The pdfrender package

Heiko Oberdiek*

2019/12/29 v1.6

Abstract

The PDF format has some graphics parameter like line width or text rendering mode. This package provides an interface for setting these parameters.

Contents

1 Documentation **2**
1.1 Usage . . . 2
1.2 Macros . . . 2
1.3 Parameters . . . 2
1.3.1 Details . . . 3
1.4 Color stack . . . 4

2 Implementation **4**
2.1 Look for pdfTeX, its mode and features . . . 6
2.2 Enable color support of LaTeX . . . 7
2.3 Hook into `\normalcolor` . . . 8
2.4 Declare and setup parameters . . . 13
2.5 Fill and stroke color support . . . 14

3 Installation **18**
3.1 Download . . . 18
3.2 Bundle installation . . . 18
3.3 Package installation . . . 18
3.4 Refresh file name databases . . . 18
3.5 Some details for the interested . . . 19

4 Acknowledgement **19**

5 References **19**

6 History **19**
[2010/01/26 v1.0] . . . 19
[2010/01/27 v1.1] . . . 20
[2010/01/28 v1.2] . . . 20
[2016/05/14 v1.3] . . . 20
[2016/05/17 v1.4] . . . 20
[2018/11/01 v1.5] . . . 20
[2019/12/29 v1.6] . . . 20

7 Index **20**

*Please report any issues at https://github.com/ho-tex/oberdiek/issues

1 Documentation

This package pdfrender defines an interface for PDF specific parameters that affects the rendering of graphics or text. The interface and its implementation uses the same technique as package color for color settings. Therefore this package is loaded to enable LaTeX's color interface.

At different places LaTeX uses `\normalcolor` to avoid that header, footer or floats are print in the current color of the main text. `\setgroup@color` is used to start a save box with the color that is set at box saving time. Package pdfrender extends these macros to add its own hooks of its parameters. Therefore LaTeX3 should generalize LaTeX 2$_\varepsilon$'s color interface.

1.1 Usage

In LaTeX the package is loaded as normal package. Options are not defined for this package.

```
\usepackage{pdfrender}
```

This package can also be used in plain TeX and even iniTeX:

```
input pdfrender.sty
```

1.2 Macros

`\pdfrender` {⟨*key value list*⟩}

The first parameter ⟨*key value list*⟩ contains a list of parameter settings. The key entry is the parameter name. The macro works like `\color` (without optional argument) for color setting.

`\textpdfrender` {⟨*key value list*⟩} {⟨*text*⟩}

In the same way as `\pdfrender` the first argument specifies the parameters that should be set. This parameter setting affects ⟨*text*⟩ only. Basically it works the same way as `\textcolor` (without optional argument).

1.3 Parameters

The following table shows an overview for the supported parameters and values:

Parameter	**Value**	**Alias**
TextRenderingMode	0	`Fill`
	1	`Stroke`
	2	`FillStroke`
	3	`Invisible`
	4	`FillClip`
	5	`StrokeClip`
	6	`FillStrokeClip`
	7	`Clip`
LineWidth	*positive number, unit is bp*	*TeX dimen*

Parameter	Value	Alias
LineCapStyle	0	Butt
	1	Round
	2	ProjectingSquare
LineJoinStyle	0	Miter
	1	Round
	2	Bevel
MiterLimit	*positive number*	
Flatness	*number between 0 and 100*	
LineDashPattern	*numbers in square brackets, followed by number, units are bp*	
RenderingIntent	AbsoluteColorimetric	
	RelativeColorimetric	
	Saturation	
	Perceptual	
FillColor		*color specification*
StrokeColor		*color specification*

1.3.1 Details

The description and specification of these parameters are available in the PDF specification [1]. Therefore they are not repeated here.

Value: The values in the second column lists or describe the values that are specified by the PDF specification.

Alias: Instead of magic numbers the package also defines some aliases that can be given as value. Example: `LineCapStyle=Round` has the same effect as `LineCapStyle=1`.

Number: The term *number* means an integer or real number. The real number is given as plain decimal number without exponent. The decimal separator is a period. At least one digit must be present.

LineWidth: As alias a TEX dimen specification can be given. This includes explicit specifications with number and unit, e.g. `LineWidth=0.5pt`. Also LATEX length registers may be used. If ε-TEX's `\dimexpr` is available, then it is automatically added. However package calc is not supported.

FillColor, StrokeColor: Package color or xcolor must be loaded before these options can be used (since version 1.2). LATEX's color support sets both colors at the same time to the same value. However parameter TextRenderingMode offers the value `FillStroke` that makes only sense, if the two color types can be set separately. If one of the options FillColor or StrokeColor is specified, then also the color is set. For compatibility with the LATEX color packages (color or xcolor), always both colors must be set. Thus if one of them is not specified, it is taken from the current color.

Both options FillColor and StrokeColor expect a LATEX color specification as value. Also the optional color model argument is supported. Example:

```
FillColor=yellow,
StrokeColor=[cmyk]{1,.5,0,0}
```

1.4 Color stack

If the pdfTEX version provides color stacks, then each parameter is assigned a page based color stack. The assignment of a stack takes place, when its parameter is set the first time. This avoids the use of color stacks that are not needed.

2 Implementation

1 ⟨∗package⟩

Reload check, especially if the package is not used with LATEX.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3   \catcode13=5 % ^^M
4   \endlinechar=13 %
5   \catcode35=6 % #
6   \catcode39=12 % '
7   \catcode44=12 % ,
8   \catcode45=12 % -
9   \catcode46=12 % .
10  \catcode58=12 % :
11  \catcode64=11 % @
12  \catcode123=1 % {
13  \catcode125=2 % }
14  \expandafter\let\expandafter\x\csname ver@pdfrender.sty\endcsname
15  \ifx\x\relax % plain-TeX, first loading
16  \else
17    \def\empty{}%
18    \ifx\x\empty % LaTeX, first loading,
19      % variable is initialized, but \ProvidesPackage not yet seen
20    \else
21      \expandafter\ifx\csname PackageInfo\endcsname\relax
22        \def\x#1#2{%
23          \immediate\write-1{Package #1 Info: #2.}%
24        }%
25      \else
26        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27      \fi
28      \x{pdfrender}{The package is already loaded}%
29      \aftergroup\endinput
30    \fi
31  \fi
32 \endgroup%
```

Package identification:

```
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34   \catcode13=5 % ^^M
35   \endlinechar=13 %
36   \catcode35=6 % #
37   \catcode39=12 % '
38   \catcode40=12 % (
39   \catcode41=12 % )
40   \catcode44=12 % ,
41   \catcode45=12 % -
42   \catcode46=12 % .
43   \catcode47=12 % /
44   \catcode58=12 % :
45   \catcode64=11 % @
46   \catcode91=12 % [
47   \catcode93=12 % ]
48   \catcode123=1 % {
49   \catcode125=2 % }
50   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51     \def\x#1#2#3[#4]{\endgroup
```

```
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@pdfrender.sty\endcsname
\ProvidesPackage{pdfrender}%
  [2019/12/29 v1.6 Access to some PDF graphics parameters (HO)]%
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode123=1 % {
  \catcode125=2 % }
  \catcode64=11 % @
  \def\x{\endgroup
    \expandafter\edef\csname PdfRender@AtEnd\endcsname{%
      \endlinechar=\the\endlinechar\relax
      \catcode13=\the\catcode13\relax
      \catcode32=\the\catcode32\relax
      \catcode35=\the\catcode35\relax
      \catcode61=\the\catcode61\relax
      \catcode64=\the\catcode64\relax
      \catcode123=\the\catcode123\relax
      \catcode125=\the\catcode125\relax
    }%
  }%
\x\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\PdfRender@AtEnd{%
    \PdfRender@AtEnd
    \catcode#1=\the\catcode#1\relax
  }%
  \catcode#1=#2\relax
}
\TMP@EnsureCode{10}{12}% ^^J
\TMP@EnsureCode{36}{3}% $
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{12}% )
\TMP@EnsureCode{42}{12}% *
\TMP@EnsureCode{43}{12}% +
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{58}{12}% :
\TMP@EnsureCode{59}{12}% ;
```

```
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{63}{12}% ?
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{94}{7}% ^ (superscript)
\TMP@EnsureCode{96}{12}% `
\TMP@EnsureCode{124}{12}% |
\def\PdfRender@AtEndHook{}
\expandafter\def\expandafter\PdfRender@AtEnd\expandafter{%
  \expandafter\PdfRender@AtEndHook
  \PdfRender@AtEnd
  \endinput
}
```

2.1 Look for pdfTEX, its mode and features

\PdfRender@newif

```
\def\PdfRender@newif#1{%
  \expandafter\edef\csname PdfRender@#1true\endcsname{%
    \let
    \expandafter\noexpand\csname ifPdfRender@#1\endcsname
    \noexpand\iftrue
  }%
  \expandafter\edef\csname PdfRender@#1false\endcsname{%
    \let
    \expandafter\noexpand\csname ifPdfRender@#1\endcsname
    \noexpand\iffalse
  }%
  \csname PdfRender@#1false\endcsname
}
```

\ifPdfRender@Stack

```
\PdfRender@newif{Stack}
```

\ifPdfRender@Match

```
\PdfRender@newif{Match}
```

\PdfRender@RequirePackage

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \def\PdfRender@RequirePackage#1[#2]{%
    \expandafter\def\expandafter\PdfRender@AtEndHook\expandafter{%
      \PdfRender@AtEndHook
      \ltx@ifpackagelater{#1}{#2}{}{%
        \@PackageWarningNoLine{pdfrender}{%
          You have requested version\MessageBreak
          `#2' of package `#1',\MessageBreak
          but only version\MessageBreak
          `\csname ver@#1.\ltx@pkgextension\endcsname'\MessageBreak
          is available%
        }%
      }%
    }%
    \input #1.sty\relax
  }%
\else
  \let\PdfRender@RequirePackage\RequirePackage
\fi
```

Luatex compatibility

```
\ifx\pdfextension\@undefined\else
```

```
  \def\pdfcolorstackinit {\pdffeedback colorstackinit}
  \protected\def\pdfcolorstack     {\pdfextension colorstack}
  \protected\def\pdfliteral        {\pdfextension literal}
\fi
\PdfRender@RequirePackage{iftex}[2019/11/07]
\PdfRender@RequirePackage{infwarerr}[2007/09/09]
\PdfRender@RequirePackage{ltxcmds}[2010/01/28]
\ifpdf
  \ltx@IfUndefined{pdfcolorstackinit}{%
    \@PackageWarning{pdfrender}{%
      Missing \string\pdfcolorstackinit
    }%
  }{%
    \PdfRender@Stacktrue
  }%
  \ltx@IfUndefined{pdfmatch}{%
    \@PackageInfoNoLine{pdfrender}{%
      \string\pdfmatch\ltx@space not found. %
      Therefore the values\MessageBreak
      of some parameters are not validated%
    }%
  }{%
    \PdfRender@Matchtrue
  }%
\else
  \@PackageWarning{pdfrender}{%
    Missing pdfTeX in PDF mode%
  }%
  \ltx@IfUndefined{newcommand}{%
    \pdfrender
    \def\pdfrender#1{}%
    \textpdfrender
    \long\def\textpdfrender#1#2{#2}%
  }{%
    \pdfrender
    \newcommand*{\pdfrender}[1]{}%
    \textpdfrender
    \newcommand{\textpdfrender}[2]{#2}%
  }%
  \expandafter\PdfRender@AtEnd
\fi%
```

2.2 Enable color support of LaTeX

```
\ltx@ifpackageloaded{color}{}{%
  \def\color@setgroup{\begingroup\set@color}%
  \let\color@begingroup\begingroup
  \def\color@endgroup{\endgraf\endgroup}%
  \def\color@hbox{\hbox\bgroup\color@begingroup}%
  \def\color@vbox{\vbox\bgroup\color@begingroup}%
  \def\color@endbox{\color@endgroup\egroup}%
  \ltx@ifundefined{bgroup}{%
    \let\bgroup={\let\egroup=}%
  }{}%
  \ltx@ifundefined{endgraf}{%
```

```
    \let\endgraf=\par
  }{}%
}
```

2.3 Hook into \normalcolor

The problem is that packages color and xcolor each overwrite \normalcolor. For example, after the package loading order color, pdfrender and xcolor the patched version of \normalcolor is overwritten by package xcolor. Also using \AtBeginDocument for patching is not enough. If package hyperref is loaded later, it might load package color using \AtBeginDocument.

\PdfRender@NormalColorHook

```
\def\PdfRender@NormalColorHook{}
```

\PdfRender@ColorSetGroupHook

```
\def\PdfRender@ColorSetGroupHook{}
```

\PdfRender@TestBox

```
\def\PdfRender@TestBox#1{%
  \setbox0=\color@hbox#1\color@endbox
}
```

\PdfRender@PatchNormalColor

```
\def\PdfRender@PatchNormalColor{%
  \ltx@ifundefined{normalcolor}{%
    \gdef\normalcolor{\PdfRender@NormalColorHook}%
  }{%
    \begingroup
      \def\PdfRender@NormalColorHook{\let\PdfRender@temp=Y}%
      \PdfRender@TestBox{%
        \let\set@color\relax
        \normalcolor
        \ifx\PdfRender@temp Y%
        \else
          \ltx@GlobalAppendToMacro\normalcolor{%
            \PdfRender@NormalColorHook
          }%
        \fi
      }%
    \endgroup
  }%
  \ifx\@nodocument\relax
    \global\let\PdfRender@PatchNormalColor\relax
  \fi
}%
```

\PdfRender@PatchColorSetGroup

```
\def\PdfRender@PatchColorSetGroup{%
  \begingroup
    \def\PdfRender@ColorSetGroupHook{\let\PdfRender@temp=Y}%
    \PdfRender@TestBox{%
      \let\set@color\relax
      \color@setgroup\color@endgroup
      \ifx\PdfRender@temp Y%
      \else
        \ltx@GlobalAppendToMacro\color@setgroup{%
          \PdfRender@ColorSetGroupHook
        }%
      \fi
    }%
  \endgroup
```

```
  \ifx\@nodocument\relax
    \global\let\PdfRender@PatchColorSetGroup\relax
  \fi
}%
```

\PdfRender@PatchColor

```
\def\PdfRender@PatchColor{%
  \PdfRender@PatchNormalColor
  \PdfRender@PatchColorSetGroup
}

\PdfRender@PatchColor
\ltx@IfUndefined{AtBeginDocument}{}{%
  \AtBeginDocument{\PdfRender@PatchColor}%
}
```

\AfterPackage is provided by package scrlfile.

```
\ltx@IfUndefined{AfterPackage}{%
}{%
  \AfterPackage{color}{\PdfRender@PatchColor}%
  \AfterPackage{xcolor}{\PdfRender@PatchColor}%
  \AfterPackage{etoolbox}{%
    \AfterEndPreamble{\PdfRender@PatchColor}%
  }%
}%
```

\AfterEndPreamble is provided by package etoolbox.

```
\ltx@IfUndefined{AfterEndPreamble}{%
}{%
  \AfterEndPreamble{\PdfRender@PatchColor}%
}%

\PdfRender@RequirePackage{kvsetkeys}[2010/01/28]
```

\PdfRender@texorpdfstring

```
\def\PdfRender@texorpdfstring{%
  \ltx@IfUndefined{texorpdfstring}\ltx@firstoftwo\texorpdfstring
}
```

\pdfrender

```
\ltx@IfUndefined{DeclareRobustCommand}%
\ltx@firstoftwo\ltx@secondoftwo
{%
  \def\pdfrender#1%
}{%
  \newcommand{\pdfrender}{}%
  \DeclareRobustCommand*{\pdfrender}[1]%
}%
{%
  \PdfRender@texorpdfstring{%
    \PdfRender@PatchNormalColor
    \global\let\PdfRender@FillColor\ltx@empty
    \global\let\PdfRender@StrokeColor\ltx@empty
    \kvsetkeys{PDFRENDER}{#1}%
    \PdfRender@SetColor
  }{}%
}
```

\textpdfrender

```
\ltx@IfUndefined{DeclareRobustCommand}%
\ltx@firstoftwo\ltx@secondoftwo
{%
  \long\def\textpdfrender#1#2%
```

```
}{%
  \newcommand{\textpdfrender}{}%
  \DeclareRobustCommand{\textpdfrender}[2]%
}%
{%
  \PdfRender@texorpdfstring{%
    \begingroup
      \pdfrender{#1}%
      #2%
    \endgroup
  }{#2}%
}
```

\ifPdfRender@Values

```
\PdfRender@newif{Values}
```

\PdfRender@NewClassValues

```
\def\PdfRender@NewClassValues#1#2#3#4{%
  \PdfRender@Valuestrue
  \PdfRender@NewClass{#1}{#2}{#3}{#4}{}%
}
```

\PdfRender@NewClass

```
\def\PdfRender@NewClass#1#2#3#4#5{%
  \PdfRender@newif{Active#1}%
  \expandafter\def\csname PdfRender@Default#1\endcsname{#2}%
  \expandafter\let\csname PdfRender@Current#1\expandafter\endcsname
      \csname PdfRender@Default#1\endcsname
  \ifPdfRender@Stack
    \expandafter\edef\csname PdfRender@Init#1\endcsname{%
      \global\chardef
      \expandafter\noexpand\csname PdfRender@Stack#1\endcsname=%
          \noexpand\pdfcolorstackinit page direct{%
            \noexpand#3%
            \expandafter\noexpand\csname PdfRender@Default#1\endcsname
          }\relax
      \noexpand\@PackageInfo{pdfrender}{%
        New color stack `#1' = \noexpand\number
        \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
      }%
      \gdef\expandafter\noexpand\csname PdfRender@Init#1\endcsname{}%
    }%
    \expandafter\edef\csname PdfRender@Set#1\endcsname{%
      \expandafter\noexpand\csname PdfRender@Init#1\endcsname
      \noexpand\pdfcolorstack
      \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
      push{%
        #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
      }%
      \noexpand\aftergroup
      \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
    }%
    \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
      \expandafter\noexpand\csname PdfRender@Init#1\endcsname
      \noexpand\pdfcolorstack
      \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
      pop\relax
    }%
  \else
    \expandafter\edef\csname PdfRender@Set#1\endcsname{%
      \noexpand\pdfliteral direct{%
        #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
```

```
      }%
      \noexpand\aftergroup
      \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
    }%
    \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
      \noexpand\pdfliteral direct{%
        #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
      }%
    }%
  \fi
  \expandafter\edef\csname PdfRender@Normal#1\endcsname{%
    \let
    \expandafter\noexpand\csname PdfRender@Current#1\endcsname
    \expandafter\noexpand\csname PdfRender@Default#1\endcsname
    \noexpand\PdfRender@Set{#1}%
  }%
  \expandafter\ltx@GlobalAppendToMacro\expandafter\PdfRender@NormalColorHook
  \expandafter{%
    \csname PdfRender@Normal#1\endcsname
  }%
  \ltx@GlobalAppendToMacro\PdfRender@ColorSetGroupHook{%
    \PdfRender@Set{#1}%
  }%
  \ifPdfRender@Values
    \kv@parse@normalized{#4}{%
      \expandafter\let\csname PdfRender@#1@\kv@key\endcsname\kv@key
      \ifx\kv@value\relax
      \else
        \expandafter\let\csname PdfRender@#1@\kv@value\endcsname\kv@key
      \fi
      \ltx@gobbletwo
    }%
    \PdfRender@define@key{PDFRENDER}{#1}{%
      \global\csname PdfRender@Active#1true\endcsname
      \def\PdfRender@Current{##1}%
      \PdfRender@SetValidateValues{#1}%
    }%
    \PdfRender@Valuesfalse
  \else
    \PdfRender@define@key{PDFRENDER}{#1}{%
      \global\csname PdfRender@Active#1true\endcsname
      \expandafter\def\csname PdfRender@Current#1\endcsname{##1}%
      \ltx@IfUndefined{PdfRender@PostProcess#1}{%
      }{%
        \csname PdfRender@PostProcess#1\endcsname
      }%
      \PdfRender@SetValidate{#1}{#4}{#5}%
    }%
  \fi
}%
```

\PdfRender@define@key

```
\ltx@IfUndefined{define@key}{%
  \def\PdfRender@define@key#1#2{%
    \expandafter\def\csname KV@#1@#2\endcsname##1%
  }%
}{%
  \let\PdfRender@define@key\define@key
}
```

\PdfRender@Set

```
\def\PdfRender@Set#1{%
  \csname ifPdfRender@Active#1\endcsname
```

```
    \csname PdfRender@Set#1\expandafter\endcsname
  \fi
}
```

`\PdfRender@Reset`

```
\def\PdfRender@Reset#1{%
  \csname ifPdfRender@Active#1\endcsname
    \csname PdfRender@Reset#1\expandafter\endcsname
  \fi
}
```

`\PdfRender@ErrorInvalidValue`

```
\def\PdfRender@ErrorInvalidValue#1{%
  \PackageError{pdfrender}{%
    Ignoring parameter setting for `#1'\MessageBreak
    because of invalid value %
    `\csname PdfRender@Current#1\endcsname'%
  }\@ehc
  \expandafter\let\csname PdfRender@Current#1\endcsname\ltx@empty
}%
```

`\PdfRender@SetValidate`

```
\ifPdfRender@Match
  \def\PdfRender@SetValidate#1#2#3{%
    \ifnum\pdfmatch{^(#2)$}{\csname PdfRender@Current#1\endcsname}=1 %
      \csname PdfRender@Set#1\expandafter\endcsname
    \else
      \PdfRender@ErrorInvalidValue{#1}%
    \fi
  }%
\else
  \def\PdfRender@SetValidate#1#2#3{%
    \expandafter\let\expandafter\PdfRender@Current
    \csname PdfRender@Current#1\endcsname
    #3%
    \ifx\PdfRender@Current\@empty
      \PdfRender@ErrorInvalidValue{#1}%
    \else
      \csname PdfRender@Set#1\expandafter\endcsname
    \fi
  }%
\fi
```

`\PdfRender@SetValidateValues`

```
\def\PdfRender@SetValidateValues#1{%
  \ltx@IfUndefined{PdfRender@#1@\PdfRender@Current}{%
    \expandafter\let\csname PdfRender@Current#1\endcsname
                    \PdfRender@Current
    \PdfRender@ErrorInvalidValue{#1}%
  }{%
    \expandafter\edef\csname PdfRender@Current#1\endcsname{%
      \csname PdfRender@#1@\PdfRender@Current\endcsname
    }%
    \csname PdfRender@Set#1\endcsname
  }%
}
```

`\PdfRender@OpValue`

```
\def\PdfRender@OpValue#1#2{#2\ltx@space#1}%
```

`\PdfRender@OpName`

```
\def\PdfRender@OpName#1#2{/#2\ltx@space#1}%
```

2.4 Declare and setup parameters

```
\PdfRender@NewClassValues{TextRenderingMode}%
                         {0}%
                         {\PdfRender@OpValue{Tr}}{%
  0=Fill,%
  1=Stroke,%
  2=FillStroke,%
  3=Invisible,%
  4=FillClip,%
  5=StrokeClip,%
  6=FillStrokeClip,%
  7=Clip,%
}%
\PdfRender@NewClass{LineWidth}{1}{\PdfRender@OpValue{w}}{%
  [0-9]+\string\.?[0-9]*|\string\.[0-9]+%
}{}%
\ltx@IfUndefined{dimexpr}{%
  \def\PdfRender@dimexpr{}%
}{%
  \let\PdfRender@dimexpr\dimexpr
}
\def\PdfRender@PostProcessLineWidth{%
  \begingroup
  \afterassignment\PdfRender@@PostProcessLineWidth
  \dimen0=\PdfRender@dimexpr\PdfRender@CurrentLineWidth bp %
  \PdfRender@let\PdfRender@relax\PdfRender@relax
}
\let\PdfRender@let\let
\let\PdfRender@relax\relax
\def\PdfRender@@PostProcessLineWidth#1\PdfRender@let{%
  \ifx\\#1\\%
    \endgroup
  \else
    \dimen0=.996264\dimen0 % 72/72.27
    \edef\x{\endgroup
      \def\noexpand\PdfRender@CurrentLineWidth{%
        \strip@pt\dimen0%
      }%
    }%
    \expandafter\x
  \fi
}
\PdfRender@NewClassValues{LineCapStyle}{0}{\PdfRender@OpValue{J}}{%
  0=Butt,%
  1=Round,%
  2=ProjectingSquare,%
}%
\PdfRender@NewClassValues{LineJoinStyle}{0}{\PdfRender@OpValue{j}}{%
  0=Miter,%
  1=Round,%
  2=Bevel,%
}%
\PdfRender@NewClass{MiterLimit}{10}{\PdfRender@OpValue{M}}{%
  [0-9]*[1-9][0-9]*\string\.?[0-9]*|%
  [0-9]*\string\.?[0-9]*[1-9][0-9]*%
}{}%
\PdfRender@NewClass{Flatness}{0}{\PdfRender@OpValue{i}}{%
  100(\string\.0*)?|[0-9][0-9](\string\.[0-9]*)?|\string\.[0-9]+%
}{}%
\PdfRender@NewClass{LineDashPattern}{[]0}{\PdfRender@OpValue{d}}{%
  \string\[%
  ( ?([0-9]+\string\.?[0-9]*|\string\.[0-9]+) ?)*%
```

```
  \string\] ?%
  ([0-9]+\string\.?[0-9]*|\string\.[0-9]+)%
}{}%
\PdfRender@NewClassValues{RenderingIntent}%
                         {RelativeColorimetric}%
                         {\PdfRender@OpName{ri}}{%
  AbsoluteColorimetric,%
  RelativeColorimetric,%
  Saturation,%
  Perceptual,%
}%
```

2.5 Fill and stroke color support

```
\PdfRender@define@key{PDFRENDER}{FillColor}{%
  \begingroup
    \def\PdfRender@Color{#1}%
    \ifx\PdfRender@Color\ltx@empty
      \global\let\PdfRender@FillColor\ltx@empty
    \else
      \PdfRender@ColorAvailable{%
        \PdfRender@TestBox{%
          \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
          \PdfRender@GetFillColor
          \ifx\PdfRender@FillColor\ltx@empty
            \@PackageWarning{pdfrender}{%
              Cannot extract fill color\MessageBreak
              from value ‘#1’%
            }%
          \fi
        }%
      }%
    \fi
  \endgroup
}
\PdfRender@define@key{PDFRENDER}{StrokeColor}{%
  \begingroup
    \def\PdfRender@Color{#1}%
    \ifx\PdfRender@Color\ltx@empty
      \global\let\PdfRender@StrokeColor\ltx@empty
    \else
      \PdfRender@ColorAvailable{%
        \PdfRender@TestBox{%
          \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
          \PdfRender@GetStrokeColor
          \ifx\PdfRender@StrokeColor\ltx@empty
            \@PackageWarning{pdfrender}{%
              Cannot extract stroke color\MessageBreak
              from value ‘#1’%
            }%
          \fi
        }%
      }%
    \fi
  \endgroup
}
```

`\PdfRender@ColorAvailable`

```
\def\PdfRender@ColorAvailable{%
  \@ifundefined{set@color}{%
    \@PackageError{pdfrender}{%
      Ignoring color options, because neither\MessageBreak
      package ‘color’ nor package ‘xcolor’ is loaded%
    }\@ehc
```

```
    \global\let\PdfRender@ColorAvailable\ltx@gobble
  }{%
    \global\let\PdfRender@ColorAvailable\ltx@firstofone
  }%
  \PdfRender@ColorAvailable
}
```

\PdfRender@TryColor

```
\def\PdfRender@TryColor{%
  \@ifnextchar[\color\PdfRender@@TryColor
}
```

\PdfRender@@TryColor

```
\def\PdfRender@@TryColor#1\ltx@empty{%
  \expandafter\color\expandafter{\PdfRender@Color}%
}
```

\PdfRender@SetColor

```
\def\PdfRender@SetColor{%
  \chardef\PdfRender@NeedsCurrentColor=0 %
  \ifx\PdfRender@FillColor\ltx@empty
    \ifx\PdfRender@StrokeColor\ltx@empty
    \else
      \edef\PdfRender@CurrentColor{%
        \noexpand\PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
      }%
      \chardef\PdfRender@NeedsCurrentColor=1 %
    \fi
  \else
    \ifx\PdfRender@StrokeColor\ltx@empty
      \edef\PdfRender@CurrentColor{%
        \PdfRender@FillColor\ltx@space\noexpand\PdfRender@StrokeColor
      }%
      \chardef\PdfRender@NeedsCurrentColor=2 %
    \else
      \edef\current@color{%
        \PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
      }%
      \set@color
    \fi
  \fi
  \ifnum\PdfRender@NeedsCurrentColor=1 %
    \PdfRender@GetFillColor
    \ifx\PdfRender@FillColor\ltx@empty
      \@PackageWarning{pdfrender}{%
        Cannot extract current fill color%
      }%
    \else
      \edef\current@color{\PdfRender@CurrentColor}%
      \set@color
    \fi
  \else
    \ifnum\PdfRender@NeedsCurrentColor=2 %
      \PdfRender@GetStrokeColor
      \ifx\PdfRender@StrokeColor\ltx@empty
        \@PackageWarning{pdfrender}{%
          Cannot extract current stroke color%
        }%
      \else
        \edef\current@color{\PdfRender@CurrentColor}%
        \set@color
      \fi
```

```
    \fi
  \fi
}
```

\PdfRender@PatternFillColor

```
\edef\PdfRender@PatternFillColor{ % space
  (%
    [0-9\string\.]+ g|%
    [0-9\string\.]+ [0-9\string\.]+ [0-9\string\.]+ rg|%
    [0-9\string\.]+ [0-9\string\.]+ %
    [0-9\string\.]+ [0-9\string\.]+ k%
  ) % space
  (.*)$%
}
```

\PdfRender@PatternStrokeColor

```
\edef\PdfRender@PatternStrokeColor{ % space
  (%
    [0-9\string\.]+ G|%
    [0-9\string\.]+ [0-9\string\.]+ [0-9\string\.]+ RG|%
    [0-9\string\.]+ [0-9\string\.]+ %
    [0-9\string\.]+ [0-9\string\.]+ K%
  ) % space
  (.*)$%
}
```

\PdfRender@MatchPattern

```
\def\PdfRender@MatchPattern#1{%
  \ifnum\pdfmatch{\PdfRender@Pattern}{\PdfRender@String}=1 %
    \xdef#1{%
      \expandafter\strip@prefix\pdflastmatch 1%
    }%
    \edef\PdfRender@String{%
      \expandafter\strip@prefix\pdflastmatch 2%
    }%
    \ifx\PdfRender@String\ltx@empty
    \else
      \expandafter\expandafter\expandafter\PdfRender@MatchPattern
      \expandafter\expandafter\expandafter#1%
    \fi
  \fi
}
```

\PdfRender@GetFillColor

```
\def\PdfRender@GetFillColor{%
  \global\let\PdfRender@FillColor\ltx@empty
  \begingroup
    \ifPdfRender@Match
      \let\PdfRender@Pattern\PdfRender@PatternFillColor
      \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
      \PdfRender@MatchPattern\PdfRender@FillColor
    \else
      \edef\current@color{\current@color\ltx@space}%
      \let\PdfRender@OP\relax
      \PdfRender@FindOp{g}0%
      \PdfRender@FindOp{G}1%
      \PdfRender@FindOp{rg}0%
      \PdfRender@FindOp{RG}1%
      \PdfRender@FindOp{k}0%
      \PdfRender@FindOp{K}1%
      \PdfRender@FilterOp 0\PdfRender@FillColor
    \fi
```

```
  \endgroup
}
```

\PdfRender@GetStrokeColor

```
\def\PdfRender@GetStrokeColor{%
  \global\let\PdfRender@StrokeColor\ltx@empty
  \begingroup
    \ifPdfRender@Match
      \let\PdfRender@Pattern\PdfRender@PatternStrokeColor
      \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
      \PdfRender@MatchPattern\PdfRender@StrokeColor
    \else
      \edef\current@color{\current@color\ltx@space}%
      \let\PdfRender@OP\relax
      \PdfRender@FindOp{g}0%
      \PdfRender@FindOp{G}1%
      \PdfRender@FindOp{rg}0%
      \PdfRender@FindOp{RG}1%
      \PdfRender@FindOp{k}0%
      \PdfRender@FindOp{K}1%
      \PdfRender@FilterOp 1\PdfRender@StrokeColor
    \fi
  \endgroup
}

\ifPdfRender@Match
  \expandafter\PdfRender@AtEnd
\fi%
```

\PdfRender@FindOp

```
\def\PdfRender@FindOp#1#2{%
  \def\PdfRender@temp##1 #1 ##2\@nil{%
    ##1%
    \ifx\\##2\\%
      \expandafter\@gobble
    \else
      \PdfRender@OP{#1}#2%
      \expandafter\@firstofone
    \fi
    {%
      \PdfRender@temp##2\@nil
    }%
  }%
  \edef\current@color{%
    \@firstofone{\expandafter\PdfRender@temp\current@color} #1 \@nil
  }%
}
```

\PdfRender@FilterOp

```
\def\PdfRender@FilterOp#1#2{%
  \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
      \current@color\PdfRender@OP{}{}%
}
```

\PdfRender@@FilterOp

```
\def\PdfRender@@FilterOp#1#2#3\PdfRender@OP#4#5{%
  \ifx\\#4#5\\%
  \else
    \ifnum#1=#5 %
      \xdef#2{#3 #4}%
    \fi
    \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
```

```
753   \fi
754 }

755 \PdfRender@AtEnd%
756 ⟨/package⟩
```

3 Installation

3.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/pdfrender.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/pdfrender.pdf Documentation.

Bundle. All the packages of the bundle 'oberdiek' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TEX Files" (CTAN:pkg/tds). Directories with `texmf` in their name are usually organized this way.

3.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

3.3 Package installation

Unpacking. The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain TEX:

```
tex pdfrender.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
pdfrender.sty → tex/generic/oberdiek/pdfrender.sty
pdfrender.pdf → doc/latex/oberdiek/pdfrender.pdf
pdfrender.dtx → source/latex/oberdiek/pdfrender.dtx
```

If you have a `docstrip.cfg` that configures and enables docstrip's TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

3.4 Refresh file name databases

If your TEX distribution (TEX Live, MiKTEX, ...) relies on file name databases, you must refresh these. For example, TEX Live users run `texhash` or `mktexlsr`.

[1] CTAN:pkg/pdfrender

3.5 Some details for the interested

Unpacking with LaTeX. The `.dtx` chooses its action depending on the format:

plain TeX: Run docstrip and extract the files.

LaTeX: Generate the documentation.

If you insist on using LaTeX for docstrip (really, docstrip does not need LaTeX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{pdfrender.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLaTeX:

```
pdflatex pdfrender.dtx
makeindex -s gind.ist pdfrender.idx
pdflatex pdfrender.dtx
makeindex -s gind.ist pdfrender.idx
pdflatex pdfrender.dtx
```

4 Acknowledgement

Friedrich Vosberg asked in the newsgroup de.comp.text.tex for the font outline feature [2].

Gaius Pupus proposed the basic method using `\pdfliteral` in this thread [3].

Rolf Niepraschk added color support [4].

5 References

[1] Adobe Systems Incorporated. *PDF Reference – Adobe Portable Document format – Version 1.7*. 6th ed. 2006. URL: http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf.

[2] Friedrich Vosberg, *Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-22. URL: https://groups.google.com/group/de.comp.text.tex/msg/f442310ac8b2d506.

[3] Gaius Pupus, *Re: Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-23. URL: https://groups.google.com/group/de.comp.text.tex/msg/95d890d77ac47eb1.

[4] Rolf Niepraschk, *Re: Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-24. URL: https://groups.google.com/group/de.comp.text.tex/msg/4eb61a5879db54db.

6 History

[2010/01/26 v1.0]

- The first version.

[2010/01/27 v1.1]

- Macros `\pdfrender` and `\textpdfrender` are made robust.
- Color extraction rewritten for the case that `\pdfmatch` is not available. This fixes wrong color assigments in case of nesting.
- Color extraction of case `\pdfmatch` is fixed for the case that the color string contains several fill or several stroke operations.

[2010/01/28 v1.2]

- Dependency from package color is removed.
- Compatibility for plain TEX and even iniTEX added.

[2016/05/14 v1.3]

- Use package luatex85 for compatibility with new LuaTeX.

[2016/05/17 v1.4]

- Documentation updates.
- adjust luatex85 reference so that it works in plain TeX.

[2018/11/01 v1.5]

- Remove luatex85 dependency

[2019/12/29 v1.6]

- iftex package.

7 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

Symbols

`\.` 483, 522, 523, 526, 530, 532, 651, 652, 653, 654, 660, 661, 662, 663
`\@PackageError` 586
`\@PackageInfo` 335
`\@PackageInfoNoLine` 180
`\@PackageWarning` 173, 189, 553, 574, 628, 639
`\@PackageWarningNoLine` 149
`\@ehc` 433, 589
`\@empty` 449
`\@firstofone` 732, 739
`\@gobble` 729
`\@ifnextchar` 597
`\@ifundefined` 585
`\@nil` 726, 735, 739
`\@nodocument` 238, 256
`\@undefined` 58, 163
`\[` 529
`\\` 499, 728, 747
`\]` 531

A

`\afterassignment` 492
`\AfterEndPreamble` 273, 278
`\aftergroup` 29, 348, 362
`\AfterPackage` 270, 271, 272
`\AtBeginDocument` 266

C

`\catcode` 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99
`\chardef` 329, 603, 610, 617
`\color` 597, 600
`\color@begingroup` 203, 205, 206
`\color@endbox` 207, 218
`\color@endgroup` 204, 207, 247
`\color@hbox` 205, 218
`\color@setgroup` 202, 247, 250
`\color@vbox` 206

\csname 14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 324, 325, 326, 328, 330, 333, 337, 339, 341, 342, 344, 346, 349, 351, 352, 354, 358, 360, 363, 365, 367, 371, 373, 374, 379, 386, 389, 394, 401, 402, 405, 413, 419, 420, 424, 425, 432, 434, 438, 439, 447, 452, 458, 462, 463, 465
\current@color 619, 632, 643, 687, 690, 707, 710, 738, 739, 744

D

\DeclareRobustCommand 290, 307
\define@key 416
\dimen 493, 502, 505
\dimexpr 488

E

\empty 17, 18
\endcsname 14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 324, 325, 326, 328, 330, 333, 337, 339, 341, 342, 344, 346, 349, 351, 352, 354, 358, 360, 363, 365, 367, 371, 373, 374, 379, 386, 389, 394, 401, 402, 405, 413, 419, 420, 424, 425, 432, 434, 438, 439, 447, 452, 458, 462, 463, 465
\endgraf 204, 212
\endinput 29, 126
\endlinechar 4, 35, 71, 77, 89

G

\gdef 222, 339

H

\hbox 205

I

\iffalse 137
\ifnum 438, 625, 636, 668, 749
\ifpdf 171
\ifPdfRender@Match 142, 436, 685, 705, 722
\ifPdfRender@Stack 141, 327
\ifPdfRender@Values 317, 384
\iftrue 132
\ifx . 15, 18, 21, 50, 58, 61, 144, 163, 229, 238, 248, 256, 387, 449, 499, 545, 552, 566, 573, 604, 605, 613, 627, 638, 675, 728, 747
\immediate 23, 52
\input 158

K

\kv@key 386, 389
\kv@parse@normalized 385
\kv@value 387, 389
\kvsetkeys 297

L

\ltx@empty 295, 296, 434, 545, 546, 550, 552, 566, 567, 571, 573, 599, 604, 605, 613, 627, 638, 675, 683, 703
\ltx@firstofone 592
\ltx@firstoftwo 282, 285, 302
\ltx@GlobalAppendToMacro 231, 250, 377, 381
\ltx@gobble 590
\ltx@gobbletwo 391
\ltx@ifpackagelater 148
\ltx@ifpackageloaded 201
\ltx@IfUndefined 172, 179, 192, 265, 268, 276, 282, 284, 301, 403, 411, 457, 485
\ltx@ifundefined 208, 211, 221
\ltx@pkgextension 153
\ltx@secondoftwo 285, 302
\ltx@space 181, 468, 469, 608, 615, 620, 687, 690, 707, 710

M

\MessageBreak 150, 151, 152, 153, 182, 430, 554, 575, 587

N

\newcommand 196, 197, 289, 306
\normalcolor 222, 228, 231
\number 336

P

\PackageError 429
\PackageInfo 26
\par 212
\pdfcolorstack 165, 343, 353
\pdfcolorstackinit 164, 174, 331
\pdfextension 163, 165, 166
\pdffeedback 164
\pdflastmatch 670, 673
\pdfliteral 166, 359, 366
\pdfmatch 181, 438, 668
\pdfrender *2*, 193, 196, 284, 312
\PdfRender@@FilterOp 743, 746
\PdfRender@@PostProcessLineWidth 492, 498
\PdfRender@@TryColor 597, 599
\PdfRender@AtEnd 95, 96, 123, 125, 199, 723, 755
\PdfRender@AtEndHook 122, 124, 146, 147
\PdfRender@Color 544, 545, 550, 565, 566, 571, 600
\PdfRender@ColorAvailable 548, 569, 584
\PdfRender@ColorSetGroupHook 216, 244, 251, 381
\PdfRender@Current 395, 446, 449, 457, 459, 463
\PdfRender@CurrentColor 607, 614, 632, 643
\PdfRender@CurrentLineWidth 493, 504
\PdfRender@define@key 393, 400, 411, 542, 563

\PdfRender@dimexpr 486, 488, 493
\PdfRender@ErrorInvalidValue 428, 441, 450, 460
\PdfRender@FillColor 295, 546, 552, 604, 608, 615, 620, 627, 683, 688, 698
\PdfRender@FilterOp . . . 698, 718, 742
\PdfRender@FindOp 692, 693, 694, 695, 696, 697, 712, 713, 714, 715, 716, 717, 725
\PdfRender@GetFillColor 551, 626, 682
\PdfRender@GetStrokeColor 572, 637, 702
\PdfRender@let 494, 496, 498
\PdfRender@MatchPattern 667, 688, 708
\PdfRender@Matchtrue 186
\PdfRender@NeedsCurrentColor 603, 610, 617, 625, 636
\PdfRender@NewClass 320, 322, 482, 521, 525, 528
\PdfRender@NewClassValues 318, 470, 511, 516, 534
\PdfRender@newif 128, 141, 142, 317, 323
\PdfRender@NormalColorHook 215, 222, 225, 232, 377
\PdfRender@OP . 691, 711, 731, 744, 746
\PdfRender@OpName 469, 536
\PdfRender@OpValue 468, 472, 482, 511, 516, 521, 525, 528
\PdfRender@PatchColor 260, 264, 266, 270, 271, 273, 278
\PdfRender@PatchColorSetGroup 242, 262
\PdfRender@PatchNormalColor 220, 261, 294
\PdfRender@Pattern 668, 686, 706
\PdfRender@PatternFillColor 649, 686
\PdfRender@PatternStrokeColor 658, 706
\PdfRender@PostProcessLineWidth . 490
\PdfRender@relax 494, 497
\PdfRender@RequirePackage 143, 168, 169, 170, 280
\PdfRender@Reset 423
\PdfRender@Set 375, 382, 418
\PdfRender@SetColor 298, 602
\PdfRender@SetValidate 407, 436
\PdfRender@SetValidateValues 396, 456
\PdfRender@Stacktrue 177
\PdfRender@String 668, 672, 675, 687, 707
\PdfRender@StrokeColor 296, 567, 573, 605, 608, 613, 615, 620, 638, 703, 708, 718
\PdfRender@temp 225, 229, 244, 248, 726, 735, 739
\PdfRender@TestBox 217, 226, 245, 549, 570
\PdfRender@texorpdfstring 281, 293, 310
\PdfRender@TryColor . . . 550, 571, 596
\PdfRender@Valuesfalse 398
\PdfRender@Valuestrue 319
\protected 165, 166
\ProvidesPackage 19, 67

R

\RequirePackage 161

S

\set@color 202, 227, 246, 622, 633, 644
\setbox 218
\strip@prefix 670, 673
\strip@pt 505

T

\texorpdfstring 282
\textpdfrender *2*, 194, 197, 301
\the . . . 77, 78, 79, 80, 81, 82, 83, 84, 97
\TMP@EnsureCode 94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121

V

\vbox 206

W

\write 23, 52

X

\x 14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 503, 508