The pst-pdf package*

Rolf Niepraschk† Hubert Gäßlein‡

2005/06/29

1 Introduction

Das Paket pst-pdf vereinfacht die Verwendung von PSTricks-Grafiken und anderem PostScript-Code in PDF-Dokumenten. Ähnlich wie beim Erstellen des Literaturverzeichnisses mit bibTEX werden zusätzlich externe Programme aufgerufen. Sie dienen in diesem Fall dazu, eine PDF-Datei, die sämtliche Grafiken enthält, zu erzeugen. Ihr Inhalt wird im endgültigen Dokument statt des ursprünglichen PostScript-Codes eingefügt.

2 Usage

2.1 Package options

active Aktiviert den Extraktionsmodus (DVI-Ausgabe). Die explizite Angabe ist normalerweise unnötig (Standard im LATEX-Modus).

inactive Keine besonderen Aktionen; es werden nur die Pakete pstricks und graphicx geladen (Standard bei Verwendung von VTEX). Kann dazu benutzt werden, um das Dokument mit LATEX in eine DVI-Datei zu wandeln und dabei die automatische Verwendung des Extraktionsmodus' zu vermeiden.

pstricks Das Paket pstricks wird geladen (Standard).

nopstricks Das Paket pstricks wird nicht geladen. Wird später festgestellt, dass pstricks doch noch anderweitig geladen wurde, wird die Umgebung `pspicture` nachträglich in der Weise behandelt, als wäre die Option "pstricks" doch angegeben worden.

draft Im pdfLATEX-Modus werden aus der Containerdatei eingefügte Grafiken nur als Rahmen dargestellt.

final Im pdfLATEX-Modus werden aus der Containerdatei eingefügte Grafiken vollständig dargestellt (Standard).

tightpage Die Abmessung Grafiken in der Containerdatei entsprechen denen der zugehörigen TEX-Boxen (Standard).

*This document corresponds to pst-pdf v1.1i, dated 2005/06/29.

†Rolf.Niepraschk@ptb.de

‡HubertJG@open.mind.de

notightpage Die Abmessung der zur Grafik gehörenden TEX-Box ist manchmal nicht korrekt, da PostScript-Anweisungen auch außerhalb der Box zeichnen können. Die Option "notightpage" führt dazu, dass die Grafiken in der Containerdatei mindestens die Größe des gesamten Blattes einnehmen. Um die Grafiken im späteren pdfLATEX-Lauf verwenden zu können, muss die Containerdatei nachbearbeitet werden, so dass die Größe der Grafiken auf die der sichtbaren Bestandteile reduziert ist. Dazu kann z. B. das Programm `pdfcrop`[1] dienen. Die Anwendung dieses Verfahrens kann die Angabe der Option "trim" erübrigen (siehe Abschnitt 2.4).

displaymath Es werden zusätzlich die mathematischen Umgebungen `displaymath`, `eqnarray` und `$$` extrahiert und im pdf-Modus als Grafik eingefügt. So können zusätzliche PSTricks-Ergänzungen leicht dem Inhalt dieser Umgebungen zugefügt werden. (Frage: Wie verhalten sich die AMSLATEX-Umgebungen?)

⟨***other***⟩ Alle anderen Optionen werden an das Paket pstricks weitergereicht.

2.2 Program calls

Die folgende Tabelle zeigt den Ablauf, der nötig ist, um ein PDF-Dokument mit PostScript-Grafiken zu erzeugen[2]. Im Vergleich dazu ist der analoge Ablauf für Literaturverzeichnisse angegeben.

PostScript-Grafiken	**Literaturverzeichnis**
`pdflatex document.tex`	`pdflatex document.tex`
Hilfsaufrufe	
`latex document.tex`	
`dvips -o document-pics.ps document.dvi`	
`ps2pdf document-pics.ps`	`bibtex document.aux`
`pdflatex document.tex`	`pdflatex document.tex`

Bei der Erzeugung wird nur Code berücksichtigt, der sich innerhalb der Umgebungen `pspicture` oder `postscript` befindet. Ebenfalls werden PostScript-Grafiken, die als Parameter von `\includegraphics` angegeben wurden, der Containerdatei hinzugefügt. Der Name dieser Datei ist standardmäßig ⟨`\jobname`⟩`-pics.pdf`. Er kann durch Umdefinieren des Makros `\PDFcontainer` geändert werden.

2.3 User commands

pspicture `\begin{pspicture}[`⟨*keys*⟩`] (`⟨*x0,x1*⟩`)(`⟨*y0,y1*⟩`) ... \end{pspicture}`

Die `pspicture`-Umgebung steht zur Verfügung, wenn nicht die Option "`nopstricks`" angegeben wurde. Sie wird so wie in PSTricks üblich verwendet. Im pdfLATEX-Modus wird ihr Inhalt nur dann dargestellt, wenn vorher die Containerdatei erzeugt wurde.

postscript `\begin{postscript}[`⟨*keys*⟩`] ... \end{postscript}`

[1] CTAN: `support/pdfcrop/`

[2] Das in der TEX-Distribution "teTEX" enthaltene Unix-Shell-Script `ps4pdf` führt alle angegebene Programmaufrufe aus. Siehe auch: CTAN: `macros/latex/contrib/ps4pdf/`

Die `postscript`-Umgebung kann beliebigen Code mit Ausnahme von Gleitumgebungen aufnehmen. Im pdfLaTeX-Modus wird ihr Inhalt ebenfalls der Containerdatei entnommen. Ist diese Datei nicht vorhanden, wird – anders als bei der `pspicture`-Umgebung – der später benötigte Platz möglicherweise nicht korrekt frei gehalten.

`\includegraphics` `\includegraphics[`⟨*keys*⟩`]{`⟨*filename*⟩`}`
Wie in graphics/graphicx definiert zu verwenden. Zusätzlich ist es nun möglich, auch im pdfLaTeX-Modus EPS-Dateien als Argument anzugeben und ihren Inhalt darzustellen. Er wird dazu ebenfalls der Containerdatei entnommen.

`\savepicture` `\savepicture{`⟨*name*⟩`}`
Die zuletzt ausgegebene Grafik (Ergebnisse der Umgebungen `pspicture`, `postscript` und der `\includegraphics`-Anweisungen mit PostScript-Dateien) wird unter dem als Parameter übergebenen Namen gespeichert.

`\usepicture` `\usepicture[`⟨*keys*⟩`]{`⟨*name*⟩`}`
Die zuvor mit `\savepicture` gespeicherte Grafik wird ausgegeben. Der optionale Parameter entpricht dem bei der Anweisung `\includegraphics` möglichen.

2.4 Command options

Das Verhalten der Anweisungen `\includegraphics`, `\usepicture` und der Umgebung `postscript` kann mit den folgenden optionalen Parametern beeinflusst werden (key-value-Syntax):

frame=⟨true|false⟩ Es wird – ähnlich wie bei der Anweisung `\fbox` – ein Rahmen um die Grafik gezeichnet. Die durch Rotation geänderte Gesamtgröße wird dabei berücksichtigt. Das Zeichnen geschieht im pdfLaTeX-Modus; vorher beim Erzeugen der Containerdatei wird dieser Parmeter ignoriert. Standard: false.

innerframe=⟨true|false⟩ Wie "`frame`" jedoch wird der Rahmen nur um die Grafik selbst, nicht aber um die resultierende Box gezeichnet.

ignore=⟨true|false⟩ Bei "`true`" wird die Grafik nicht ausgegeben. Bei Angabe von `\savepicture{`⟨*name*⟩`}` kann sie später jedoch an anderer Stelle mit `\usepicture` verwendet werden. Standard: false.

showname=⟨true|false⟩ Gibt in kleiner Schrift den tatsächlich verwendeten Dateinamen unter der Grafik aus. Standard: false.

namefont=⟨font commands⟩ Beinflusst die Schriftart, die bei "`showname=true`" benuzt wird. Standard: `\ttfamily\tiny`

Alle Parameter können auch global per `\setkeys{Gin}{`⟨*key=value*⟩`}` gesetzt werden.

3 Implementation

1 ⟨∗package⟩

3.1 Package options

```
\newcommand*\ppf@TeX@mode{-1}
\newcommand*\ppf@draft{false}
\newif\if@ppf@PST@used\@ppf@PST@usedtrue
\newif\if@ppf@tightpage \@ppf@tightpagetrue
\DeclareOption{active}{\def\ppf@TeX@mode{0}}
\DeclareOption{inactive}{\def\ppf@TeX@mode{9}}
\DeclareOption{ignore}{\def\ppf@TeX@mode{999}}
\DeclareOption{pstricks}{\@ppf@PST@usedtrue}
\DeclareOption{nopstricks}{\@ppf@PST@usedfalse}
\DeclareOption{displaymath}{%
  \PassOptionsToPackage\CurrentOption{preview}}
\DeclareOption{draft}{\def\ppf@draft{true}}
\DeclareOption{final}{\def\ppf@draft{false}%
  \PassOptionsToPackage\CurrentOption{graphicx}}
\DeclareOption{notightpage}{\@ppf@tightpagefalse}%
\DeclareOption{tightpage}{\@ppf@tightpagetrue}%
\DeclareOption*{%
  \PassOptionsToPackage\CurrentOption{pstricks}}
\ProcessOptions\relax
\ifnum\ppf@TeX@mode=999\relax\expandafter\endinput\fi
```

3.2 Compilertests

Es wird getestet, welcher TEX compiler in welchem Modus läuft (siehe '`graphics.cfg`' von teTEX/TEXLive). Entsprechend dem Ergebnis bekommen die Umgebungen `pspicture` und `postscript` unterschiedliche Funktionalität. Der Test wird nur ausgeführt, wenn nicht die Paketoptionen `active` oder `inactive` angegeben wurden.

```
\ifnum\ppf@TeX@mode=-1\relax
  \begingroup
```

Default (TEX with a dvi-to-ps converter)

```
    \chardef\x=0 %
```

Check pdfTEX

```
    \@ifundefined{pdfoutput}{}{%
      \ifcase\pdfoutput\else
        \chardef\x=1 %
      \fi
    }%
```

Check VTEX

```
    \@ifundefined{OpMode}{}{\chardef\x=2 }%
  \expandafter\endgroup
  \ifcase\x
```

⇒ dvi mode

```
    \def\ppf@TeX@mode{0}%
  \or
```

⇒ pdfTEX is running in pdf mode

```
    \def\ppf@TeX@mode{1}%
  \else
```

⇒ VTEX is running

```
    \def\ppf@TeX@mode{9}%
```

```
  \fi
\fi

\newcommand*\PDFcontainer{}
\edef\PDFcontainer{\jobname-pics.pdf}
\newcounter{pspicture}
\newcommand*\ppf@other@extensions[1]{}
\newcommand*\usepicture[2][]{}
\newcommand*\savepicture[1]{}

\RequirePackage{graphicx}%
\let\ppf@Ginclude@graphics\Ginclude@graphics
\let\ppf@Gin@extensions\Gin@extensions
\let\ppf@Gin@ii\Gin@ii

\newif\if@ppf@pdftex@graphic
\newif\ifGin@frame\Gin@framefalse
\newif\ifGin@innerframe\Gin@innerframefalse
\newif\ifGin@showname\Gin@shownamefalse
\newif\ifGin@ignore\Gin@ignorefalse
```

`\ifpr@outer` wird eigentlich im Paket preview definiert. Wir müssen es aber bereits hier zusätzlich tun, da sonst TEX u. U. beim Parsen der \ifcase-Struktur "außer Tritt" kommt.

```
\newif\ifpr@outer
```

`\ppf@is@pdfTeX@graphic` Parameter `#1` ist der Name einer Grafikdatei mit oder ohne Endung, Parameter `#2` enthält die gültigen Dateiendungen im pdf-Modus, Parameter `#3` enthält die gültigen Dateiendungen im dvi-Modus. Ist es möglich, die Grafik im pdf-Modus zu verarbeiten, werden die Anweisungen in `#4` ausgeführt, sonst die in `#5`.

```
\newcommand*\ppf@is@pdfTeX@graphic[5]{%
  \@ppf@pdftex@graphicfalse%
  \begingroup
    \edef\pdfTeXext{#2}%
```

Statt Einladen einer identifizierten Grafik nur Test der Grafikendung.

```
    \def\Gin@setfile##1##2##3{%
      \edef\@tempb{##2}%
      \@for\@tempa:=\pdfTeXext\do{%
        \ifx\@tempa\@tempb\global\@ppf@pdftex@graphictrue\fi}}%
```

Es müssen Dateitypen beider Moden gefunden werden, um die Fehlermeldung "File '`#1`' not found" zu vermeiden.

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

Testaufruf. Dabei Ausgabe vollständig verhindern.

```
    \pr@outerfalse\ppf@Ginclude@graphics{#1}%
  \endgroup
  \if@ppf@pdftex@graphic#4\else#5\fi
}

\if@ppf@PST@used\RequirePackage{pstricks}\fi
\ifcase\ppf@TeX@mode\relax
```

3.3 Extraction mode (dvi output)

Die Umgebung `pspicture` behält die Definition aus `pstricks.tex`. Ausschließlich der Code der Umgebungen `pspicture` und `postscript` sowie `\includegraphics` mit PS-Dateien bewirken Einträge in die DVI-Datei. Der restliche Code des Dokuments wird bei der Ausgabe der DVI-Datei ignoriert. Nach Wandlung der DVI-Datei über PostScript ("`dvips`") nach PDF (Datei `\PDFcontainer`) nimmt jede Grafik genau eine Seite der pdf-Datei ein. Der TEX-Compiler mit DVI-Ausgabe sowie die Paketoption "`active`" erzwingen diesen Modus.

```
\PackageInfo{pst-pdf}{%
  MODE: \ppf@TeX@mode\space (dvi -- extraction mode)}
\RequirePackage[active,dvips,tightpage]{preview}[2005/01/29]%
\newcommand*\ppf@PreviewBbAdjust{}
\newcommand*\ppf@RestoreBbAdjust{%
  \let\PreviewBbAdjust\ppf@PreviewBbAdjust}%
```

Es werden auch die im pdfLATEX-Modus erlaubten Endungen von Grafikdateien benötigt.

```
\begingroup
  \let\AtBeginDocument\@gobble \let\PackageWarningNoLine\@gobbletwo
  \def\pdftexversion{121}\input{pdftex.def}%
  \edef\x{\endgroup\def\noexpand\ppf@other@extensions{\Gin@extensions}
  }%
\x
```

Für die im PDF-Modus möglichen Grafikformate dürfen keine speziellen Regeln definiert sein (z. B. wegen 'dvips'-Erweiterungen). Für sie wird die universelle EPS-Regel verwendet, damit sie zumindest gefunden werden.

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

In diesem Modus keine Funktion.

```
\define@key{Gin}{innerframe}[true]{}%
\define@key{Gin}{frame}[true]{}%
\define@key{Gin}{ignore}[true]{}%
\define@key{Gin}{showname}[true]{}%
\define@key{Gin}{namefont}{}%

\if@ppf@tightpage\else
  \def\PreviewBbAdjust{%
    -.5\paperwidth -.5\paperheight .5\paperwidth .5\paperheight}%
  \AtEndDocument{%
    \PackageWarningNoLine{pst-pdf}{Picture container needs cropping.}}%
\fi
```

`postscript` Die Umgebung postscript wertet die trim-Option in derselben Weise wie `\includegraphics` aus (Angaben ohne Maßeinheit werden als bp interpretiert).

```
\newenvironment{postscript}[1][]%
{%
  \global\let\ppf@PreviewBbAdjust\PreviewBbAdjust
  \if@ppf@tightpage
    \begingroup
      \setkeys{Gin}{#1}%
```

```
        \xdef\PreviewBbAdjust{%
          -\Gin@vllx bp -\Gin@vlly bp \Gin@vurx bp \Gin@vury bp}%
      \endgroup
    \fi
    \ignorespaces
  }%
  {\aftergroup\ppf@RestoreBbAdjust}%

  \PreviewEnvironment{postscript}%
  \AtBeginDocument{%
    \@ifundefined{PSTricksLoaded}{}%
    {%
```

`pspicture` Originaldefinition preview bekannt machen.

```
      \PreviewEnvironment{pspicture}%
```

`psmatrix` Originaldefinition preview bekannt machen.

```
      \@ifundefined{psmatrix}{}%
      {%
        \PreviewEnvironment{psmatrix}%
        \newcommand*\ppf@set@mode{}%
        \newcommand*\ppf@test@mmode{%
        \ifmmode
          \ifinner
            \let\ppf@set@mode=$%
          \else
            \def\ppf@set@mode{$$}%
          \fi
        \else
          \let\ppf@set@mode=\@empty
        \fi
        }%
        \let\ppf@psmatrix=\psmatrix
        \expandafter\let\expandafter\ppf@pr@psmatrix%
          \expandafter=\csname pr@\string\psmatrix\endcsname
        \let\ppf@endpsmatrix=\endpsmatrix
        \def\psmatrix{\ppf@test@mmode\ppf@psmatrix}
        \expandafter\def\csname pr@\string\psmatrix\endcsname{%
          \ppf@set@mode\ppf@pr@psmatrix}%
        \def\endpsmatrix{\ppf@endpsmatrix\ppf@set@mode}%
      }%
```

Internes Makro `\pst@object` bekanntmachen, um manchen PSTricks-Code außerhalb von pspicture-Umgebungen ebenfalls verwenden zu können. Derzeit sind Aufrufe der folgenden Art möglich:

`\pst@object {`⟨*m*⟩`}`⟨`*`⟩`[⟨*o*⟩]{⟨*o*⟩}{⟨*o*⟩}(⟨*o*⟩)(⟨*o*⟩)(⟨*o*⟩)
(`m` = notwendig, `*` = optional, `o` = optional)

Mehr als drei optionale Argumente am Ende des Aufrufs, wie beispielsweise bei `\psline` denkbar, sind noch nicht möglich.

```
      \PreviewMacro[{{}*[]%
        ?\bgroup{#{#1}{{#1}}}{}%
        ?\bgroup{#{#1}{{#1}}}{}%
```

```
        ?({#{(#1)}{({#1})}}{}%
        ?({#{(#1)}{({#1})}}{}%
        ?({#{(#1)}{({#1})}}{}%
        }]{\pst@object}}
```

Mehrfaches testweises Setzen von Tabelleninhalten durch “tabularx” verhindern.

```
    \@ifundefined{tabularx}{}{%
      \def\tabularx#1#2{\tabular{#2}}%
      \newcolumntype{X}{c}%
      \let\endtabularx=\endtabular}%
  }%
```

`\Ginclude@graphics` Alle Grafiken mit bekanntem Format (z. B. EPS-Dateien) werden normal verarbeitet, was in diesem Modus bedeutet, dass sie der Preview-Funktionalität unterliegen. Andere Grafiken (z. B. PDF-Dateien) werden ignoriert.

```
  \def\Ginclude@graphics#1{%
    \ifpr@outer
```

Im allgemeinen Fall sollen pdfTEX-Grafiken bevorzugt werden (Einfügen erst im pdfTEX-Modus). Ist nur eine DVIPS-Graphik vorhanden, dann wirkt wieder die Originaldefintion und Registrierung beim preview-Paket muss erfolgen.

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

Dummy-Box, um Division durch Null bei Skalierung/Rotation zu vermeiden. Wird ansonsten ignoriert.

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

Innerhalb von PS-Umgebungen (`pspicture` usw.) muss sich `\includegraphics` wie die Originaldefinition verhalten (nur die DVIPS-Graphik-Typen sind gültig).

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

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

3.4 pdfLaTeX mode (pdf output)

Ist die Datei `\PDFcontainer` (default: ⟨*\jobname*⟩`-pics.pdf`) vorhanden, so wird der Inhalt der Umgebungen `pspicture` und `postscript` ignoriert. Stattdessen wird die zugehörige Grafik aus der Datei `\PDFcontainer` eingebunden.

```
  \PackageInfo{pst-pdf}{MODE: \ppf@TeX@mode\space (pdfTeX mode)}%
  \@temptokena{%
    \let\Gin@PS@file@header\@gobble\let\Gin@PS@literal@header\@gobble
    \let\Gin@PS@raw\@gobble\let\Gin@PS@restored\@gobble
    \@ifundefined{PSTricksLoaded}{}{%
```

Für PSTricks < 2.0 nötig.

```
      \PSTricksOff
      \@ifundefined{c@lor@to@ps}{\def\c@lor@to@ps#1 #2\@@{}}{}}}%
```

PostScript-Ausgabe jetzt verhindern und später noch einmal.

```
\the\@temptokena
\expandafter\AtBeginDocument\expandafter{\the\@temptokena}%
\@ifundefined{PSTricksLoaded}{}
{%
```

Zum Parsen der Argumente von PSTricks' `\pst@object` laden wir preview im active-Modus, restaurieren aber die standardmäßigen Definitionen von `\output` und `\shipout`. `\pr@startbox` und `\pr@endbox` dienen hier nur dazu, um `\pst@object` wirkungslos zu machen und stattdessen die zugehörige Grafik aus der Containerdatei einzuladen. Derzeit werden nur maximal 3 optionale Parameter in runden Klammern am Ende von `\pst@object` unterstützt, was für viele, aber nicht für alle Fälle ausreichend ist.

```
  \newtoks\ppf@temptoken
  \ppf@temptoken\expandafter{\the\output}%
  \let\ppf@nofiles\nofiles \let\nofiles\relax
  \RequirePackage[active]{preview}[2005/01/29]%
  \let\shipout=\pr@shipout \let\nofiles\ppf@nofiles
  \output\expandafter{\the\ppf@temptoken}%
  \ppf@temptoken{}%
```

`\pr@startbox`, `\pr@endbox`: Gegenüber Originaldefinition vereinfacht.

```
  \long\def\pr@startbox#1#2{%
  \ifpr@outer
    \toks@{#2}%
    \edef\pr@cleanup{\the\toks@}%
    \setbox\@tempboxa\vbox\bgroup
    \everydisplay{}%
    \pr@outerfalse%
    \expandafter\@firstofone
  \else
    \expandafter\@gobble
  \fi{#1}}%
  \def\pr@endbox{%
  \egroup
  \setbox\@tempboxa\box\voidb@x
  \ppf@@getpicture
  \pr@cleanup}%
```

(Siehe auch identische Definition im DVI-Modus.)

```
  \AtBeginDocument{%
    \@ifundefined{pst@object}{}%
    {%
      \PreviewMacro[{{}*[]%
        ?\bgroup{#{#1}{{#1}}}{}%
        ?\bgroup{#{#1}{{#1}}}{}%
        ?({#{(#1)}{({#1})}}{}%
        ?({#{(#1)}{({#1})}}{}%
        ?({#{(#1)}{({#1})}}{}%
        }]{\pst@object}}%
    }%
  }%
```

Es werden auch die im DVI-Modus erlaubten Endungen von Grafikdateien benötigt.

```
  \begingroup
    \input{dvips.def}%
    \edef\x{\endgroup\def\noexpand\ppf@other@extensions{\Gin@extensions}}%
  \x
```

Dummy-Definition für die im DVI-Modus gültigen Dateitypen.

```
  \DeclareGraphicsRule{*}{eps}{*}{}%
  \define@key{Gin}{innerframe}[true]{%
    \lowercase{\Gin@boolkey{#1}}{innerframe}}%
  \define@key{Gin}{frame}[true]{%
    \lowercase{\Gin@boolkey{#1}}{frame}}%
  \define@key{Gin}{ignore}[true]{%
    \lowercase{\Gin@boolkey{#1}}{ignore}}%
  \define@key{Gin}{frame@@}{%
```

(Nur intern zu benutzen!)

```
    \edef\@tempa{\toks@{\noexpand\frame{\the\toks@}}}%
    \ifcase#1\relax
      \ifGin@innerframe\else\let\@tempa\relax\fi
    \or
      \ifGin@frame\else\let\@tempa\relax\fi
    \fi
    \@tempa
  }%
  \define@key{Gin}{showname}[true]{%
    \lowercase{\Gin@boolkey{#1}}{showname}}%
  \define@key{Gin}{namefont}{%
    \begingroup
      \@temptokena\expandafter{\ppf@namefont#1}%
      \edef\x{\endgroup\def\noexpand\ppf@namefont{\the\@temptokena}}%
    \x
  }%
  \newcommand*\ppf@filename{}%
  \newcommand*\ppf@namefont{\tiny\ttfamily}%
  \newcommand*\ppf@Gin@keys{}%
  \let\ppf@Gin@setfile\Gin@setfile
```

`\Gin@setfile` Realen Dateinamen und ggf. Seitenzahl zur späteren Verwendung merken.

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

`\Gin@ii` Auswertung der Optionen “frame”, “ignore” usw. sowie weiterer Spezialfälle.

```
  \def\Gin@ii[#1]#2{%
    \begingroup
```

Der Wert `\ifGin@innerframe` muss bereits vor Zeichnen des inneren Rahmens bekannt sein. Die Werte für `\ifGin@showname` und `\ppf@namefont` müssen auch nach Darstellung der Grafik verfügbar sein. Daher durch eine Gruppe geschützt vorher Auswertung der Optionen.

```
      \setkeys{Gin}{#1}%
      \@temptokena{#1}\def\@tempb{#2}%
```

Leerer Dateiname beim Aufruf von `\usepicture` aus.

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

Grafiken aus Containerdatei sind bereits skaliert usw. Nicht noch einmal, daher optionalen Parameter ignorieren.

```
        {%
          \ifx\@tempb\PDFcontainer
            \@temptokena{page=\GPT@page}%
          \fi
        }%
        {%
          \refstepcounter{pspicture}%
          \@temptokena{page=\the\c@pspicture}\def\@tempb{\PDFcontainer}%
        }%
      \fi
      \ifGin@ignore\else
```

"`frame@@=0`" = innerer Rahmen, "`frame@@=1`" = äußerer Rahmen.

```
        \edef\@tempa{\noexpand\ppf@Gin@ii[frame@@=0,\the\@temptokena,
          frame@@=1]{\@tempb}}%
        \@tempa
        \ifGin@showname
          \ppf@namefont
          \raisebox{-\ht\strutbox}[0pt][0pt]{\llap{\ppf@filename}}%
          \gdef\ppf@filename{}%
        \fi
      \fi
    \endgroup
  }%

  \IfFileExists{\PDFcontainer}%
  {%
```

`\ppf@container@max` Die Anzahl der in der Containerdatei enthaltenen Seiten.

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

    \AtEndDocument{%
      \ifnum\c@pspicture>\z@
```

Warnung ist nur sinnvoll, wenn überhaupt Grafiken benötigt wurden.

```
        \ifnum\c@pspicture=\ppf@container@max\else
          \PackageWarningNoLine{pst-pdf}{%
            `\PDFcontainer' contains \ppf@container@max\space pages
            \MessageBreak but \the\c@pspicture\space pages are requested:
            \MessageBreak File `\PDFcontainer' is no more valid!
            \MessageBreak Recreate it
          }%
        \fi
      \fi
    }%
  }%
  {%
    \def\ppf@container@max{0}%
    \AtEndDocument{%
      \ifnum\c@pspicture>\z@
        \filename@parse{\PDFcontainer}%
        \PackageWarningNoLine{pst-pdf}{%
```

```
          File `\PDFcontainer' not found.\MessageBreak
          Use the following commands to create it:\MessageBreak
          ----------------------------------------------------
          \MessageBreak
          latex \jobname.tex\MessageBreak
          dvips -o \filename@base.ps \jobname.dvi\MessageBreak
          ps2pdf \filename@base.ps\MessageBreak
          ----------------------------------------------------
        }%
      \fi
    }%
}%
```

`\ppf@isnum` Ist Parameter `#1` numerisch, werden Anweisungen in `#2` sonst die in `#3` ausgeführt (siehe bibtopic.sty).

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

`postscript` `pspicture` Beide Umgebungen ignorieren ihren Inhalt und laden stattdessen die zugehörige Grafik aus der Containerdatei. Auf den Wert des dabei benutzten Zählers (`pspicture`) kann per `\label`/`\ref` zugegriffen werden.

`psmatrix`

```
\newcommand*\ppf@set@mode{}%
\newcommand*\ppf@test@mmode{%
\ifmmode
  \ifinner
    \let\ppf@set@mode=$%
  \else
    \def\ppf@set@mode{$$}%
  \fi
\else
  \let\ppf@set@mode=\@empty
\fi
}
\newenvironment{postscript}[1][]
{%
  \ppf@test@mmode
  \gdef\ppf@Gin@keys{}%
  \def\@tempa{postscript}\ifx\@tempa\@currenvir\gdef\ppf@Gin@keys{#1}\fi
```

Innerhalb der Umgebung ist das Parsen der Argumente von `\pst@object` unnötig, daher wieder Originaldefinition verwenden.

```
  \expandafter\let\expandafter\pst@object
    \csname pr@\string\pst@object\endcsname
  \pr@outerfalse
```

Nötig für `\psmatrix`.

```
  \@makeother\&%
  \def\Gin@ii[##1]##2{}\setbox\@tempboxa=\vbox\bgroup
  \ppf@set@mode
}%
{\ppf@set@mode\egroup\aftergroup\ppf@@getpicture}%
\AtBeginDocument{%
```

```
    \@ifundefined{PSTricksLoaded}{}%
    {%
      \iffalse
        \PreviewEnvironment{pspicture}% Warum geht's nicht?
        \g@addto@macro\pspicture{%
          %%\pr@outerfalse% nötig, oder sowieso schon?
          \@makeother\&% nötig?
          \def\Gin@ii[#1]#2{}%
        }%
        \g@addto@macro\endpspicture{\ppf@@getpicture}%
      \else
        \def\pst@@@picture[#1](#2,#3)(#4,#5){\postscript}%
        \def\endpspicture{\endpostscript\endgroup}%
      \fi
      \@ifundefined{psmatrix}{}%
      {\let\psmatrix=\postscript\let\endpsmatrix=\endpostscript}%
    }%
  }%
```

`\savepicture` Speichert die Nummer der aktuellen Grafik in einem Makro mit Namen `\ppf@@@#1`.

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

`\usepicture` Fügt Grafik mit symbolischem Namen `#2` ein. Der Name muss vorher mit `\savepicture{`⟨*Name*⟩`}` vereinbart worden sein. Statt des Namens kann auch eine Zahl angegeben werden, die dann direkt eine Grafik aus der Containerdatei adressiert. Der optionale Parameter `#1` entspricht dem bei `\includegraphics`.

```
  \renewcommand*\usepicture[2][]{%
    \@ifundefined{ppf@@@#2}%
    {%
      \ppf@isnum{#2}%
      {\ppf@getpicture{#1}{#2}}%
      {\@latex@error{picture '#2' undefined}\@ehc}%
    }%
    {%
      \begingroup
        \def\Ginclude@graphics##1{%
          \xdef\ppf@filename{#2}%
          \setbox\z@\hbox{\pdfrefximage\@nameuse{ppf@@@#2}}%
          \Gin@nat@height\ht\z@ \Gin@nat@width\wd\z@
          \def\Gin@llx{0} \let\Gin@lly\Gin@llx
          \Gin@defaultbp\Gin@urx{\Gin@nat@width}%
          \Gin@defaultbp\Gin@ury{\Gin@nat@height}%
          \Gin@bboxtrue\Gin@viewport@code
          \Gin@nat@height\Gin@ury bp%
          \advance\Gin@nat@height-\Gin@lly bp%
          \Gin@nat@width\Gin@urx bp%
          \advance\Gin@nat@width-\Gin@llx bp%
          \Gin@req@sizes
          \ht\z@\Gin@req@height \wd\z@\Gin@req@width
          \leavevmode\box\z@}%
        \define@key{Gin}{type}{}%
        \includegraphics[scale=1,#1]{}%
```

```
      \endgroup
    }}%
```

`\ppf@getpicture` Fügt die Seite (Grafik) mit Nummer `#2` aus der Containerdatei ein. Parameter `#1`: Optionen wie bei `\includegraphics`.

```
  \newcommand*\ppf@getpicture[2]{%
    \@tempcnta=#2\relax%
    \ifnum\@tempcnta>\ppf@container@max
      \PackageWarningNoLine{pst-pdf}{%
        pspicture No. \the\@tempcnta\space undefined}%
    \else
      \includegraphics[draft=\ppf@draft,#1,page=\the\@tempcnta]%
        {\PDFcontainer}%
    \fi
    \gdef\ppf@Gin@keys{}}%
```

`\ppf@@getpicture` Fügt die nächste Seite (Grafik) aus der Containerdatei ein.

```
  \newcommand*\ppf@@getpicture{%
    \ifpr@outer
      \refstepcounter{pspicture}%
      \expandafter\ppf@getpicture\expandafter{\ppf@Gin@keys}%
      {\the\c@pspicture}%
    \fi}%

\else
```

3.5 Inactive Mode

Es werden nur die Pakete pstricks und graphicx geladen – keine weitere Einflussnahme. Die Paketoption "`inactive`" sowie der VTEX-Compiler erzwingen diesen Modus.

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

\InputIfFileExists{pst-pdf.cfg}{%
  \PackageInfo{pst-pdf}{Local config file pst-pdf.cfg used}}{}
⟨/package⟩
```