36   \else
37     ⇒ DVI mode
38     \def\ppf@TeX@mode{0}%
39   \fi
40 \fi
41 \fi
42 \fi
43 \fi
44 \newcommand*\PDFcontainer{}
45 \edef\PDFcontainer{\jobname-pics.pdf}
46 \newcounter{pspicture}
47 \newcommand*\ppf@other@extensions[1]{%
48   \newcommand*\usepicture[2][{}]{%

```

```

44 \newcommand*\savepicture[1]{%

pst-pdf-defs
45 \newenvironment*{pst-pdf-defs}{%
46   \endgroup
47 %    ??? \@currentvline
48 }{%
49   \begingroup
50   \def\@currentvir{pst-pdf-defs}%
51 }

52 \RequirePackage{graphicx}[2017/06/01]%
53 \let\ppf@Gininclude@graphics\Gininclude@graphics
54 \let\ppf@Gin@extensions\Gin@extensions
55 \let\ppf@Gin@ii\Gin@ii

56 \newif\ifppf@pdftex@graphic
57 \newif\ifGin@frame\Gin@framefalse
58 \newif\ifGin@innerframe\Gin@innerframefalse
59 \newif\ifGin@showname\Gin@shownamefalse
60 \newif\ifGin@ignore\Gin@ignorefalse

\ifpr@outer in fact is defined in package preview. We have to do it here too since
otherwise TEX could “stumble and fall” while parsing the \ifcase structure.

61 \newif\ifpr@outer

\ppf@is@pdfTeX@graphic Parameter #1 is the name of a graphics file with or without extension, parameter
#2 contains the valid extensions in PDF mode, parameter #3 contains the valid
extensions in DVI mode. If it works to process the graphics in PDF mode, then
the statements in #4 are executed, otherwise those in #5.

62 \newcommand*\ppf@is@pdfTeX@graphic[5]{%
63   \@ppf@pdftex@graphicfalse%
64   \begingroup
65   \edef\pdfTeXtext{\detokenize\expandafter{#2}}%

Instead of loading the found graphics, only a test on file name extension.

66   \def\Gin@setfile##1##2##3{%
67     \@expandtwoargs\in@{,\detokenize\expandafter{##2},}{,\pdfTeXtext,}%
68     \ifin@global\@ppf@pdftex@graphictrue\fi}%

File types for both modes need to be determined to prevent a wrong error message
“File ‘#1’ not found”.

69   \edef\Gin@extensions{#2,#3}%

Trial invocation. Output is completely inhibited.

70   \pr@outerfalse\ppf@Gininclude@graphics{#1}%
71   \endgroup
72   \ifppf@pdftex@graphic#4\else#5\fi
73 }

74 \ifcase\ppf@TeX@mode\relax

```

3.3 Extraction mode (DVI output)

The `pspicture` environment retains any definition from `pstricks.tex`. Only the code from the environments `pspicture` and `postscript` as well as `\includegraphics` with PostScript files leads to records into the DVI file. The remainder of the document's code is ignored for output. After conversion of the DVI file via PostScript ("dvips") into PDF (`\PDFcontainer` file) each graphics takes exactly one page in the `\PDFcontainer` file. The TeX compiler with DVI output and the package option "active" both force this mode.

```

75 \PackageInfo{pst-pdf}{%
76   MODE: \ppf@TeX@mode\space (dvi -- extraction mode)}
77 \nofiles
78 \let\makeindex\@empty \let\makeglossary\@empty \let\printindex\@empty
79 \renewcommand*\makeindex[1][]{}%
80 \renewcommand*\makeglossary[1][]{}%
81 \renewcommand*\printindex[1][]{}%
82 \AtBeginDocument{\overfullrule=\z@}%
83 \ifppf@PST@used\RequirePackage{pstricks}\fi
84 \RequirePackage{active,dvips,tightpage}{preview}[2005/01/29]%
85 \newcommand*\ppf@PreviewBbAdjust{}
86 \newcommand*\ppf@RestoreBbAdjust{%
87   \let\PreviewBbAdjust\ppf@PreviewBbAdjust}%

```

The pdfLaTeX mode compliant graphics file formats are needed too.

```

88 \begingroup
89   \let\AtBeginDocument\@gobble \let\PackageWarningNoLine\@gobbletwo
90   \chardef\pdftexversion=121 %
91   \newcount\pdfoutput
92   \pdfoutput=1 %
93   \input{pdftex.def}%
94   \edef\x{\endgroup\def\noexpand\ppf@other@extensions{\Gin@extensions}
95   }%
96   \x

```

In PDF mode no rules must be defined for its compliant (PNG, JPEG, PDF) graphics file formats (because of for example 'dvips' extensions). The universal EPS rule is used to at least find these files.

```

97 \AtBeginDocument{%
98   \@ifpackageloaded{keyval}{%
99     \def\KV@errx#1{\PackageInfo{keyval}{#1}}%
100   }{}%
101   \@ifpackageloaded{xkeyval}{%
102     \def\XKV@err#1{\PackageInfo{xkeyval}{#1}}%
103   }{}%

```

In this mode undefined keys should not be an error.

```

104   \@for\@tempa:=\ppf@other@extensions\do{%
105     \expandafter\let\csname Gin@rule@\@tempa\endcsname\relax}%
106   \DeclareGraphicsRule{*}{eps}{*}{}%

```

No function in this mode.

```

107 \define@key{Gin}{innerframe}[true]{}%
108 \define@key{Gin}{frame}[true]{}%
109 \define@key{Gin}{ignore}[true]{}%
110 \define@key{Gin}{showname}[true]{}%

```

```

111 \define@key{Gin}{namefont}{}%
112 \ifundefined{Gin@page}{\define@key{Gin}{page}{}{}}{}

113 \if@ppf@tightpage\else
114 \def\PreviewBbAdjust{%
115 -600pt -600pt 600pt 600pt}%
116 \AtEndDocument{%
117 \PackageWarningNoLine{pst-pdf}{Picture container needs cropping.}}%
118 \fi

```

postscript The postscript environment utilises the trim option in the same manner as does `\includegraphics` (any specification without dimension is interpreted as if given in bp).

```

119 \newenvironment{postscript}[1][]{%
120 {%
121 \global\let\ppf@PreviewBbAdjust\PreviewBbAdjust
122 \if@ppf@tightpage
123 \begingroup
124 \setkeys{Gin}{#1}%
125 \xdef\PreviewBbAdjust{%
126 -\Gin@vllx bp -\Gin@vllly bp \Gin@vurx bp \Gin@vury bp}%
127 \endgroup
128 \fi
129 \ignorespaces
130 }%
131 {\aftergroup\ppf@RestoreBbAdjust}%

132 \PreviewEnvironment{postscript}%
133 \AtBeginDocument{%
134 \ifundefined{PSTricksLoaded}{}%
135 {%

```

pspicture Announce preview original definition.

```

136 \PreviewEnvironment{pspicture}%

```

psmatrix Announce preview original definition.

```

137 \ifundefined{psmatrix}{}%
138 {%
139 \PreviewEnvironment{psmatrix}%
140 \newcommand*\ppf@set@mode{}%
141 \newcommand*\ppf@test@mmode{%
142 \ifmmode
143 \ifinner
144 \let\ppf@set@mode=$%
145 \else
146 \def\ppf@set@mode{$$}%
147 \fi
148 \else
149 \let\ppf@set@mode=\@empty
150 \fi
151 }%
152 \let\ppf@psmatrix=\psmatrix
153 \expandafter\let\expandafter\ppf@pr@psmatrix%
154 \expandafter=\cename pr@\string\psmatrix\endcename

```

```

155     \let\ppf@endpsmatrix=\endpsmatrix
156     \def\psmatrix{\ppf@test@mode\ppf@psmatrix}
157     \expandafter\def\csname pr@string\psmatrix\endcsname{%
158         \ppf@set@mode\ppf@pr@psmatrix}%
159     \def\endpsmatrix{\ppf@endpsmatrix\ppf@set@mode}%
160 }%

```

Announce internal macro `\pst@object` to enable the use of some PSTricks code outside of `pspicture` environments. At the moment invocations of the following kind are feasible:

```

\pst@object {<m>}<*>[<o>]{<o>}{<o>}<(>o)>(<o>)<(>o)>(<o>)
(m = necessary, * = optional, o = optional)

```

More than three optional arguments at the call's end, as in `\psline` possible, do not work yet.

```

161     \PreviewMacro[{}*[]%
162     ?\bggroup{##1}{\#1}}{}%
163     ?\bggroup{##1}{\#1}}{}%
164     ?({\#1}{\#1}}{\#1}}{}%
165     ?({\#1}{\#1}}{\#1}}{}%
166     ?({\#1}{\#1}}{\#1}}{}%
167     ]{\pst@object}}

```

Prevent multiple test-wise setting of table contents by “tabularx”.

```

168     \@ifundefined{tabularx}{}{}%
169     \newcolumntype{X}{c}%
170     \expandafter\let\expandafter\tabularx\csname tabular*\endcsname
171     \expandafter\let\expandafter\endtabularx\csname endtabular*\endcsname
172 }%

```

Support of `\includegraphicx` from the package `psfragx`.

```

173     \@ifundefined{pfx@includegraphicx}{}{}%
174     \PreviewMacro[{}{}]{\pfx@includegraphicx}%
175 }%

```

`\Gscale@@box` Disable scaling.

```

176     \def\Gscale@@box#1#2#3{%
177         \toks@{\mbox}%
178     }

```

`\Ginclude@graphics` All graphics content of well known format (for instance EPS files) is treated in a regular way, which in this mode denotes that it is subject to `preview` functions. Other graphics content (for instance PDF files) is ignored.

```

179     \def\Ginclude@graphics#1{%
180         \ifpr@outer

```

Generally pdfTeX supported graphics formats are intended to be preferred (inclusion in final pdfTeX run). If it's a PostScript type graphics, then the original definition is in function again and registration for the `preview` package is necessary in order to convert this PostScript type graphics into PDF.

```

181         \ppf@is@pdfTeX@graphic{#1}{\ppf@other@extensions}{\Gin@extensions}%

```


Dummy box to prevent a division by zero while scaling or rotating. Otherwise ignored.

```
182     {\rule{10pt}{10pt}}%
183     {\ppf@Ginclude@graphics{#1}}%
184     \else
```

Inside a PostScript environment (pspicture etc.) `\includegraphics` has to behave as in its original definition (only DVIPS supported graphics formats are allowed).

```
185     \ppf@Ginclude@graphics{#1}%
186     \fi
187 }%

188 \PreviewMacro[{}]{\ppf@Ginclude@graphics}%
189 \let\pdfliteral\@gobble%
190 \or
```

3.4 pdf_{La}T_EX mode (PDF output)

When the `\PDFcontainer` file (default: `\jobname`-pics.pdf) exists, the contents of the environments `pspicture` and `postscript` is ignored. Instead the corresponding graphics from the `\PDFcontainer` file is used.

```
191 \PackageInfo{pst-pdf}{MODE: \ppf@TeX@mode\space (pdfTeX mode)}%
Prevent pdfTeX's message Non-PDF special ignored!.

192 \ifppf@PST@used
193   \let\ppf@temp\AtBeginDvi\let\AtBeginDvi\@gobble
194   \def\c@lor@to@ps#1 #2\@{}
195   \RequirePackage{pstricks}\let\AtBeginDvi\ppf@temp
196 \fi

197 \@temptokena{%
198   \let\Gin@PS@file@header\@gobble\let\Gin@PS@literal@header\@gobble
199   \let\Gin@PS@raw\@gobble\let\Gin@PS@restored\@gobble
200   \@ifundefined{PSTricksLoaded}{-}%
```

Necessary if PSTricks < 2.0.

```
201   \PSTricksOff
202   \def\c@lor@to@ps#1 #2\@{}
203 }%
204 }
```

PostScript output is now inhibited and later once again.

```
205 \the\@temptokena
206 \expandafter\AtBeginDocument\expandafter
207   {\the\@temptokena\@temptokena{}}%
208 \@ifundefined{PSTricksLoaded}{-}%
```

To parse the arguments of `PSTricks'` `\pst@object` we load `preview` in active mode, but restore the default definitions of `\output` and `\shipout`. `\pr@startbox` and `\pr@endbox` serve here only to disable `\pst@object` and to load the corresponding graphics from the `\PDFcontainer` file. At present a maximum of three optional parameters in round braces (parenthesis) at the end of `\pst@object` is supported, which is sufficient, but not always enough.

```
209 \newtoks\ppf@output
210 \ppf@output\expandafter{\the\output}%
```

```

211 \let\ppf@nofiles=\nofiles \let\nofiles=\relax
212 \let\ppf@shipout=\shipout
213 \RequirePackage[active]{preview}[2005/01/29]%
214 \let\shipout=\ppf@shipout \let\ppf@shipout=\relax
215 \let\nofiles=\ppf@nofiles \let\ppf@nofiles=\relax
216 \output\expandafter{\the\ppf@output} \ppf@output{}%

```

\pr@startbox, \pr@endbox: simpler over original definitions.

```

217 \long\def\pr@startbox#1#2{%
218   \ifpr@outer
219     \toks@{#2}%
220     \edef\pr@cleanup{\the\toks@}%
221     \setbox\@tempboxa\vbox\bgroup
222     \everydisplay{}%
223     \pr@outerfalse%
224     \expandafter\@firstofone
225   \else
226     \expandafter\@gobble
227   \fi{#1}}%
228 \def\pr@endbox{%
229   \egroup
230   \setbox\@tempboxa\box\voidb@x
231   \ppf@@getpicture
232   \pr@cleanup}%

```

(See also the identical definition in DVI mode.)

```

233 \AtBeginDocument{%
234   \ifundefined{pst@object}{}%
235   {%
236     \PreviewMacro[{}*[]%
237     ?\bgroup{#{#1}{#{#1}}}{}%
238     ?\bgroup{#{#1}{#{#1}}}{}%
239     ?({#{#1}){({#1})}){}%
240     ?({#{#1}){({#1})}){}%
241     ?({#{#1}){({#1})}){}%
242     }]{\pst@object}}%
243   }%
244 }%

```

Too the supported file name extensions from DVI mode are needed.

```

245 \begingroup
246   \input{dvips.def}%
247   \edef\x{\endgroup\def\noexpand\ppf@other@extensions{\Gin@extensions}}%
248   \x

```

Dummy definition for in DVI mode supported file formats.

```

249 \DeclareGraphicsRule{*}{eps}{*}{}%
250 \define@key{Gin}{innerframe}[true]{%
251   \lowercase{\Gin@boolkey{#1}}{innerframe}}%
252 \define@key{Gin}{frame}[true]{%
253   \lowercase{\Gin@boolkey{#1}}{frame}}%
254 \define@key{Gin}{ignore}[true]{%
255   \lowercase{\Gin@boolkey{#1}}{ignore}}%
256 \define@key{Gin}{frame@@}{%

```

(For internal use only!)

```

257 \edef\@tempa{\toks@{\noexpand\frame{\the\toks@}}}%
258 \ifcase#1\relax
259 \ifGin@innerframe\else\let\@tempa\relax\fi
260 \or
261 \ifGin@frame\else\let\@tempa\relax\fi
262 \fi
263 \@tempa
264 }%
265 \define@key{Gin}{showname}[true]{%
266 \lowercase{\Gin@boolkey{#1}}{showname}}%
267 \define@key{Gin}{namefont}{%
268 \beginngroup
269 \@temptokena\expandafter{\ppf@namefont#1}%
270 \edef\x{\endgroup\def\noexpand\ppf@namefont{\the\@temptokena}}%
271 \x
272 }%
273 \newcommand*\ppf@filename{}%
274 \newcommand*\ppf@namefont{\tiny\ttfamily}%
275 \newcommand*\ppf@Gin@keys{}%
276 \let\ppf@Gin@setfile\Gin@setfile

```

\Gin@setfile Save real file name and, if applicable, page number for later use.

```

277 \def\Gin@setfile#1#2#3{\ppf@Gin@setfile{#1}{#2}{#3}%
278 \xdef\ppf@filename{%
279 #3\ifx\Gin@page\@empty\else(\Gin@page)\fi}}%

```

\Gin@ii Examine the options “frame”, “ignore”, etc. as soon as other special cases.

```

280 \def\Gin@ii[#1]#2{%
281 \beginngroup

```

The value of `\ifGin@innerframe` has to be known before the inner frame is drawn. The values for `\ifGin@showname` and `\ppf@namefont` need to be available after rendering the graphics too. Thus beforehand and protected inside a group examine the options.

```

282 \@temptokena{#1}\def\ppf@tempb{#2}%

```

Finds empty file name when calling `\usepicture`.

```

283 \ifx\ppf@tempb\@empty\else
284 \ppf@is@pdfTeX@graphic{#2}{\Gin@extensions}{\ppf@other@extensions}%

```

Graphics out of `\PDFcontainer` are complete – scaled, rotated, etc. Don’t apply these things again and therefore ignore the optional parameters.

```

285 {%
286 \setkeys{Gin}{#1}%
287 \ifx\ppf@tempb\PDFcontainer
288 \@temptokena{page=\Gin@page}%
289 \fi
290 }%
291 {%
292 \refstepcounter{pspicture}%
293 \@temptokena{page=\the\c@pspicture}\def\ppf@tempb{\PDFcontainer}%
294 }%
295 \fi
296 \ifGin@ignore\else

```

“frame@@=0” = inner frame, “frame@@=1” = outer frame.

```

297     \edef\@tempa{\noexpand\ppf@Gin@ii[frame@@=0,\the\@temptokena,
298         frame@@=1]{\ppf@tempb}}}%
299     \@tempa
300     \ifGin@showname
301         \ppf@namefont
302         \raisebox{-\ht\strutbox}[Opt][Opt]{\llap{\ppf@filename}}%
303         \gdef\ppf@filename{}%
304     \fi
305 \fi
306 \endgroup
307 }%

308 \IfFileExists{\PDFcontainer}%
309 {%

```

\ppf@container@max The number of pages as contained in \PDFcontainer file.

```

310     \pdfximage{\PDFcontainer}%
311     \edef\ppf@container@max{\the\pdflastximagepages}%

312     \AtEndDocument{%
313         \ifnum\c@pspicture>\z@

```

A warning only makes sense when a graphics is needed at all.

```

314         \ifnum\c@pspicture=\ppf@container@max\else
315             \PackageWarningNoLine{pst-pdf}{%
316                 ‘\PDFcontainer’ contains \ppf@container@max\space pages
317                 \MessageBreak but \the\c@pspicture\space pages are requested:
318                 \MessageBreak File ‘\PDFcontainer’ is no more valid!
319                 \MessageBreak Recreate it
320             }%
321         \fi
322     \fi
323 }%
324 }%
325 {%
326     \def\ppf@container@max{0}%
327     \AtEndDocument{%
328         \ifnum\c@pspicture>\z@
329             \filename@parse{\PDFcontainer}%
330             \PackageWarningNoLine{pst-pdf}{%
331                 File ‘\PDFcontainer’ not found.\MessageBreak
332                 Use the following commands to create it:\MessageBreak
333                 -----
334                 \MessageBreak
335                 latex \jobname.tex\MessageBreak
336                 dvips -o \filename@base.ps \jobname.dvi\MessageBreak
337                 ps2pdf \filename@base.ps\MessageBreak
338                 -----
339             }%
340         \fi
341     }%
342 }%

```

`\ppf@isnum` If parameter #1 is numeric, the instructions in #2, otherwise those in #3 are executed (see `bibtopic.sty`).

```
343 \newcommand\ppf@isnum[1]{%
344   \if!\ifnum9<1#1!\else_\fi\expandafter\@firstoftwo
345   \else\expandafter\@secondoftwo\fi}%
```

`psmatrix` Both environments ignore their contents and load instead the corresponding graphics out of the `\PDFcontainer` file. The value of the herein used `pspicture` counter's value can be used in `\label/\ref`.

`postscript`

```
346 \newcommand*\ppf@set@mode{%
347 \newcommand*\ppf@test@mmode{%
348 \ifmmode
349   \ifinner
350     \let\ppf@set@mode=$%
351   \else
352     \def\ppf@set@mode{${}%
353   \fi
354 \else
355   \let\ppf@set@mode=\@empty
356 \fi
357 }

358 \RequirePackage{environ}%
359 \newenvironment{postscript}[1][]{%
360   \def\@tempa{postscript}%
361   \ifx\@tempa\@currenvir
362     \def\ppf@Gin@keys{#1}%
363   \else
364     \def\ppf@Gin@keys{}%
365   \fi
366   \ppf@@getpicture
367   \Collect@Body\@gobble}{}%
368 \AtBeginDocument{%
369   \@ifundefined{PSTricksLoaded}{}{%
370     \def\pst@@@picture[#1](#2,#3)(#4,#5){\postscript}%
371     \def\endpspicture{\endpostscript\endgroup}%
372     \@ifundefined{psmatrix}{}{%
373       \let\psmatrix=\postscript
374       \let\endpsmatrix=\endpostscript}%
375   }%
376   \@ifundefined{pfx@includegraphicx}{}{%
```

The useless redefinition of `\includegraphics` in pdfTeX mode (package `psfragx`) is leading to double insertion of the result. We go back to the original meaning.

```
377   \let\includegraphics=pfx@includegraphics
378   \def\pfx@includegraphicx#1#2{\ppf@@getpicture}%
379 }%
380 }%
```

`\savepicture` Saves the recent graphics' number in a macro named `\ppf@@@#1`.

```
381 \def\savepicture#1{%
382   \expandafter\xdef\csname ppf@@@#1\endcsname{\the\pdflastximage}}%
```

`\usepicture` Inserts graphics with symbolic name #2. This name has to be declared beforehand in `\savepicture{<name>}`. Instead of a name a number can be used too, which directly addresses a graphics in the `\PDFcontainer` file. The optional parameter #1 corresponds to the one in `\includegraphics`.

```

383 \renewcommand*\usepicture[2][]{%
384   \@ifundefined{ppf@@@#2}%
385   {%
386     \ppf@isnum{#2}%
387     {\ppf@getpicture{#1}{#2}}%
388     {\@latex@error{picture '#2' undefined}\@ehc}%
389   }%
390   {%
391     \begingroup
392     \def\Gin@include@graphics##1{%
393       \xdef\ppf@filename{#2}%
394       \setbox\z@\hbox{\pdfrefximage\@nameuse{ppf@@@#2}}%
395       \Gin@nat@height\ht\z@ \Gin@nat@width\wd\z@
396       \def\Gin@llx{0} \let\Gin@lly\Gin@llx
397       \Gin@defaultbp\Gin@urx{\Gin@nat@width}%
398       \Gin@defaultbp\Gin@ury{\Gin@nat@height}%
399       \Gin@bboxtrue\Gin@viewport@code
400       \Gin@nat@height\Gin@ury bp%
401       \advance\Gin@nat@height-\Gin@lly bp%
402       \Gin@nat@width\Gin@urx bp%
403       \advance\Gin@nat@width-\Gin@llx bp%
404       \Gin@req@sizes
405       \ht\z@\Gin@req@height \wd\z@\Gin@req@width
406       \leavevmode\box\z@}%
407     \define@key{Gin}{type}{}%
408     \includegraphics[scale=1,#1]{}%
409   \endgroup
410 }%

```

`\ppf@getpicture` Inserts the page (graphics) with number #2 from the `\PDFcontainer` file. Parameter #1: any option as in `\includegraphics`.

```

411 \newcommand*\ppf@getpicture[2]{%
412   \@tempcnta=#2\relax%
413   \ifnum\@tempcnta>\ppf@container@max
414     \PackageWarningNoLine{pst-pdf}{%
415       pspicture No. \the\@tempcnta\space undefined}%
416   \else
417     \includegraphics[draft=\ppf@draft,#1,page=\the\@tempcnta]%
418     {\PDFcontainer}%
419   \fi
420   \gdef\ppf@Gin@keys{}}%

```

`\ppf@@getpicture` Inserts next page (graphics) from the `\PDFcontainer` file.

```

421 \newcommand*\ppf@@getpicture{%
422   \ifpr@outer
423     \refstepcounter{pspicture}%
424     \expandafter\ppf@getpicture\expandafter{\ppf@Gin@keys}%
425     {\the\c@pspicture}%
426   \fi}%

```

`pst-pdf-defs` Environment without grouping. The character `&` has the catcode “other”. Useful for user-defined macro definitions with e. g. `psmatrix` inside.

```

427 \renewenvironment*{pst-pdf-defs}%
428 {%
429     \endgroup
430 %     ??? \@currencline
431     \chardef\ppf@temp=\catcode'\&%
432     \@makeother\&%
433 }{%
434     \catcode'\&=\ppf@temp
435     \begingroup
436     \def\@currenvir{pst-pdf-defs}%
437 }
438 \else

```

3.5 Inactive Mode

Only the packages `pstricks` and `graphicx` are loaded – no further exertion of influence. The package option “inactive” as soon as the `VTEX` compiler force this mode.

```

439 \PackageInfo{pst-pdf}{MODE: \ppf@TeX@mode\space (inactive mode)}%
440 \newenvironment{postscript}[1] []{\ignorespaces}{\relax}
441 \let\ppf@is@pdfTeX@graphic\relax
442 \fi

443 \InputIfFileExists{pst-pdf.cfg}{%
444     \PackageInfo{pst-pdf}{Local config file pst-pdf.cfg used}}{}
445 \endpackage

```