

# The `texpower` package\*

Stephan Lehmke  
`Stephan.Lehmke@cs.uni-dortmund.de`

April 9, 2005

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Disclaimer</b>	<b>2</b>
<b>3</b>	<b>Implementation</b>	<b>2</b>
3.1	Options and general setup . . . . .	2
3.1.1	General options . . . . .	2
3.1.2	Color options . . . . .	3
3.2	Color management, color emphasis and highlighting . . . . .	6
3.2.1	Color management kernel . . . . .	6
3.2.2	Commands for calculating new colors . . . . .	7
3.3	Color name and color set management . . . . .	11
3.3.1	Implementation of the <code>coloremph</code> option . . . . .	18
3.3.2	Implementation of the <code>colormath</code> option . . . . .	18
3.4	Structured rules, box and page backgrounds . . . . .	23
3.4.1	Structured rules . . . . .	23
3.4.2	Structured box backgrounds . . . . .	35
3.4.3	Structured page backgrounds . . . . .	38
3.4.4	Implementation of <code>\backgroundstyle</code> . . . . .	49
3.5	Panels . . . . .	56
3.5.1	Panel-specific user level commands . . . . .	57
3.5.2	Implementation of automatic dimension calculation . . . . .	59
3.5.3	Actually typeset panels . . . . .	62
3.6	Navigation helpers . . . . .	63
3.7	Set acrobat reader's page transition mode . . . . .	67
3.8	Set acrobat reader's automatic page advancing feature . . . . .	68
3.9	TeXPower kernel . . . . .	70
3.9.1	Overload <code>\shipout</code> . . . . .	70
3.9.2	The kernel functions to be executed at <code>\shipout</code> . . . . .	71

---

\*This document corresponds to `texpower` v0.2, dated 2005/04/08.

3.9.3	Implementation of ‘fixcolorstack’ option	73
3.9.4	Kernel functions for overloading <code>\output</code>	76
3.9.5	Kernel functions for re-inserting page contents	78
3.9.6	Implementation of <code>\pause</code>	79
3.9.7	Implementing <code>\stepwise</code> and all functions surrounding it	79
3.9.8	Command administration	80
3.9.9	Registers	80
3.9.10	Custom commands for displaying step contents	81
3.9.11	Custom commands for ‘hiding’ stepcontents ...	81
3.9.12	Displaying and hiding of step contents	83
3.9.13	Implementation of <code>\step</code> , <code>\switch</code> and relatives	87
3.9.14	Implementation of <code>\stepwise</code>	97
3.9.15	Implementation of the <code>fragilesteps</code> environment	103
3.9.16	Input system-specific settings	104

## 1 Introduction

LaTeX Package for creating ‘dynamic’ presentations.

The user documentation is found in `manual.tex` and the FAQ. Only the implementation documentation is covered in this document.

The TeXPower Bundle can be found at <http://texpower.sourceforge.net/>

## 2 Disclaimer

This is still work in progress.

During the subsequent error correction and extension of the functionality, the syntax and implementation of the macros are liable to change.

Even though we are using dtx-files, these are still not fully documented dtx-files.

## 3 Implementation

We need the programming tools provided by these packages.

```
1 \RequirePackage{ifthen}
2 \RequirePackage{calc}
3 \RequirePackage{keyval}
```

### 3.1 Options and general setup

#### 3.1.1 General options

The option `verbose` turns on some automatic messages.

```
4 \newboolean{verbose@TP}
5 \DeclareOption{verbose}{\setboolean{verbose@TP}{true}}
```

The (global) option `display` is respected and turns on the ‘dynamic’ features.

```
6 \provideboolean{display}
7 \DeclareOption{display}{\setboolean{display}{true}}
```

The option `printout` turns off the ‘dynamic’ features. Can be used to undo a default setting of `display`.

```
8 \DeclareOption{printout}{\setboolean{display}{false}}
```

The option `fixcolorstack` switches on a ‘color stack correction’ method which undoes damage to the driver’s color stack when “color push” and “color pop” specials are duplicated.

```
9 \newboolean{fixcolorstack@TP}
10 \DeclareOption{fixcolorstack}{\setboolean{fixcolorstack@TP}{true}}
```

The option `oldfiltering` reverts to the old (pre v0.2) aggressive/robust filtering of whatsits.

```
11 \newboolean{oldfiltering@TP}
12 \DeclareOption{oldfiltering}{\setboolean{oldfiltering@TP}{true}}
13 \newcommand{\oldfilteringon}{\setboolean{oldfiltering@TP}{true}}
14 \newcommand{\oldfilteringoff}{\setboolean{oldfiltering@TP}{false}}
```

The option `nineminutes` sets the page duration of every single page to a high value (of about nine minutes; this seems to be a hardcoded upper limit in acrobat 5; see below). This way, a setting in acrobat reader’s fullscreen dialogue is masked. Otherwise, pages without an explicit page duration setting don’t get any page duration setting at all, so they will follow the dialogue setting.

```
15 \newboolean{nineminutes@TP}
16 \DeclareOption{nineminutes}{\setboolean{nineminutes@TP}{true}}
```

### 3.1.2 Color options

The following switch indicates whether color management should be turned on at all.

```
17 \newboolean{TPcolor}
```

The option `coloremph` makes `\em` and `\emph` switch the text color instead of the font shape.

```
18 \newboolean{coloremph@TP}
19 \DeclareOption{coloremph}
20 {\setboolean{TPcolor}{true}\setboolean{coloremph@TP}{true}}
```

The option `colormath` makes math formulae be color highlighted.

```
21 \newboolean{colormath@TP}
22 \DeclareOption{colormath}
23 {\setboolean{TPcolor}{true}\setboolean{colormath@TP}{true}}
```

The option `colorhighlight` makes highlighting commands use colors.

```
24 \newboolean{colorhighlight@TP}
25 \DeclareOption{colorhighlight}
26 {\setboolean{TPcolor}{true}\setboolean{colorhighlight@TP}{true}}
```

The option `whitebackground` selects standard colors for white backgrounds.

```
27 \newboolean{whitebackground@TP}
28 \setboolean{whitebackground@TP}{true}% This is the default.
29 \DeclareOption{whitebackground}
30 {\setboolean{TPcolor}{true}\setboolean{whitebackground@TP}{true}}
```

The option `lightbackground` selects standard colors for light (but not white) backgrounds.

```
31 \newboolean{lightbackground@TP}
32 \DeclareOption{lightbackground}
33 {\setboolean{TPcolor}{true}\setboolean{lightbackground@TP}{true}}
```

The option `blackbackground` selects standard colors for black backgrounds.

```
34 \newboolean{blackbackground@TP}
35 \DeclareOption{blackbackground}
36 {\setboolean{TPcolor}{true}\setboolean{blackbackground@TP}{true}}
```

The option `darkbackground` selects standard colors for dark (but not black) backgrounds.

```
37 \newboolean{darkbackground@TP}
38 \DeclareOption{darkbackground}
39 {\setboolean{TPcolor}{true}\setboolean{darkbackground@TP}{true}}
```

Load the config file with default options if file exists.

```
40 \InputIfFileExists{tpoptions.cfg}{}{}
```

Process options.

```
41 \ProcessOptions
```

```
42
```

```
43 \ifthenelse{\boolean{display}}{
```

```
44 {\PackageInfo{texpower}{Producing display version. Dynamic features activated.}}
```

```
45 {\PackageInfo{texpower}{Producing printout version. Dynamic features inactive.}}
```

General option-driven initialization.

If the verbose option is set, we give a lot of context information when an error is raised.

```
46 \ifthenelse{\boolean{verbose@TP}}{\setcounter{errorcontextlines}{10000}}{}
```

Driver-specific defaults.

We provide a switch which (hopefully) allows to distinguish whether postscript specials (as used by PSTricks) can be used safely or not.

```
47 \newboolean{psspecialsallowed}
```

```
48 \setboolean{psspecialsallowed}{true} % optimistic default
```

The switch `\ifpdf` is to determine whether pdfLaTeX is being run and outputting pdf, using Heiko Oberdiek's faultproof pdf detector:

```
49 \@ifundefined{pdftrue}{
```

```
50 \IfFileExists{ifpdf.sty}{\RequirePackage{ifpdf}}{%
```

```
51 \expandafter\newif\csname ifpdf\endcsname
```

```
52 \ifx\pdfoutput\undefined
```

```
53 \else
```

```
54 \ifx\pdfoutput\relax
```

```

55   \else
56   \ifcase\pdfoutput
57   \else
58   \pdftrue
59   \fi
60   \fi
61   \fi
62 }
63 }{% \ifpdf is defined - nothing to do
64 }
65
66 \ifpdf\setboolean{psspecialsallowed}{false}\fi

```

Class-specific defaults.

The following switches centering of slides off for the slides document class because this would disturb dynamic building of slides.

```

67 \@ifclassloaded{slides}%
68 {%
69   \ifthenelse{\boolean{display}}{%
70     {\let\@topfil\relax}%
71   }%
72 }
73 {}

```

Some registers and macros for general use throughout texpower.sty.

```

74 \newcounter{tmpcnta@TP}
75 \newcounter{tmpcntb@TP}
76
77 \newlength{\tempdima@TP}
78 \newlength{\tempdimb@TP}
79
80 \newbox\tempbox@TP
81
82 \newboolean{carryon@TP}

```

These are needed for calculating the size of the page background box.

```

83 \newcommand{\TPpagewidth}{\strip@pt\paperwidth truept}
84 \newcommand{\TPpageheight}{\strip@pt\paperheight truept}
85 % \AtBeginDocument
86 % {%
87 %   \edef\TPpagewidth{\strip@pt\paperwidth truept}%
88 %   \edef\TPpageheight{\strip@pt\paperheight truept}%
89 % }

```

`\mkfactor` `\mkfactor{<cs>}{<exp>}` is a helper command for automatically generating the fixed point numbers between 0 and 1 which are employed by the color calculation commands. `<exp>` can be anything which can stand behind `*` in calc (for instance: `\value{counter}/\value{maxcounter}` or `\ratio` or whatever). `<cs>` should be a valid macro name. `<exp>` is converted into a fixed-point representation which is then assigned to `<cs>`.

```

90 \newcommand{\mkfactor}[2]%
91 {\setlength{\tempdima@TP}{1pt*#2}\edef#1{\strip@pt\tempdima@TP}}

```

Make a string representation of a length expression.

```

92 \newcommand{\mklength@TP}[2]
93 {\setlength{\tempdima@TP}{#2}\edef#1{\the\tempdima@TP}}
94
95 \newcommand{\mklength}{}
96 \let\mklength\mklength@TP

```

## 3.2 Color management, color emphasis and highlighting

Initialization.

If we are to use colors, we need the color package.

```

97 \ifthenelse{\boolean{TPcolor}}{\RequirePackage{color}}{}%

```

### 3.2.1 Color management kernel

Only load the kernel if TeXPower’s color management is active.

```

98 \ifthenelse{\boolean{TPcolor}}{% Yes.

```

We need a hook which can be defined otherwise to turn off colors.

```

99 \let\setcolor@TP=\color%

```

Overload `\definecolor` to store a ‘driver-independent’ copy of the color definition for later use by `\colorbetween` and relatives.

```

100 \let\o@definecolor@TP=\definecolor%
101 \def\definecolor#1#2#3%
102 {%
103   \o@definecolor@TP{#1}{#2}{#3}%
104   \expandafter\edef\csname colordef@TP@#1\endcsname%
105   {\csname processcolor@TP@#2\endcsname{#3}}%
106   }%

```

Repeat `color.sty`’s standard color definitions to make the original definitions available to TeXPower. Begin excerpt from `color.sty`:

```

107 \@ifundefined{c@lor@namefile}{}{\input{\c@lor@namefile}}
108
109 \ifx\color@gray\undefined
110   \ifx\color@rgb\undefined
111     \else
112       \definecolor{black}{rgb}{0,0,0}
113       \definecolor{white}{rgb}{1,1,1}
114     \fi
115   \else
116     \definecolor{black}{gray}{0}
117     \definecolor{white}{gray}{1}
118   \fi
119 \ifx\color@rgb\undefined\else
120   \definecolor{red}{rgb}{1,0,0}

```

```

121 \definecolor{green}{rgb}{0,1,0}
122 \definecolor{blue}{rgb}{0,0,1}
123 \fi
124 \ifx\color@cmk\undefined\else
125 \definecolor{cyan}{cmk}{1,0,0,0}
126 \definecolor{magenta}{cmk}{0,1,0,0}
127 \definecolor{yellow}{cmk}{0,0,1,0}
128 \fi

```

End excerpt from color.sty.

### 3.2.2 Commands for calculating new colors

`\interpolate@TP` Calculates the weighted average between two fixed point values.

```

129 \newcommand{\interpolate@TP}[3]%
130 {%
131 \setlength{\tempdima@TP}{1pt-#1pt}% Calculate the second factor for the weigh
132 \edef\secondfactor@TP{\strip@pt\tempdima@TP}%
133 \setlength{\tempdima@TP}{#2pt*\real{#1}+#3pt*\real{\secondfactor@TP}}% Calculate the weight
134 \ifthenelse{\lengthtest{\tempdima@TP<0pt}}% Bound the result to the interval [0,1] (j
135 {\setlength{\tempdima@TP}{0pt}}% factor was not from [0,1]).
136 {\ifthenelse{\lengthtest{\tempdima@TP>1pt}}{\setlength{\tempdima@TP}{1pt}}{}}%
137 \edef\result@TP{\strip@pt\tempdima@TP}%
138 }

```

`\interpolate@three@TP` Interpolates a three-piece color value.

```

139 \def\interpolate@three@TP#1,#2,#3;#4,#5,#6;#7%
140 {%
141 \interpolate@TP{#7}{#1}{#4}% First intermediary value.
142 \edef\newcolordef@TP{\result@TP,}% Store first value.
143 \interpolate@TP{#7}{#2}{#5}% Second intermediary value.
144 \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
145 \interpolate@TP{#7}{#3}{#6}% Third intermediary value.
146 \edef\newcolordef@TP{\newcolordef@TP\result@TP}% Store third value.
147 }

```

`\interpolate@four@TP` Interpolates a four-piece color value.

```

148 \def\interpolate@four@TP#1,#2,#3,#4;#5,#6,#7,#8;#9%
149 {%
150 \interpolate@TP{#9}{#1}{#5}% First intermediary value.
151 \edef\newcolordef@TP{\result@TP,}% Store first value.
152 \interpolate@TP{#9}{#2}{#6}% Second intermediary value.
153 \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
154 \interpolate@TP{#9}{#3}{#7}% Third intermediary value.
155 \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store third value.
156 \interpolate@TP{#9}{#4}{#8}% Fourth intermediary value.
157 \edef\newcolordef@TP{\newcolordef@TP\result@TP}% Store fourth value.
158 }

```

`\convert@cmkvalue@rgbvalue@TP` Converts one color value from CMYK to rgb.

```

159 \def\convert@cmykvalue@rgbvalue@TP#1#2%
160 {%
161   \setlength{\tempdima@TP}{1pt-#1pt-#2pt}%
162   \ifthenelse{\lengthtest{\tempdima@TP<0pt}}{\setlength{\tempdima@TP}{0pt}}{}%
163   \edef\result@TP{\strip@pt\tempdima@TP}%
164   }%

```

`\convert@cmyk@rgb@TP` Converts CMYK color to rgb.

```

165 \def\convert@cmyk@rgb@TP#1,#2,#3,#4;%
166 {%
167   \convert@cmykvalue@rgbvalue@TP{#1}{#4}%
168   \edef\newcolordef@TP{\result@TP,}% Store first value.
169   \convert@cmykvalue@rgbvalue@TP{#2}{#4}%
170   \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
171   \convert@cmykvalue@rgbvalue@TP{#3}{#4}%
172   \edef\newcolordef@TP{\newcolordef@TP\result@TP}% Store third value.
173   }

```

`\convert@RGBvalue@rgbvalue@TP` Converts one color value from RGB to rgb.

```

174 \def\convert@RGBvalue@rgbvalue@TP#1%
175 {%
176   \setlength{\tempdima@TP}{#1pt/255}%
177   \edef\result@TP{\strip@pt\tempdima@TP}%
178   }%

```

`\convert@RGB@rgb@TP` Converts RGB color to rgb.

```

179 \def\convert@RGB@rgb@TP#1,#2,#3;%
180 {%
181   \convert@RGBvalue@rgbvalue@TP{#1}%
182   \edef\newcolordef@TP{\result@TP,}% Store first value.
183   \convert@RGBvalue@rgbvalue@TP{#2}%
184   \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
185   \convert@RGBvalue@rgbvalue@TP{#3}%
186   \edef\newcolordef@TP{\newcolordef@TP\result@TP}% Store third value.
187   }

```

`\colorbetween[<factor>]{<target>}{<source1>}{<source2>}` calculates a ‘weighted average’ between two colors. `<source1>` and `<source2>` are the names of the two colors. `<factor>` (default: 0.5) is a fixed-point number between 0 and 1 giving the ‘weight’ for the interpolation between `<source1>` and `<source2>`. `<target>` is the name to be given to the resulting mixed color. If `<factor>` is 1, then `<target>` will be identical to `<source1>` (up to color model conversions, see below), if `<factor>` is 0, then `<target>` will be identical to `<source2>`, if `<factor>` is 0.5, then `<target>` will be exactly in the middle between `<source1>` and `<source2>`.

`\colorbetween` supports the following color models: rgb, RGB, gray, cmyk, hsb. If both colors are of the same model, the resulting color is also of the respective model. If `<source1>` and `<source2>` are from different models, then `<target>` will always be an rgb color. The only exception is the hsb color model:





```

235     \fi
236   }%
237   \newcommand{\processcolor@TP@RGB}[3]%           What if the first color is an RGB color?
238   {%
239     \convert@RGB@rgb@TP##1;%                     Convert to rgb ...
240     \expandafter\processcolor@TP@rgb\expandafter{\newcolordef@TP}{##2}{##3}% ... and contin
241   }%
242   \newcommand{\processcolor@TP@hsb}[3]%          What if the first color is an hsb color?
243   {%
244     \ifx##2\processcolor@TP@hsb%                Are both colors hsb colors?
245       \interpolate@three@TP##1;##3;##1;%        Calculate interpolated values.
246       \edef\newcolordef@TP{{\hsb}\newcolordef@TP}}% Store the result
247     \else
248       \PackageError{texpower}{Don't know how to convert an hsb color!}
249     \fi
250   }%
251   \expandafter\let\expandafter
252   \firstcol@TP\csname colordef@TP@#3\endcsname   % Retrieve definition of color <source1>.
253   \expandafter\expandafter\expandafter\firstcol@TP% and apply (remember \processcolor... is
254   \csname colordef@TP@#4\endcsname%              to definition of color <source2>.
255   \edef\end@TP%                                  Define color <target> (outside the encl
256   {\endgroup\noexpand\definecolor{#2}\newcolordef@TP}%
257   \end@TP
258   }% matches \newcommand{\colorbetween}

```

`\complement@TP` Calculates the complement of a fixed point value.

```

259   \newcommand{\complement@TP}[1]%
260   {%
261     \setlength{\tempdima@TP}{1pt-#1pt}%
262     \edef\result@TP{\strip@pt\tempdima@TP}%
263   }

```

`\complement@three@TP` Complements a three-piece color value.

```

264   \def\complement@three@TP#1,#2,#3;%
265   {%
266     \complement@TP{#1}%
267     \edef\newcolordef@TP{\result@TP,}%          Store first value.
268     \complement@TP{#2}%
269     \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
270     \complement@TP{#3}%
271     \edef\newcolordef@TP{\newcolordef@TP\result@TP}% Store third value.
272   }

```

`\grabfourth@TP` Separates the fourth element of a four-piece color value from the rest.

```

273   \def\grabfourth@TP#1,#2,#3,#4;%
274   {%
275     \def\mem@TP{#4}%                             Store fourth element.
276     \def\result@TP{#1,#2,#3;}%                   Store first three elements.
277   }

```

```

\complementcolor \complementcolor{<target>}{<source>} calculates the numerical complement
of a color. <source> is the name of the color to be complemented. <target>
is the name to be given to the resulting color. \complementcolor supports the
following color models: rgb, RGB, gray, cmyk, hsb.
278 \newcommand{\complementcolor}[2]%
279 {%
280     \begingroup%                               Make the definition of \processcolor... local.
281     \newcommand{\processcolor@TP@rgb}[1]%       What if the color is an rgb color?
282     {%
283         \complement@three@TP##1;%             Calculate complemented values.
284         \edef\newcolordef@TP{{rgb}{\newcolordef@TP}}% Store the result
285     }%
286     \newcommand{\processcolor@TP@gray}[1]%     What if the color is a gray color?
287     {%
288         \complement@TP{##1}%                 Calculate complemented value.
289         \edef\newcolordef@TP{{gray}{\result@TP}}% Store the result
290     }%
291     \newcommand{\processcolor@TP@cmyk}[1]%     What if the color is a cmyk color?
292     {%
293         \grabfourth@TP##1;%                   Remember fourth element.
294         \expandafter\complement@three@TP\result@TP% Calculate complemented values of first thr
295         \edef\newcolordef@TP{{cmyk}{\newcolordef@TP,\mem@TP}}% Store the result, putting back t
296     }%
297     \newcommand{\processcolor@TP@RGB}[1]%     What if the color is an RGB color?
298     {%
299         \convert@RGB@rgb@TP##1;%             Convert to rgb ...
300         \expandafter\processcolor@TP@rgb\expandafter{\newcolordef@TP}% ... and continue.
301     }%
302     \newcommand{\processcolor@TP@hsb}[1]%     What if the color is an hsb color?
303     {%
304         \complement@three@TP##1;%             Calculate complemented values.
305         \edef\newcolordef@TP{{hsb}{\newcolordef@TP}}% Store the result
306     }%
307     \csname colordef@TP@#2\endcsname%         Execute definition of color <source> (which cont
308     \edef\end@TP%                             Define color <target> (outside the enclosing gro
309     {\endgroup\noexpand\definecolor{#1}\newcolordef@TP}%
310     \end@TP
311     }% matches \newcommand{\complementcolor}
312     }% matches \ifthenelse{\boolean{TPcolor}}{% Yes.
313     {% No. Do nothing.
314     }

```

### 3.3 Color name and color set management

`\replacecolor` `\replacecolor[<tset>]{<tcoll>}[<sset>]{<scoll>}` will make `<tcoll>` have the same definition as `<scoll>` (if `<scoll>` is defined at all), where `|tcolli` and `|scolli` are color names as given in the first argument of `\definecolor`. If (one of) `<tset>` and `<sset>` are given, the colors will be taken from the respective color sets.

```

315 \newcommand{\replacecolor}
316 {%
317   \let\replacecolor@hook@TP=\@gobble% This hook can be used for variant checking (see below).
318   \replacecolor@TP%                Pick up arguments.
319 }
320
321 \newcommand{\replacecolor@TP}[2][ ]% Pick up the first two arguments of \replacecolor.
322 {%
323   \ifthenelse{\equal{#1}{}}{\edef\tcolname@TP{#2}}{\edef\tcolname@TP{#2@#1}}% Construct 'real'
324   \@replacecolor@TP%                Read second argument.
325 }%

326 \ifthenelse{\boolean{TPcolor}}% Only if TeXPower's color management is active.
327 {% Yes.
328   \newcommand{\undefinecolor@TP}[1]%                Make a color undefined.
329   {\expandafter\let\csname\string\color @#1\endcsname=\@undefined}%
330
331
332   \newcommand{\ifcolorexists@TP}[3]%                Conditional for testing whether
333   {\@ifundefined{\string\color @#1}{#3}{#2}}%      Test whether a given color
334
335
336   \newcommand{\@replacecolor@TP}[2][ ]%            Second part of \replacecolor
337   {%
338     \ifthenelse{\equal{#1}{}}{\edef\scolname@TP{#2}}{\edef\scolname@TP{#2@#1}}% Construct 'real'
339     \ifcolorexists@TP{\scolname@TP}%                Does the source
340     {% Yes.
341       \replacecolor@hook@TP{\tcolname@TP}%          Execute hook.
342       \expandafter\let\csname\string\color @\tcolname@TP\expandafter\endcsname% Make value of t
343       \csname\string\color @\scolname@TP\endcsname% identical with
344       \expandafter\let\csname colordef@TP@\tcolname@TP\expandafter\endcsname% Make definition
345       \csname colordef@TP@\scolname@TP\endcsname% identical with
346     }%
347     {% No. Do nothing.
348     }%
349   }%

```

The set of TeXPower's 'standard colors' and some commands to manipulate them.

`\colors@TP` `\colors@TP` is the list of all standard colors defined by texpower. The list is empty initially.

```

350 \newcommand{\colors@TP}{ }

```

`\removecolor@TP` Removes a color name from the list.

```

351 \newcommand{\removecolor@TP}[1]%
352 {%
353   \def\processme@TP##1%                This macro does the real work.
354   {%
355     \ifthenelse{\equal{#1}{##1}}%      Is this the color to be removed?

```

```

356     {% Yes. Do nothing, so it vanishes.
357     }
358     {% No. Re-insert.
359     \expandafter\def\expandafter\colors@TP\expandafter{\colors@TP\processme@TP{##1}}%
360     }%
361     }%
362     \expandafter\let\expandafter\colors@TP\expandafter\empty% Initialize \colors@TP.
363     \colors@TP% Execute \processme@TP for every c
364     }

```

`\addTPcolor` `\addTPcolor{<color>}` adds the color named `<color>` to TeXPower's list of standard colors.

```

365 \newcommand{\addTPcolor}[1]%
366 {%
367     \removecolor@TP{#1}% Remove this color from the list (to avoid duplicates).
368     \expandafter\def\expandafter\colors@TP\expandafter{\colors@TP\processme@TP{#1}}% ... and in
369     \register@normalvariant@TP{#1}% Register the normal variant for this color.
370     }

```

`\defineTPcolor` `\defineTPcolor[<set>]{<name>}{<model>}{<def>}` acts like `\definecolor{<name>}{<model>}{<def>}`, but

1. color `<name>` is automatically added to the list of standard colors and
2. if the optional parameter is given, the color is defined in the color set `<set>` instead of the current color set.

```

371 \newcommand{\defineTPcolor}[4] []
372 {%
373     \addTPcolor{#2}% Add color to the list.
374     \ifthenelse{\equal{#1}{}}% Color from the current color set?
375     {\definecolor{#2}{#3}{#4}}% Yep. Just define the color.
376     {\definecolor{#2@#1}{#3}{#4}}% No. Add color set identifier.
377     }

```

Some commands for manipulating whole color sets.

`\replacecolors@TP` Low level command for replacing a complete color set.

```

378 \newcommand{\replacecolors@TP}%
379 {%
380     \@ifstar% The starred version will put the color set into
381     {\let\replacecolor@hook@TP=\register@normalvariant@TP\@replacecolors@TP}
382     {\let\replacecolor@hook@TP=\@gobble\@replacecolors@TP}%
383     }
384
385 \newcommand{\@replacecolors@TP}[4]% This part does the real work.
386 {%
387     \def\processme@TP##1{\replacecolor@TP[#1]{#2##1}[#3]{#4##1}}%
388     \colors@TP
389     }

```

`\usecolorset` `\usecolorset{<set>}` switches to color set <set>.

```
390 \newcommand{\usecolorset}[1]%
391 {%
392   \replacecolors@TP*{}{}{#1}{}%   Replace normal variant (registering variants).
393   \replacecolors@TP{}{d}{#1}{d}%   Replace dimmed variant.
394   \replacecolors@TP{}{e}{#1}{e}%   Replace enhanced variant.
395   \color{textcolor}%               Activate textcolor.
396   \pagecolor{pagecolor}%           Activate pagecolor.
397   }%
```

`\dumpcolorset` `\dumpcolorset{<set>}` saves all standard colors from the current color set to the color set <set>.

```
398 \newcommand{\dumpcolorset}[1]%
399 {%
400   \nonnormalwarnings@TP{Dumping color set #1}% Output a warning for every color not in the no
401   \replacecolors@TP{#1}{}{}{}%           Dump normal variant (hopefully).
402   \replacecolors@TP{#1}{d}{}{d}%         Dump dimmed variant.
403   \replacecolors@TP{#1}{e}{}{e}%         Dump enhanced variant.
404   }%
```

Commands for color variants.

```
405 \newcommand{\registervariant@TP}[2]%           Remember which variant a color is currently in
406 {\expandafter\def\csname cvar@#1@TP\endcsname{#2}}
407
408 \newcommand{\register@normalvariant@TP}[1]%    Register that a color is now in the normal var
409 {\registervariant@TP{#1}{}{}}
410
411 \newcommand{\currentvariant@TP}[1]%           Return the current variant of a color.
412 {\csname cvar@#1@TP\endcsname}
413
414 \newcommand{\ifnormalvariant@TP}[3]%          Conditional for checking whether a color is in
415 {\ifthenelse{\equal{\currentvariant@TP{#1}{}{}}{#2}{#3}}
416
417 \newcommand{\nonnormalwarnings@TP}[1]%        Checks the current variant for every standard
418 {%                                             if it's not the normal one.
419   \def\processme@TP##1%
420   {%
421     \ifnormalvariant@TP{##1}{}
422     {%
423       \PackageWarning{texpower}
424       {#1\MessageBreak when color ##1 is in \currentvariant@TP{##1} variant}%
425     }%
426   }%
427   \colors@TP
428 }
```

Default dim level for automatic color dimming.

```
429 \newcommand{\dimlevel}{.7}
```

`\dimcolor[<level>]{<color>}` dims the color named <color>. It checks whether an explicit 'dimmed' variant `d;colorj` exists. If yes, <color> is replaced

by `d<color>`. Otherwise, the dimmed color is calculated by interpolating between `pagecolor` and `<color>`. The parameter for `\colorbetween` is given by the optional argument `<level>` (default: `\dimlevel`).

```

430 \newcommand{\dimcolor}[2][\dimlevel]
431 {%
432   \ifnormalvariant@TP{#2}%           Color in the normal variant?
433   {% Yes.
434     \registervariant@TP{#2}{d}%       Register dimmed variant.
435     \ifcolorexists@TP{d#2}%           Dedicated dimmed color found?
436     {\replacecolor{#2}{d#2}}%        Yes. use that one.
437     {\colorbetween[#1]{#2}{pagecolor}{#2}}% No. Dim numerically using \colorbetween.
438   }%
439   }{% No. Do nothing.
440   }

```

`\dimcolors[<level>]` dims all standard colors using `\dimcolor`. See the description of `\dimcolor` for details.

```

441 \newcommand{\dimcolors}[1][\dimlevel]
442 {%
443   \def\processme@TP##1{\dimcolor[#1]{##1}}%
444   \colors@TP
445   }%

```

Default enhance level for automatic color enhancing.

```

446 \newcommand{\enhancelevel}{.5}

```

`\enhancecolor[<level>]{<color>}` enhances the color named `<color>`. It checks whether an explicit ‘enhanced’ variant `e<color>` exists. If yes, `<color>` is replaced by `e<color>`. Otherwise, the enhanced color is calculated by ‘extrapolating’ from `pagecolor` and `<color>`. The parameter for `\colorbetween` is given by the optional argument `<level>` (default: `\enhancelevel`).

```

447 \newcommand{\enhancecolor}[2][\enhancelevel]
448 {%
449   \ifnormalvariant@TP{#2}%           Color in the normal variant?
450   {%
451     \registervariant@TP{#2}{e}%       Register enhanced variant.
452     \ifcolorexists@TP{e#2}%           Dedicated enhanced color found?
453     {\replacecolor{#2}{e#2}}%        Yes. use that one.
454     {\colorbetween[-#1]{#2}{pagecolor}{#2}}% No. Enhance numerically using \colorbetween.
455   }%
456   }{%
457   }%

```

`\enhancecolors[<level>]` enhances all standard colors using `\enhancecolor`. See the description of `\enhancecolor` for details.

```

458 \newcommand{\enhancecolors}[1][\enhancelevel]
459 {%
460   \def\processme@TP##1{\enhancecolor[#1]{##1}}%
461   \colors@TP
462   }%

```

Replace all colors from the current color set by a single color.

```
463 \newcommand{\replacecolorsbyone@TP}[2]%  
464 {%  
465   \def\processme@TP##1{\replacecolor{##1}{##2}}%  
466   \colors@TP  
467 }
```

The color used to make things ‘vanish’.

```
468 \newcommand{\vanishcolor}{pagecolor}  
\vanishcolors replaces all standard colors by \vanishcolor.  
469 \newcommand{\vanishcolors}[1][\vanishcolor]{\replacecolorsbyone@TP}{##1}}
```

TeXPower’s predefined color sets and commands to activate them. Redefine in tpcolors.cfg as convenient.

```
470 \input{tpcolors.cfg}  
\whitebackground sets the standard colors up for white background.  
471 \newcommand{\whitebackground}%  
472 {%  
473   \usecolorset{whitebg}%  
474 }
```

When the whitebackground option (or no background option, but some other color-activating option like colormath) is given, `\whitebackground` is executed automatically (at the end of the package to ensure that `texpower.cfg` was read).

```
475 \ifthenelse{\boolean{whitebackground@TP}}  
476 {\AtEndOfPackage{\whitebackground}}  
477 {}
```

`\lightbackground` sets the standard colors up for ‘light’ background.

```
478 \newcommand{\lightbackground}%  
479 {%  
480   \usecolorset{lightbg}%  
481 }
```

When the lightbackground option is given, `\lightbackground` is executed automatically.

```
482 \ifthenelse{\boolean{lightbackground@TP}}{\AtEndOfPackage{\lightbackground}}{}
```

`\darkbackground` sets the standard colors up for ‘dark’ background.

```
483 \newcommand{\darkbackground}%  
484 {%  
485   \usecolorset{darkbg}%  
486 }
```

Execute `\darkbackground` automatically if the darkbackground option was given.

```
487 \ifthenelse{\boolean{darkbackground@TP}}{\AtEndOfPackage{\darkbackground}}{}
```

`\blackbackground` sets the standard colors up for black background.

```
488 \newcommand{\blackbackground}%  
489 {%  
490   \usecolorset{blackbg}%  
491 }
```



Execute `\blackbackground` automatically if the `blackbackground` option was given.

```
492 \ifthenelse{\boolean{blackbackground@TP}}{\AtEndOfPackage{\blackbackground}}{}
```

If TeXPower's color management is active, setup LaTeX color management to use the dedicated colors.

```
493 \ifthenelse{\boolean{TPcolor}}
```

```
494 {%
```

```
495   \renewcommand{\normalcolor}{\color{textcolor}}%       \normalcolor should produce textcolor.
```

```
496
```

```
497   \let\o@textnormal@TP=\textnormal%                   \textnormal should also set text co
```

```
498   \def\textnormal#1{\o@textnormal@TP{\normalcolor#1}}
```

Make sure current color is correct for the rest of the preamble.

```
499   \AtEndOfPackage{\color{textcolor}\let\default@color\current@color}
```

The following is deferred to the beginning of the document to allow redefinitions of colors and loading of packages. We set page and text color and make `amsmath`'s `\text` command switch to text color.

```
500   \AtBeginDocument%
```

```
501   {%
```

```
502     \pagecolor{pagecolor}\color{textcolor}%
```

```
503     \ifpackageloaded{amstext}%
```

```
504     {%
```

```
505       \let\o@text@TP=\text%
```

```
506       \def\text#1{\o@text@TP{\normalcolor\expandafter\everymath\expandafter{\the\everymath\co
```

```
507       }%
```

```
508     }%
```

```
509   }%
```

```
510 }
```

```
511 {}
```

If TeXPower's color management is active, set page and text color at the beginning of the document.

```
512 \ifthenelse{\boolean{TPcolor}}{\AtBeginDocument{\pagecolor{pagecolor}\color{textcolor}}{}
```

```
513 }% matches \ifthenelse{\boolean{TPcolor}}{% Yes.
```

```
514 {% No; provide dummies.
```

```
515 \let\setcolor@TP=\gobble%
```

```
516 \newcommand{\@replacecolor@TP}[2] [] {}%
```

```
517 \let\addTPcolor=\gobble
```

```
518 \newcommand{\defineTPcolor}[4] [] {}%
```

```
519 \let\usecolorset=\gobble
```

```
520 \let\dumpcolorset=\gobble
```

```
521 \newcommand{\dimcolor}[2] [] {}
```

```
522 \newcommand{\dimcolors}[1] [] {}
```

```
523 \newcommand{\enhancecolor}[2] [] {}
```

```
524 \newcommand{\enhancecolors}[1] [] {}
```

```
525 \newcommand{\vanishcolors}[1] [] {}
```

```
526 }
```

### 3.3.1 Implementation of the coloremph option

```

527 \ifthenelse{\boolean{coloremph@TP}}%           Should \emph use color?
528 {% Yes;
529   \DeclareRobustCommand{\em}%                   Redefine \em.
530   {%
531     \@nomath\em \color{emcolor}%               Change color.
532     \replacecolor{tmp@TP}{emcolor}%           Exchange emcolor and altemcolor.
533     \replacecolor{emcolor}{altemcolor}%
534     \replacecolor{altemcolor}{tmp@TP}%
535   }%
536 }%
537 {}% No; keep original definition.

```

### 3.3.2 Implementation of the colormath option

Note that the following code is quite fragile and contains some modifications of LaTeX internals. Thus it is likely to cause trouble, especially in conjunction with other packages modifying the LaTeX kernel. The array package is supported, but no explicit support of other packages exists. If you experience strange and inexplicable errors while the colormath option is active, first of all try switching it off to see whether anything changes. The implementation of colormath is likely to change several times before the first beta release, so expect backward incompatible changes in behaviour.

```

538 \ifthenelse{\boolean{colormath@TP}}%           Should we color math?
539 {% Yes.
540   \AtBeginDocument
541   {%

```

The most basic magical incantation: Color inline math using `\everymath`. Beware of side effects of this hack.

```

542   \expandafter\everymath\expandafter{\the\everymath\color{mathcolor}}%

```

Color displayed math by overloading LaTeX's own math environments. Note that this doesn't work for the TeX notation `$$`, which is deprecated in LaTeX anyway. Note further that for the `eqnarray` and `eqnarray*` environments, the current implementation places the color change command **outside** the math environment (for technical reasons; maybe this can be remedied by a more sophisticated implementation), which will almost invariably lead to unwanted extra vertical space before and after equation arrays. Currently there is no clean remedy, apart from using `amsmath`'s `align` environment.

```

543   \let\o@dm@TP=\[%                               Save the original definitions of begi
544   \let\o@enddm@TP=\]%                             LaTeX's displayed math environments.
545   \let\o@eqa@TP=\eqnarray%
546   \let\o@endeqa@TP=\endeqnarray%
547   \expandafter\let\expandafter\o@eqastar@TP\csname eqnarray*\endcsname%
548   \expandafter\let\expandafter\o@endeqastar@TP\csname endeqnarray*\endcsname%
549   \def\[{ \o@dm@TP\begin\color{mathcolor}}%       Redefine the begin and end macros for
550   \def\} {\end\o@enddm@TP}%                       environments, adding the color change
551   \def\eqnarray{\begin\color{mathcolor}\o@eqa@TP}% level of grouping.

```

```

552 \def\endeqnarray{\o@endeqa@TP\endgroup\@ignoretrue}%
553 \@namedef{eqnarray*}{\begingroup\color{mathcolor}\o@eqastar@TP}
554 \@namedef{endeqnarray*}{\o@endeqastar@TP\endgroup\@ignoretrue}
555 \@ifpackageloaded{amsmath}% Amsmath's displayed math environment
556 {% \everymath hack because they are 'fa
557 \ifpackagelater{amsmath}{2000/01/15}% As amsmath 1.x redefines the equat
558 {% variant of gather, treating it as
559 \let\o@eq@TP=\equation% environment would lead to problems
560 \let\o@endeq@TP=\endequation% made only if amsmath 1.x is not lo
561 \def\equation{\o@eq@TP\begingroup\color{mathcolor}}%
562 \def\endequation{\endgroup\everymath{}\o@endeq@TP}%
563 }%
564 }%
565 }%
566 {%
567 \let\o@eq@TP=\equation%
568 \let\o@endeq@TP=\endequation%
569 \def\equation{\o@eq@TP\begingroup\color{mathcolor}}%
570 \def\endequation{\endgroup\everymath{}\o@endeq@TP}%
571 }%
572 }% matches \AtBeginDocument{

```

Sometimes, a math environment is used for something other than displaying math. The macro `\origmath` will put its argument in math mode, but turn off coloring. If another math environemt should be nested inside the argument of `\origmath`, it will be coloured.

```

573 \newcommand{\origmath}[1]{\everymath{}\ensuremath{\everymath{\color{mathcolor}}#1}}%

```

We need to redefine some LaTeX macros which internally use math mode, to make sure that not all tabulars and parboxes are coloured. Note that this can break packages which mess with tabular themselves.

```

574 \renewcommand*\labelitemi{\origmath{\m@th\bullet}}%
575 \@ifpackageloaded{array}% The array package redefines \@tabular
576 {%
577 \def\@tabular{%
578 \leavevmode
579 \hbox \bgroup \everymath{\$}\everymath{\color{mathcolor}}\col@sep\tabcolsep \let\d@llarbeg
580 \let\d@llarend\endgroup
581 \@tabarray
582 }%
583 \@ifpackageloaded{colortbl}
584 {%
585 \def\@classz{\@classx
586 \@tempcnta \count@
587 \prepnext@tok
588 \expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil
589 \@addtopreamble{%
590 \setbox\z@\hbox\bgroup\bgroup
591 \ifcase \@chnum
592 \hskip\stretch{.5}\kern\z@

```

```

593         \dollarbegin
594         \insert@column
595         \dollarend\hskip\stretch{.5}\or
596         \dollarbegin \insert@column \dollarend \hfill \or
597         \hfill\kern\z@ \dollarbegin \insert@column \dollarend \or
598         \@startvcenter
599         \@startpbox{\@nextchar}\insert@column \@endpbox $\or % $
600         \vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
601         \vbox \@startpbox{\@nextchar}\insert@column \@endpbox
602         \fi
603         \egroup\egroup
604         \begingroup
605         \CT@setup
606         \CT@column@color
607         \CT@row@color
608         \CT@do@color
609         \endgroup
610         \@tempdima\ht\z@
611         \advance\@tempdima\minrowclearance
612         \vrule\@height\@tempdima\@width\z@
613         \unhbox\z@}%
614     \prepnext@tok}%
615 }
616 {%
617     \def\@classz{\@classx
618         \@tempcnta \count@
619         \prepnext@tok
620         \@addtopreamble{\ifcase \@chnum
621             \hfil
622             \dollarbegin
623             \insert@column
624             \dollarend \hfil \or
625             \hskip1sp\dollarbegin \insert@column \dollarend \hfil \or
626             \hfil\hskip1sp\dollarbegin \insert@column \dollarend \or
627             \@startvcenter
628             \@startpbox{\@nextchar}\insert@column \@endpbox $\or % $
629             \vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
630             \vbox \@startpbox{\@nextchar}\insert@column \@endpbox
631             \fi}\prepnext@tok}%
632     }
633     \DeclareRobustCommand\@startvcenter{\everymath{}}$\everymath{\color{mathcolor}}\vcenter}% $
634     \expandafter\def\expandafter\@mkpream\expandafter#\expandafter1%
635     \expandafter{%
636         \expandafter\let\expandafter\@startvbox\expandafter\relax
637         \@mkpream{#1}}
638 }
639 {%
640     \def\@tabular{\leavevmode \hbox \bgroup \everymath{}}$\everymath{\color{mathcolor}}\let\@aco
641         \let\@classz\@tabclassz
642         \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray% $

```

```

643     }%
644   }
645 \long\def\@iiiparbox#1#2[#3]#4#5{%
646   \leavevmode
647   \@pboxswfalse
648   \setlength\@tempdima{#4}%
649   \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@par}%
650   \ifx\@empty#2\else\ifx\relax#2\else
651     \setlength\@tempdimb{#2}%
652     \def\@parboxto{to\@tempdimb}%
653     \fi\fi
654     \if#1b\vbox
655       \else\if #1t\vtop
656         \else\ifmode\vcenter
657           \else\@pboxswtrue \everymath{\$}\everymath{\color{mathcolor}}\vcenter
658         \fi\fi\fi
659     \@parboxto{\let\hss\vss\let\unhbox\unvbox
660       \csname bm@#3\endcsname}%
661     \if@pboxsw \m@th$\fi
662   \@end@tempboxa}
663 \let\o@textsuperscript@TP=\textsuperscript
664 \def\textsuperscript#1{\everymath{\o@textsuperscript@TP{\everymath{\color{mathcolor}}#1}}}%
665 }% matches \ifthenelse{\boolean{colormath@TP}}{% Yes.
666 {% No; keep original definition.
667 \let\origmath=\ensuremath%           \origmath needs to have a sensible definition.
668 }

```

New highlighting and emphasis commands. Most of them have a sensible alternative definition if the `colorhighlight` option is not given.

`\code{<text>}` will display `|text|` in a ‘code-like’ style (for shell commands or macro names). `\codeswitch` switches to the style used by `\code`, for use e.g. in verbatim environments.

```

669 \ifthenelse{\boolean{colorhighlight@TP}}%           Color highlighting enabled?
670 {% Yes; code is displayed typewriter-style, bold and in a special color.
671   \DeclareRobustCommand{\code}[1]{\textcolor{codecolor}{\textbf{\texttt{#1}}}}%
672   \DeclareRobustCommand{\codeswitch}{\color{codecolor}\bfseries\ttfamily}%
673 }
674 {% No; code is displayed just in typewriter-style and bold.
675   \DeclareRobustCommand{\code}[1]{\textbf{\texttt{#1}}}%
676   \DeclareRobustCommand{\codeswitch}{\bfseries\ttfamily}%
677 }

```

`\macroname` `\macroname{<text>}` acts like `\code`, but adds a backslash in front.

```
678 \newcommand{\macroname}[1]{\code{\textbackslash#1}}
```

`\commandapp` `\commandapp[<opt>]{<name>}{<arg>}` displays a macro with an argument. `<name>` is the macro name, `<opt>` is an optional argument, `jargj` is the macro argument. Note that only one pair of braces is added for `<arg>`; for several arguments, `\}{\}` needs to be used inside `<arg>` to separate arguments.

```
679 \newcommand{\commandapp}[3][ ]{\code{\macroname{#2}\ifthenelse{\equal{#1}{}}{\{#1\}}{\{#3\}}}
```

```

\carg \carg{<text>} displays a ‘symbolic argument’, i.e. <text> in code style enclosed
in pointy braces.
680 \newcommand{\carg}[1]{\code{\origmath{\left<\code{#1}\right>}}}

\underl \underl{<text>} emphasises <text> using a special color if the colorhighlight
option is given and by boldfacing otherwise.
681 \ifthenelse{\boolean{colorhighlight@TP}}%           Color highlighting enabled?
682 {% Yes;
683 \DeclareRobustCommand{\underl}{\textcolor{underlcolor}}% Use color to highlight.
684 }
685 {% No;
686 \DeclareRobustCommand{\underl}{\textbf}%           Use bold face.
687 }

\concept \concept{<text>} emphasises <text> using a special color if the colorhighlight
option is given and by boldfacing otherwise. To be used for emphasizing names of
(new) concepts.
688 \ifthenelse{\boolean{colorhighlight@TP}}%           Color highlighting enabled?
689 {% Yes;
690 \DeclareRobustCommand{\concept}{\textcolor{conceptcolor}}% Use color to highlight.
691 }
692 {% No;
693 \DeclareRobustCommand{\concept}{\textbf}%           Use bold face.
694 }

\inactive \inactive{<text>} emphasises <text> using a special color if the colorhighlight
option is given. Nothing is done if the option is not given. To be used for ‘de-
emphasizing’ things not currently of interest.
695 \ifthenelse{\boolean{colorhighlight@TP}}%           Color highlighting enabled?
696 {% Yes;
697 \DeclareRobustCommand{\inactive}{\textcolor{inactivecolor}}% Use color to highlight.
698 }
699 {% No;
700 \DeclareRobustCommand{\inactive}{\monochromeinactive}% Use monochrome default.
701 }
702
703 \providecommand{\monochromeinactive}{}% What should \inactive do if colors can’t be used? We pr
704 % user definitions.

\present \present[<opt>]{<text>} puts its argument into an \fbox with coloured back-
ground. If <opt> is given, it is added to the left of the box without taking any
space, i.e. it will overlap text to the left of the box. This addition is useful mainly
for adding ‘constraints’ to things presented in a description or center environment.
705 \ifthenelse{\boolean{colorhighlight@TP}}%           Color highlighting enabled?
706 {% Yes; use a colored box.
707 \newcommand{\present}[2][\leavevmode\llap{\textbf{\footnotesize#1}\,}]{\fcolorbox{textcolor}{
708 \newcommand{\mkpbox@TP}[1]{\fcolorbox{presentcolor}{presentcolor}{#1}}% Internal macro for us
709 }

```

```

710 {% No; use an \fbox.
711 \newcommand{\present}[2][\leavevmode\llap{\textbf{\footnotesize#1}\,}\fbox{#2}}%
712 \newcommand{\mkpbox@TP}[1]{\fbox{#1}}%
713 }

```

**presentbox** The presentbox environment creates a coloured patch of width `\linewidth` with a minipage inside. If the `colorhighlight` option is not given, an `\fbox` containing the minipage is created.

```

714 \newsavebox{\pbox@TP}% Container for the minipage to be boxed.
715 \newenvironment{presentbox}%
716 {%
717 \par\smallskip% First a small space to separate the area
718 \begin{lrbox}{\pbox@TP}% Save the contents in a minipage inside
719 \noindent
720 \begin{minipage}{\linewidth-2\fboxsep-2\fboxrule}% Reduce the width of the minipage to leave
721 \replacecolor{presentcolor}{pagecolor}% If \present is used inside the colored
722 }%
723 {%
724 \end{minipage}
725 \end{lrbox}%
726 \noindent\mkpbox@TP{\usebox{\pbox@TP}}% This typesets the saved minipage inside
727 \smallskip% A small space to separate the area from
728 \par
729 }

```

### 3.4 Structured rules, box and page backgrounds

#### 3.4.1 Structured rules

Some configurable defaults for rules and box backgrounds.

Default number of stripes for gradient rules and box backgrounds.

```
730 \newcommand{\rulestripes}{10}
```

Default stripe overlap for avoiding ‘gaps’ in color gradients.

```
731 \newcommand{\stripeoverlap}{.15pt}
```

Default gradient progression for rules and box backgrounds (single gradients or first part of double gradients).

```
732 \newcommand{\rulefirstgradprogression}{1}
```

Default gradient progression for rules and box backgrounds (second part of double gradients).

```
733 \newcommand{\rulesecondgradprogression}{1}
```

Default position of the ‘middle’ color of a double gradient.

```
734 \newcommand{\rulegradmidpoint}{.5}
```

The following are used internally when making color gradients.

```
735 \newcounter{stripe@TP}
```

```
736
```

```
737 \newcounter{stripes@TP}
```

```

738
739 \newcommand{\firstgradprogression@TP}{1}
740
741 \newcommand{\secondgradprogression@TP}{1}
742
743 \newcounter{gradprogression@TP}
744
745 \newcommand{\gradmidpoint@TP}{.5}

Special versions of \mkfactor which apply gradient progressions.
746 \newcommand{\mkgradfirstfactor@TP}{\mkgradfactor@TP\firstgradprogression@TP}
747
748 \newcommand{\mkgradsecondfactor@TP}{\mkgradfactor@TP\secondgradprogression@TP}
749
750 \newcommand{\mkgradfactor@TP}[3]% Calculate a factor modified by a 'progression' parameter.
751 {%
752   \mkfactor{#2}{#3}% Calculate the unmodified factor.
753   \setcounter{gradprogression@TP}{#1}% Factor definition may contain a calc-expression
754   \ifthenelse{\value{gradprogression@TP}=1}{}% Progression value 1 is neutral.
755   {%
756     \ifthenelse{\value{gradprogression@TP}<0}% 'Negative' progression?
757     {% Yes.
758       \@tempcnta-\value{gradprogression@TP}\relax% Complement progression wrt 0.
759       \mkfactor{#2}{1-1pt*\real{#2}}% Complement factor definition wrt 1pt.
760     }
761     {\@tempcnta\value{gradprogression@TP}\relax}% No; Use progression as given.
762     \whiledo{\the\@tempcnta>1}% Calculate (factor definition)^(progression).
763     {\advance\@tempcnta by -1\relax\mkfactor{#2}{\real{#2}*\real{#2}}}%
764     \ifthenelse{\value{gradprogression@TP}<0}% 'Negative' progression?
765     {% Yes.
766       \mkfactor{#2}{1-1pt*\real{#2}}% Complement result wrt 1pt.
767     }
768   }%
769 }%
770 }

```

`\vgradrule` `\vgradrule[<stripes>][<startmodel>]{<startcolor>}[<endmodel>]{<endcolor>}[<raise>]{<width>}` creates a rule-like object consisting of a vertical color gradient composed of horizontal stripes.

The topmost stripe has color `{<startcolor>}`, the bottommost stripe has color `{<endcolor>}`. Inbetween, color changes gradually from top to bottom. The colors are specified exactly as for the `\color` command: if the optional argument `<startmodel>` is given, `<startcolor>` contains an explicit definition of a color from model `<startmodel>`, otherwise `<startcolor>` is the name of a defined color. The same holds for `<endmodel>` and `<endcolor>`.

The arguments `[<raise>]{<width>}{<height>}` work exactly as for the `\rule` command.

The optional argument `<stripes>`, if given, should contain a (calc expression for a) number specifying the number of stripes. If `<stripes>` is not given, the



default is the content of `\rulestripes` (default 10).

There is one more parameter which is not given as an argument. The control sequence `\rulefirstgradprogression` should expand to an (calc expression for an) integer. This value (default 1) controls the ‘order’ of progression from `<startcolor>` to `<endcolor>`. The default value 1 means linear progression. 2 means quadratic progression, i.e. color values ‘nearer’ to `<endcolor>` are reached ‘later’ (the square of 0.5, for instance, is 0.25, i.e. in the geometric middle point of the rule produced, the color gradient will have traveled only to one quarter of the ‘distance’ between `<startcolor>` and `<endcolor>`). 3 means cubic progression and so on. 0 and -1 mean the same as 1. -2 means quadratic progression “from bottom to top”, i.e. color values ‘nearer’ to `<endcolor>` are reached ‘earlier’, and analogously for -3, -4, ...

If you wish to give the second optional argument but not the first, just write `\vgradrule[] [<startmodel>]...`

```

771 \newcommand{\vgradrule}[1] []% Pick up first optional argument: [<stripes>]
772 {%
773   \let\firstgradprogression@TP=\rulefirstgradprogression%   Use progression parameter for rules.
774   \ifthenelse{\equal{#1}{}}%                                   First optional argument given?
775   {\setcounter{stripes@TP}{\rulestripes}}%                   No; use default value.
776   {\setcounter{stripes@TP}{#1}}%                               Yes.
777   \vgradrule@TP%                                             Pick up [<startmodel>]{<startcolor>}
778 }
779
780 \newcommand{\vgradrule@TP}[2] []% Pick up next pair of arguments: [<startmodel>]{<startcolor>}.
781 {%
782   \ifthenelse{\equal{#1}{}}%                                   <startmodel> given?
783   {\replacecolor{startcolor@TP}{#2}}%                         No; <startcolor> is a color name.
784   {\definecolor{startcolor@TP}{#1}{#2}}%                     Yes; {<startmodel>}{<startcolor>} is a color definit
785   \@vgradrule@TP%                                             Pick up [<endmodel>]{<endcolor>}.
786 }
787
788 \newcommand{\@vgradrule@TP}[2] []% Pick up next pair of arguments: [<endmodel>]{<endcolor>}.
789 {%
790   \ifthenelse{\equal{#1}{}}
791   {\replacecolor{endcolor@TP}{#2}}
792   {\definecolor{endcolor@TP}{#1}{#2}}%
793   \@@vgradrule@TP%                                             Pick up rule arguments and proceed.
794 }

Helper command for making one stripe. Can be overladed for making histograms.
795 \newcommand{\hstripe@TP}[4]%
796 {\hbox{\setcolor@TP{stripecolor@TP}\rule{#2}{#3}}#4}

Main part of \vgradrule.
797 \newcommand{\@@vgradrule@TP}[3] [0pt]%
798 {%
799   \ifthenelse{\value{stripes@TP}<2}%                           A ‘pathological case’ which can happen in animations
800   %                                                             requested, a division by zero error would be produce
801   {\mbox{\setcolor@TP{startcolor@TP}\rule{#1}{#2}{#3}}}%     In this case, just produce a colored

```

```

802  {%
803    \raisebox{#1}%                               Evaluate the <raise> argument of the rule.
804    {%
805      \vbox%                                       A vbox with \offinterlineskip allows to align the st
806      {%
807        \offinterlineskip
808        \setcounter{stripe@TP}{0}%               Initialize the number of the current stripe.
809        \whiledo{\value{stripe@TP}<\value{stripes@TP}}
810        {%
811          \mkgradfirstfactor@TP{\tmp@TP}%       Make the weight for \colorbetween, based on the numb
812          {\value{stripe@TP}/(\value{stripes@TP}-1)}% and the first gradient progression.
813          \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{startcolor@TP}% Calculate stripe
814          \stepcounter{stripe@TP}%
815          \ifthenelse{\value{stripe@TP}=\value{stripes@TP}}%   Last stripe?
816          {\hstripe@TP{\tmp@TP}{#2}{(#3)/\value{stripes@TP}}{}}% Yes; make stripe w/o overlap.
817          {%                                       No; add a kern to make stripes
818            \hstripe@TP{\tmp@TP}{#2}{(#3)/\value{stripes@TP}+\stripeoverlap}{\kern-\stripeoverl
819            }%
820          }% matches \whiledo{\value{stripe@TP}<\value{stripes@TP}}{%
821          }% matches \vbox{%
822          }% matches \raisebox{#1}{%
823          }% matches second argument of \ifthenelse{\value{stripes@TP}<2}
824          }% matches \newcommand{\@@vgradrule@TP}[3][Opt]{%

```

```

\dblvggradrule \dblvggradrule[<midpoint>][<stripes>][<startmodel>]
                {<startcolor>}[<midmodel>]{<midcolor>}[<endmodel>]
                {<endcolor>}[<raise>]{<width>}{<height>} creates a rule-like object consisting of a vertical color gradient composed of horizontal stripes.

```

The behaviour is exactly like `\vgradrule`, only in addition to the defined ‘start’ and ‘end’ color, there is an additional defined ‘middle’ color. The color gradient first progresses from the start to the middle color and then from the middle to the end color.

[<midmodel>]{<midcolor>} specify the middle color exactly as described for the other colors in the description of `\vgradrule`.

The additional optional parameter <midpoint> is a fixed-point value specifying where in the produced gradient the middle color is placed. 0 means the middle color replaces the start color; 1 means the middle color replaces the end color; 0.5 means the middle color is placed in the geometric middle between start and end color. If <midpoint> is not given, the default is the content of `\rulegradmidpoint` (default 0.5).

There is another parameter not passed as an argument: while `\rulefirstgradprogression` specifies the order of progression from first to middle color as described for `\vgradrule`, `\rulesecondgradprogression` specifies the order of progression from middle to end color.

If you wish to give one from the first row of optional arguments but not the other(s), any one can be replaced by `[]` to use the default.

```

825 \newcommand{\dblvggradrule}[1][ ]% Pick up first optional argument: [<midpoint>]
826 {%

```

```

827 \let\firstgradprogression@TP=\rulefirstgradprogression% Use progression parameters for rules
828 \let\secondgradprogression@TP=\rulesecondgradprogression
829 \ifthenelse{\equal{#1}{}}% First optional argument given?
830 {\let\gradmidpoint@TP=\rulegradmidpoint}% No; use default value.
831 {\def\gradmidpoint@TP{#1}}% Yes.
832 \dblvgradrule@TP% Pick up second optional argument.
833 }
834
835 \newcommand{\dblvgradrule@TP}[1] []% Pick up second optional argument: [<stripes>]
836 {%
837 \ifthenelse{\equal{#1}{}}
838 {\setcounter{stripes@TP}{\rulestripes}}
839 {\setcounter{stripes@TP}{#1}}%
840 \@dblvgradrule@TP
841 }
842
843 \newcommand{\@dblvgradrule@TP}[2] []% Pick up next pair of arguments: [<startmodel>]{<startcolor>}.
844 {%
845 \ifthenelse{\equal{#1}{}}
846 {\replacecolor{startcolor@TP}{#2}}
847 {\definecolor{startcolor@TP}{#1}{#2}}%
848 \@@dblvgradrule@TP
849 }
850
851 \newcommand{\@@dblvgradrule@TP}[2] []% Pick up next pair of arguments: [<midmodel>]{<midcolor>}.
852 {%
853 \ifthenelse{\equal{#1}{}}
854 {\replacecolor{midcolor@TP}{#2}}
855 {\definecolor{midcolor@TP}{#1}{#2}}%
856 \@@@dblvgradrule@TP
857 }
858
859 \newcommand{\@@@dblvgradrule@TP}[2] []% Pick up next pair of arguments: [<endmodel>]{<endcolor>}.
860 {%
861 \ifthenelse{\equal{#1}{}}
862 {\replacecolor{endcolor@TP}{#2}}
863 {\definecolor{endcolor@TP}{#1}{#2}}%
864 \@@@@dblvgradrule@TP% Pick up rule arguments and proceed.
865 }
866
867 \newcommand{\@@@@dblvgradrule@TP}[3] [Opt]% Main part of \dblvgradrule.
868 {%
869 \ifthenelse{\value{stripes@TP}<2}% The gradient code is not equipped for m
870 {\mbox{\setcolor@TP{midcolor@TP}\rule[#1]{#2}{#3}}}% In this case, just produce a rule color
871 {%
872 \raisebox{#1}% Evaluate the <raise> argument of the ru
873 {%
874 \vbox% A vbox with \offinterlineskip allows to
875 {% vertically.
876 \offinterlineskip

```

```

877     \setcounter{stripe@TP}{0}%           Initialize the number of the current st
878     \setcounter{tmpcnta@TP}%           Number of the 'middle' stripe.
879     {\value{stripes@TP}*\real{\gradmidpoint@TP}}%
880     \whiledo{\value{stripe@TP}<\value{tmpcnta@TP}}% Produce the upper part of the gradient.
881     {%
882         \mkgradfirstfactor@TP{\tmp@TP}%           Make the weight for \colorbetween, base
883         {\value{stripe@TP}/\value{tmpcnta@TP}}%           current stripe and the first gradient p
884         \colorbetween[\tmp@TP]{stripecolor@TP}{midcolor@TP}{startcolor@TP}% Calculate stripe
885         \hstripe@TP{\tmp@TP}%           Make stripe with
886         {\#2}{\#3}/\value{stripes@TP}+\stripeoverlap}{\kern-\stripeoverlap}% This is the upper
887         \stepcounter{stripe@TP}%
888     }%
889     \stepcounter{tmpcnta@TP}%
890     \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}% Only one stripe left to produce?
891     {\hstripe@TP{1}{\#2}{\#3}/\value{stripes@TP}}% Just produce one stripe colored with
892     {%
893         \whiledo{\value{stripe@TP}<\value{stripes@TP}}% Produce the lower part of the gradi
894         {%
895             \mkgradsecondfactor@TP{\tmp@TP}%           Make the weight for \colorbetween.
896             {(\value{stripe@TP}-\value{tmpcnta@TP}+1)/(\value{stripes@TP}-\value{tmpcnta@TP})}%
897             \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{midcolor@TP}% Calculate stripe
898             \stepcounter{stripe@TP}%
899             \ifthenelse{\value{stripe@TP}=\value{stripes@TP}}% Last stripe?
900             {\hstripe@TP{\tmp@TP}{\#2}{\#3}/\value{stripes@TP}}% Yes; make stripe w/o overlap
901             {% No; add kern to make stripes
902                 \hstripe@TP{\tmp@TP}{\#2}{\#3}/\value{stripes@TP}+\stripeoverlap}{\kern-\stripeove
903             }%
904             }% matches \whiledo{\value{stripe@TP}<\value{stripes@TP}}%
905             }% matches second argument of \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}%
906             }% matches \vbox{%
907             }% matches \raisebox{\#1}{%
908             }% matches second argument of \ifthenelse{\value{stripes@TP}<2}%
909             }% matches \newcommand{\@@@dblvgradrule@TP}[3][Opt]%

```

`\hgradrule` `\hgradrule`[<stripes>][<startmodel>]{<startcolor>}[<endmodel>]{<endcolor>}[<raise>]{<width>}{<height>} creates a rule-like object consisting of a horizontal color gradient composed of vertical stripes. Parameters are exactly as those for `\vgradrule` (replacing 'top' by 'left' and 'bottom' by 'right'). See there for explanations.

```

910 \newcommand{\hgradrule}[1][ ]% Pick up first optional argument: [<stripes>].
911 {%
912     \let\firstgradprogression@TP=\rulefirstgradprogression% Use progression parameter for rules.
913     \ifthenelse{\equal{\#1}{}}% First optional argument given?
914     {\setcounter{stripes@TP}{\rulestripes}}% No; use default value.
915     {\setcounter{stripes@TP}{\#1}}% Yes.
916     \hgradrule@TP% Pick up [<startmodel>]{<startcolor>}
917 }
918
919 \newcommand{\hgradrule@TP}[2][ ]% Pick up next pair of arguments: [<startmodel>]{<startcolor>}.

```

```

920 {%
921 \ifthenelse{\equal{#1}{}}
922 {\replacecolor{startcolor@TP}{#2}}
923 {\definecolor{startcolor@TP}{#1}{#2}}%
924 \@hgradrule@TP
925 }
926
927 \newcommand{\@hgradrule@TP}[2] []
928 {%
929 \ifthenelse{\equal{#1}{}}
930 {\replacecolor{endcolor@TP}{#2}}
931 {\definecolor{endcolor@TP}{#1}{#2}}%
932 \@@hgradrule@TP% Pick up rule arguments and proceed.
933 }
934
935 \newcommand{\vstripe@TP}[4]% Helper command for making _one_ stripe. Can be overla
936 {\setcolor@TP{stripecolor@TP}\rule{#2}{#3}{#4}}
937
938 \newcommand{\@@hgradrule@TP}[3] [Opt]% Main part of \hgradrule.
939 {%
940 \ifthenelse{\value{stripes@TP}<2}% A 'pathological case' which can happen in animations
941 % requested, a division by zero error would be produce
942 {\mbox{\setcolor@TP{startcolor@TP}\rule{#1}{#2}{#3}}}% In this case, just produce a colored
943 {%
944 \raisebox{#1}% Evaluate the <raise> argument of the rule.
945 {%
946 \setcounter{stripe@TP}{0}% Initialize the number of the current stripe.
947 \whiledo{\value{stripe@TP}<\value{stripes@TP}}
948 {%
949 \mkgradfirstfactor@TP{\tmp@TP}% Make the weight for \colorbetween, based on the numb
950 {\value{stripe@TP}/(\value{stripes@TP}-1)}% and the first gradient progression.
951 \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{startcolor@TP}% Calculate stripe co
952 \stepcounter{stripe@TP}%
953 \ifthenelse{\value{stripe@TP}=\value{stripes@TP}}% Last stripe?
954 {\vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}}{#3}{}}% Yes; make stripe w/o overlap.
955 {% No; add a kern to make stripes o
956 \vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}+\stripeoverlap}{#3}{\kern-\stripeoverlap
957 }%
958 }% matches \whiledo{\value{stripe@TP}<\value{stripes@TP}}{%
959 }% matches \raisebox{#1}{%
960 }% matches second argument of \ifthenelse{\value{stripes@TP}<2}%
961 }% matches \newcommand{\@@hgradrule@TP}[3] [Opt]{%

```

`\dblhgradrule` `\dblhgradrule[<midpoint>][<stripes>][<startmodel>]`  
`{<startcolor>}[<midmodel>]{<midcolor>}[<endmodel>]`  
`{<endcolor>}[<raise>]{<width>}{<height>}` creates a rule-like object consisting of a horizontal color gradient composed of vertical stripes.  
Parameters are exactly as those for `\dblvgradrule` (replacing 'top' by 'left' and 'bottom' by 'right'). See there for explanations.

```

962 \newcommand{\dblhgradrule}[1] []% Pick up first optional argument: [<midpoint>].
963 {%
964 \let\firstgradprogression@TP=\rulefirstgradprogression% Use progression parameters for rules
965 \let\secondgradprogression@TP=\rulesecondgradprogression
966 \ifthenelse{\equal{#1}{}}{\let\gradmidpoint@TP=\rulegradmidpoint}{\def\gradmidpoint@TP{#1}}%
967 \dblhgradrule@TP
968 }
969
970 \newcommand{\dblhgradrule@TP}[1] []% Pick up second optional argument: [<stripes>].
971 {%
972 \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}%
973 \@dblhgradrule@TP
974 }
975
976 \newcommand{\@dblhgradrule@TP}[2] []% Pick up next pair of arguments: [<startmodel>]{<startcolor>
977 {%
978 \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
979 \@@dblhgradrule@TP
980 }
981
982 \newcommand{\@@dblhgradrule@TP}[2] []
983 {%
984 \ifthenelse{\equal{#1}{}}{\replacecolor{midcolor@TP}{#2}}{\definecolor{midcolor@TP}{#1}{#2}}%
985 \@@@dblhgradrule@TP
986 }
987
988 \newcommand{\@@@dblhgradrule@TP}[2] []
989 {%
990 \ifthenelse{\equal{#1}{}}
991 {\replacecolor{endcolor@TP}{#2}}
992 {\definecolor{endcolor@TP}{#1}{#2}}%
993 \@@@@dblhgradrule@TP% Pick up rule arguments and proceed.
994 }
995
996 \newcommand{\@@@@dblhgradrule@TP}[3] [0pt]% Main part of \dblhgradrule.
997 {%
998 \ifthenelse{\value{stripes@TP}<2}% The gradient code is not equipped for ma
999 {\mbox{\setcolor@TP{midcolor@TP}\rule[#1]{#2}{#3}}}% In this case, just produce a rule colo
1000 {%
1001 \raisebox{#1}% Evaluate the <raise> argument of the rul
1002 {%
1003 \setcounter{stripe@TP}{0}% Initialize the number of the current str
1004 \setcounter{tmpcnta@TP}% Number of the 'middle' stripe.
1005 {\value{stripes@TP}*\real{\gradmidpoint@TP}}%
1006 \whiledo{\value{stripe@TP}<\value{tmpcnta@TP}}% Produce the left part of the gradient.
1007 {%
1008 \mkgradfirstfactor@TP{\tmp@TP}% Make the weight for \colorbetween, base
1009 {\value{stripe@TP}/\value{tmpcnta@TP}}% current stripe and the first gradient p
1010 \colorbetween[\tmp@TP]{stripecolor@TP}{midcolor@TP}{startcolor@TP}% Calculate stripe co
1011 \vstripe@TP{\tmp@TP}% Make stripe with ov

```

```

1012      {(#2)/\value{stripes@TP}+\stripeoverlap}{#3}{\kern-\stripeoverlap}% This is the left part
1013      \stepcounter{stripe@TP}%
1014      }%
1015      \stepcounter{tmpcnta@TP}%
1016      \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}% Only one stripe left to produce?
1017      {\vstripe@TP{1}{(#2)/\value{stripes@TP}}{#3}{}}% Just produce one stripe colored with
1018      {%
1019      \whiledo{\value{stripe@TP}<\value{stripes@TP}}% Produce the right part of the gradient
1020      {%
1021      \mkgradsecondfactor@TP{\tmp@TP}% Make the weight for \colorbetween.
1022      {(\value{stripe@TP}-\value{tmpcnta@TP})/(\value{stripes@TP}-\value{tmpcnta@TP}-1)}%
1023      \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{midcolor@TP}% Calculate stripe color
1024      \stepcounter{stripe@TP}%
1025      \ifthenelse{\value{stripe@TP}=\value{stripes@TP}}% Last stripe?
1026      {\vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}}{#3}{}}% Yes; make stripe w/o overlap.
1027      {% Add kern to make stripes overlap
1028      \vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}+\stripeoverlap}{#3}{\kern-\stripeoverl
1029      }%
1030      }% matches \whiledo{\value{stripe@TP}<\value{stripes@TP}}{%
1031      }% matches second argument of \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}{%
1032      }% matches \raisebox{#1}{%
1033      }% matches second argument of \ifthenelse{\value{stripes@TP}<2}%
1034      }% matches \newcommand{\@@@dblgradrule@TP}[3][Opt]{%

```

Clipbox stuff. The first part isn't used currently.

```

1035 % PDF:
1036 % \def\clipbox{\@ifnextchar[{\clipbox@}{\clipbox@[Opt]}}
1037 % \def\clipbox@[#1]#2{%
1038 %   \begingroup
1039 %     \setlength{\@tempdima}{#1}%
1040 %     \setbox\@tempboxa=\hbox{%
1041 %       \color@begingroup
1042 %         #2%
1043 %       \color@endgroup
1044 %     }%
1045 %     \leavevmode\hbox to \wd\@tempboxa{%
1046 %       \@ifundefined{dimexpr}{%
1047 %         \@defbp\x{-\@tempdima}%
1048 %         \@tempdimc=\dp\@tempboxa
1049 %         \advance\@tempdimc by \@tempdima
1050 %         \@defbp\y{-\@tempdimc}%
1051 %         \@tempdimc=\wd\@tempboxa
1052 %         \advance\@tempdimc by 2\@tempdima
1053 %         \@defbp\w{\@tempdimc}%
1054 %         \@tempdimc=\dp\@tempboxa
1055 %         \advance\@tempdimc by \ht\@tempboxa
1056 %         \advance\@tempdimc by 2\@tempdima
1057 %         \@defbp\h{\@tempdimc}%
1058 %       \pdfliteral{%
1059 %         q % gsave

```

```

1060 %         \x\space\y\space\w\space\h\space re % rectangle
1061 %         W n% make clip-path
1062 %     }%
1063 % }{% e-TeX
1064 %     \pdfliteral{%
1065 %         q % gsave
1066 %         \@dintobp{-\@tempdima} % x
1067 %         \@dintobp{-\dp\@tempboxa-\@tempdima} % y
1068 %         \@dintobp{\wd\@tempboxa+2\@tempdima} % width
1069 %         \@dintobp{\dp\@tempboxa+\ht\@tempboxa+2\@tempdima} % height
1070 %         re % rectangle
1071 %         W n% make clip-path
1072 %     }%
1073 % }%
1074 % \rlap{\unhbox\@tempboxa}%
1075 % \pdfliteral{%
1076 %     Q% grestore
1077 % }%
1078 % \hss
1079 % }%
1080 % \endgroup
1081 % }
1082 % \begingroup\expandafter\expandafter\expandafter\endgroup
1083 % \expandafter\ifx\csname dimexpr\endcsname\relax
1084 %     \def\@defbp#1#2{%
1085 %         \setlength{\@tempdimb}{#2}%
1086 %         \setlength{\@tempdimb}{.99626401\@tempdimb}%
1087 %         \edef#1{\strip@pt\@tempdimb}%
1088 %     }
1089 % \else
1090 %     \def\@dintobp#1{%
1091 %         \strip@pt\dimexpr.99626401\dimexpr#1\relax\relax
1092 %     }
1093 % \fi
1094 %
1095 % PS:
1096 % \def\clipbox{\@ifnextchar[{\clipbox@}{\clipbox@[Opt]}}
1097 % \def\clipbox@[#1]#2{%
1098 %     \begingroup
1099 %         \setlength{\@tempdima}{#1}%
1100 %         \setbox\@tempboxa=\hbox{%
1101 %             \color@begingroup
1102 %                 #2%
1103 %             \color@endgroup
1104 %         }%
1105 %         \leavevmode\hbox to \wd\@tempboxa{%
1106 %             \@ifundefined{dimexpr}{%
1107 %                 \@defpt\x{-\@tempdima}%
1108 %                 \@tempdimc=\ht\@tempboxa
1109 %                 \advance\@tempdimc by \@tempdima

```



```

1110 %      \@defpt\y{-\@tempdimc}%
1111 %      \@tempdimc=\wd\@tempboxa
1112 %      \advance\@tempdimc by 2\@tempdima
1113 %      \@defpt\w{\@tempdimc}%
1114 %      \@tempdimc=\dp\@tempboxa
1115 %      \advance\@tempdimc by \ht\@tempboxa
1116 %      \advance\@tempdimc by 2\@tempdima
1117 %      \@defpt\h{\@tempdimc}%
1118 %      \special{ps:%
1119 %          gsave %
1120 %          SDict begin %
1121 %          \x\space PTtoDVIPS \y\space PTtoDVIPS rmoveto %
1122 %          currentpoint %
1123 %          \w\space PTtoDVIPS \h\space PTtoDVIPS rectclip %
1124 %          end%
1125 %      }%
1126 %  }{% e-TeX
1127 %      \special{ps:%
1128 %          gsave %
1129 %          SDict begin %
1130 %          \@dimtopt{-\@tempdima} PTtoDVIPS % x
1131 %          \@dimtopt{-\ht\@tempboxa-\@tempdima} PTtoDVIPS % y
1132 %          rmoveto currentpoint %
1133 %          \@dimtopt{\wd\@tempboxa+2\@tempdima} PTtoDVIPS % width
1134 %          \@dimtopt{\dp\@tempboxa+\ht\@tempboxa+2\@tempdima} %
1135 %          PTtoDVIPS % height
1136 %          rectclip %
1137 %          end%
1138 %      }%
1139 %  }%
1140 %      \rlap{\unhbox\@tempboxa}%
1141 %      \special{ps:gsrestore}%
1142 %      \hss
1143 %  }%
1144 %  \endgroup
1145 % }
1146 % \special{!%
1147 % /PTtoDVIPS{72.27 div Resolution mul}def%
1148 % % rectclip is a level 2 feature
1149 % /rectclip where{pop}{%
1150 % /rectclip{%
1151 %     newpath %
1152 %     4 2 roll moveto %
1153 %     exch dup 0 rlineto %
1154 %     exch 0 exch rlineto %
1155 %     neg 0 rlineto %
1156 %     closepath %
1157 %     clip %
1158 %     newpath %
1159 % }bind def%

```

```

1160 % }%
1161 %   ifelse%
1162 % }
1163 % \begingroup\expandafter\expandafter\expandafter\endgroup
1164 % \expandafter\ifx\csname dimexpr\endcsname\relax
1165 %   \def\defpt#1#2{%
1166 %     \setlength{\@tempdimb}{#2}%
1167 %     \edef#1{\strip@pt\@tempdimb}%
1168 %   }
1169 % \else
1170 %   \def\@dimtopt#1{%
1171 %     \strip@pt\dimexpr#1\relax
1172 %   }
1173 % \fi
1174
1175
1176 \ifpdf
1177   \providecommand{\clipbox}[2][\z@]
1178   {%
1179     \setlength{\@tempdima}{#1}%
1180     \setbox\@tempboxa=
1181     \hbox{\kern\@tempdima\vbox{\offinterlineskip\kern\@tempdima\hbox{#2}\kern\@tempdima}\kern\@tempdima}%
1182     \pdfxform\@tempboxa
1183     \leavevmode
1184     \hbox
1185     {%
1186       \kern-\@tempdima
1187       \vbox{\offinterlineskip\kern-\@tempdima\hbox{\pdfrefxform\pdflastxform}\kern-\@tempdima}%
1188       \kern-\@tempdima
1189     }%
1190   }
1191 \else
1192   \providecommand{\clipbox}[2][\z@]{\leavevmode\hbox{#2}}
1193 \fi
1194
1195   \dgradslope stuff.
1196 \newcommand{\dgradslope}{1,1}
1197
1198 \newcounter{dgradhslope@TP}
1199
1200 \newcounter{dgradvslope@TP}
1201 \def\setdgradslope(#1,#2){\setcounter{dgradhslope@TP}{#1}\setcounter{dgradvslope@TP}{#2}}
1202 \dgradrule stuff.
1203 \newcommand{\dgradrule}[1]{}
1204 {%
1205   \ifthenelse{\equal{#1}{}}{\expandafter\setdgradslope\expandafter(\dgradslope)}{\setdgradslope}
1206   \dgradrule@TP
1207 }

```

```

1207 \newcommand{\dgradrule@TP}[2] []
1208 {%
1209 \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
1210 \dgradrule@TP
1211 }
1212
1213 \newcommand{\@dgradrule@TP}[2] []
1214 {%
1215 \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
1216 \@@dgradrule@TP
1217 }
1218
1219 \newcommand{\dstripewd@TP}{.7pt}
1220
1221 \newcommand{\@@dgradrule@TP}[3] [Opt]
1222 {%
1223 \raisebox{#1}
1224 {%
1225 \thicklines
1226 \setcounter{stripe@TP}{0}%
1227 \setcounter{tmpcnta@TP}{1*\ratio{#3}{\dstripewd@TP}}%
1228 \setcounter{stripes@TP}{1*\ratio{#3}{\dstripewd@TP}+1*\ratio{#2}{\dstripewd@TP}}%
1229 \@xarg\c@dgradhslope@TP\@yarg\c@dgradvslope@TP
1230 \makebox[\dstripewd@TP]
1231 {%
1232 \whiledo{\value{stripe@TP}<\value{tmpcnta@TP}}
1233 {%
1234 \mkfactor{\tmp@TP}{\value{stripe@TP}/(\value{stripes@TP}-1)}%
1235 \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{startcolor@TP}%
1236 \setlength{\@linelen}{\dstripewd@TP*\value{stripe@TP}}%
1237 \raisebox{\dstripewd@TP*(\value{tmpcnta@TP}-\value{stripe@TP})}
1238 {\makebox[Opt]{\setcolor@TP{stripecolor@TP}\hbox to Opt{\@sline\hss}}}%
1239 \stepcounter{stripe@TP}%
1240 }%
1241 }%
1242 \whiledo{\value{stripe@TP}<\value{stripes@TP}}
1243 {%
1244 \mkfactor{\tmp@TP}{\value{stripe@TP}/(\value{stripes@TP}-1)}%
1245 \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{startcolor@TP}%
1246 \makebox[\dstripewd@TP]{\setcolor@TP{stripecolor@TP}.}%
1247 \stepcounter{stripe@TP}%
1248 }%
1249 }%
1250 }%

```

### 3.4.2 Structured box backgrounds

`\vgradbox` `\vgradbox[<stripes>][<startmodel>]{<startcolor>[<endmodel>]{<endcolor>}{<content>}`  
creates an mbox containing <content>, which has a background made up of a ver-

tical color gradient. In fact, the background exceeds the extent of `<content>` by the value of `\fboxsep` on every side, just like the `\colorbox` command from the color package.

The gradient background is constructed using the `\vgradrule` command; see the description of `\vgradrule` on the way the gradient is constructed and on the meaning of the parameters `<stripes>`, `<startmodel>`, `<startcolor>`, `<endmodel>`, `<endcolor>` and the additional parameter `\rulefirstgradprogression` which has the same meaning for `\vgradbox` as for `\vgradrule`.

```

1251 \newcommand{\vgradbox}[1] []% Pick up first optional argument: [<stripes>].
1252 {%
1253   \let\firstgradprogression@TP=\rulefirstgradprogression%   Use progression parameter for rules.
1254   \ifthenelse{\equal{#1}{}}%                                   First optional argument given?
1255   {\setcounter{stripes@TP}{\rulestripes}}%                   No; use default value.
1256   {\setcounter{stripes@TP}{#1}}%                               Yes.
1257   \vgradbox@TP%                                             Pick up remaining optional arguments
1258 }
1259
1260 \newcommand{\vgradbox@TP}[2] []
1261 {%
1262   \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#2}}%
1263   \@vgradbox@TP
1264 }
1265
1266 \newcommand{\@vgradbox@TP}[2] []
1267 {%
1268   \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
1269   \@@gradbox@TP\@vgradrule@TP%                               Make a generic background box with vertical
1270 }
1271
1272 \newcommand{\@@gradbox@TP}[2]%                               Generic background box.
1273 {%
1274   \leavevmode%                                             Make box behave like \mbox.
1275   \setbox\tempbox@TP
1276   \hbox{\kern\fboxsep{\set@color#2}\kern\fboxsep}% An \hbox containing <contents> plus addition
1277   \rlap%                                                  Underlay box background with rule command pa
1278   {%
1279     #1[-\fboxsep-\dp\tempbox@TP]%                           Box depth augmented by 'white' space.
1280     {\wd\tempbox@TP}%                                       Space on the sides has already been added.
1281     {\ht\tempbox@TP+\dp\tempbox@TP+2\fboxsep}}%           Total height.
1282   \box\tempbox@TP%                                         Overlay box contents.
1283   }%

```

`\hgradbox` `\hgradbox[<stripes>][<startmodel>]{<startcolor>}<endmodel>}{<endcolor>}{<content>}`  
 acts like `\vgradbox`, but creates the background using `\hgradrule`. See comments above.

```

1284 \newcommand{\hgradbox}[1] []
1285 {%
1286   \let\secondgradprogression@TP=\rulesecondgradprogression

```

```

1287 \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}
1288 \hgradbox@TP
1289 }
1290
1291 \newcommand{\hgradbox@TP}[2] []
1292 {%
1293 \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
1294 \@hgradbox@TP
1295 }
1296
1297 \newcommand{\@hgradbox@TP}[2] []
1298 {%
1299 \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
1300 \@@gradbox@TP\@@hgradrule@TP
1301 }

```

`\dblvgradbox` `\dblvgradbox` [`<midpoint>`] [`<stripes>`] [`<startmodel>`]  
`<startcolor>`] [`<midmodel>`] [`<midcolor>`] [`<endmodel>`] [`<endcolor>`] [`<contents>`]  
acts like `\vgradbox`, but creates the background using `\dblvgradrule` (hence  
the additional parameters). See comments above (and the description of  
`\dblvgradrule` concerning the meaning of `\dblvgradrule` parameters).

```

1302 \newcommand{\dblvgradbox}[1] []
1303 {%
1304 \let\firstgradprogression@TP=\rulefirstgradprogression% Use progression parameters for rules
1305 \let\secondgradprogression@TP=\rulessecondgradprogression
1306 \ifthenelse{\equal{#1}{}}{\let\gradmidpoint@TP=\rulegradmidpoint}{\def\gradmidpoint@TP{#1}}%
1307 \dblvgradbox@TP
1308 }
1309
1310 \newcommand{\dblvgradbox@TP}[1] []
1311 {%
1312 \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}
1313 \@dblvgradbox@TP
1314 }
1315
1316 \newcommand{\@dblvgradbox@TP}[2] []
1317 {%
1318 \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
1319 \@@dblvgradbox@TP
1320 }
1321
1322 \newcommand{\@@dblvgradbox@TP}[2] []
1323 {%
1324 \ifthenelse{\equal{#1}{}}{\replacecolor{midcolor@TP}{#2}}{\definecolor{midcolor@TP}{#1}{#2}}%
1325 \@@@dblvgradbox@TP
1326 }
1327
1328 \newcommand{\@@@dblvgradbox@TP}[2] []
1329 {%

```

```

1330 \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
1331 \@@gradbox@TP\@@@dblgradrule@TP
1332 }

\dblhgradbox \dblhgradbox[<midpoint>][<stripes>][<startmodel>]
    {<startcolor>}[<midmodel>]{<midcolor>}[<endmodel>]{<endcolor>}{<contents>}
acts like \dblgradbox, but creates the background using \dblhgradrule. See
comments above.

1333 \newcommand{\dblhgradbox}[1] []
1334 {%
1335 \let\firstgradprogression@TP=\rulefirstgradprogression
1336 \let\secondgradprogression@TP=\rulesecondgradprogression
1337 \ifthenelse{\equal{#1}{}}{\let\gradmidpoint@TP=\rulegradmidpoint}{\def\gradmidpoint@TP{#1}}%
1338 \dblgradbox@TP
1339 }
1340
1341 \newcommand{\dblhgradbox@TP}[1] []
1342 {%
1343 \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}%
1344 \@dblgradbox@TP
1345 }
1346
1347 \newcommand{\@dblhgradbox@TP}[2] []
1348 {%
1349 \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#2}}%
1350 \@dblgradbox@TP
1351 }
1352
1353 \newcommand{\@@dblhgradbox@TP}[2] []
1354 {%
1355 \ifthenelse{\equal{#1}{}}{\replacecolor{midcolor@TP}{#2}}{\definecolor{midcolor@TP}{#1}{#2}}%
1356 \@@@dblgradbox@TP
1357 }
1358
1359 \newcommand{\@@@dblhgradbox@TP}[2] []
1360 {%
1361 \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
1362 \@@@gradbox@TP\@@@dblgradrule@TP
1363 }

```

### 3.4.3 Structured page backgrounds

`\backgroundstyle` `\backgroundstyle[<options>]{<style>}` is the central command for structured page backgrounds. It works like `\pagestyle` and other commands of this type. This means `<style>` is a symbolic name specifying the general method by which the page background is constructed. The detailed construction is influenced by parameters which can be set in `<options>`. If given, the optional parameter `<options>` should contain a list of settings in “keyval” manner. The keyval method (which is used by the `\includegraphics` command from the `graphicx`

package, for instance) is based on associating a symbolic name with every parameter. `<options>` is then a comma-separated list of parameter settings of the form `<name>=<value>`, where `<name>` is the symbolic name of the parameter to be set and `<value>` is the value it is to be set to. Not every `<style>` evaluates every parameter. In the following, a description of all styles, together with lists of the parameters employed, is given. It is followed by a list of all parameters. Note that some parameter names internally access the same parameter. For instance, parameters `startcolor` and `startcolordef` both set the start color of a color gradient. In case of conflict, the last setting in the list `<options>` will prevail. It is noted in the list of parameters which other parameters are overwritten.

`<style>` may have one of the following values:

none	No background. This means the page background is whatever it would be if <code>\backgroundstyle</code> wasn't used at all (for instance, a plain area of color page-color if one of the color options has been given). Parameters used: none.
plain	Plain background. This means the page background is whatever it would be if <code>\backgroundstyle</code> wasn't used at all (for instance, a plain area of color page-color if one of the color options has been given). In addition to background style "none", the background style "plain" does produce panel backgrounds. The colors and dimensions of a "top panel", "bottom panel", "left panel", and "right panel" can be specified. Parameters used: <code>hpanels</code> , <code>autopanelcolor</code> , <code>toppanelcolor</code> , <code>bottompanelcolor</code> , <code>leftpanelcolor</code> , <code>rightpanelcolor</code> , <code>toppanelcolordef</code> , <code>bottompanelcolordef</code> , <code>leftpanelcolordef</code> , <code>rightpanelcolordef</code> , <code>toppanelheight</code> , <code>bottompanelheight</code> , <code>leftpanelwidth</code> , <code>rightpanelwidth</code> .

vgradient	<p>Vertical gradient. The page background is constructed using the <code>\vgradrule</code> command. In addition to the usual parameters of gradient rules, the vgradient background style allows to leave space for headers, footers, or panels. The colors and dimensions of a “top panel”, “bottom panel”, “left panel”, and “right panel” can be specified. The gradient rule fills the rectangular space left between the specified panels.</p> <p>Parameters used: stripes, firstgradprogression, startcolor, startcolordef, endcolor, endcolordef, hpanels, autopanels, toppanelcolor, bottompanelcolor, leftpanelcolor, rightpanelcolor, toppanelcolordef, bottompanelcolordef, leftpanelcolordef, rightpanelcolordef, toppanelheight, bottompanelheight, leftpanelwidth, rightpanelwidth.</p>
hgradient	<p>Horizontal gradient. The page background is constructed using the <code>\hgradrule</code> command. See the description of vgradient concerning panels.</p> <p>Parameters used: stripes, firstgradprogression, startcolor, startcolordef, endcolor, endcolordef, hpanels, autopanels, toppanelcolor, bottompanelcolor, leftpanelcolor, rightpanelcolor, toppanelcolordef, bottompanelcolordef, leftpanelcolordef, rightpanelcolordef, toppanelheight, bottompanelheight, leftpanelwidth, rightpanelwidth.</p>



- `doublevgradient` ‘Double’ vertical gradient. The page background is constructed using the `\dblvggradrule` command. See the description of `vgradient` concerning panels. Parameters used: `stripes`, `gradmidpoint`, `firstgradprogression`, `secondgradprogression`, `startcolor`, `startcolordef`, `midcolor`, `midcolordef`, `endcolor`, `endcolordef`, `hpanels`, `autopanelcolor`, `bottompanelcolor`, `leftpanelcolor`, `rightpanelcolor`, `toppanelcolordef`, `bottompanelcolordef`, `leftpanelcolordef`, `rightpanelcolordef`, `toppanelheight`, `bottompanelheight`, `leftpanelwidth`, `rightpanelwidth`.
- `doublehgradient` ‘Double’ horizontal gradient. The page background is constructed using the `\dblhgradrule` command. See the description of `vgradient` concerning panels. Parameters used: `stripes`, `gradmidpoint`, `firstgradprogression`, `secondgradprogression`, `startcolor`, `startcolordef`, `midcolor`, `midcolordef`, `endcolor`, `endcolordef`, `hpanels`, `autopanelcolor`, `bottompanelcolor`, `leftpanelcolor`, `rightpanelcolor`, `toppanelcolordef`, `bottompanelcolordef`, `leftpanelcolordef`, `rightpanelcolordef`, `toppanelheight`, `bottompanelheight`, `leftpanelwidth`, `rightpanelwidth`.

Now, a list of all parameters and their meaning. In the following,

- `<n>` denotes a (calc expression for a) nonnegative integer
- `<i>` denotes a (calc expression for an) integer
- `<r>` denotes a fixed-point number
- `<l>` denotes a (calc expression for a) length
- `<c>` denotes the name of a defined color
- `<cm>` denotes a valid color model name (in the sense of the color package)
- `<cd>` denotes a valid color definition (in the sense of the color package) wrt a given `<cm>` parameter
- `<t>` denotes a ‘truth value’ in the sense of the `ifthen` package: either true or false. As usual for `keyval`, if `=<t>` is omitted, the default true is assumed.

Parameter	Meaning
<code>stripes=&lt;n&gt;</code>	Set the <code>&lt;stripes&gt;</code> parameter of gradient rules to <code>&lt;n&gt;</code> . Default: <code>\bgndstripes</code> Used by: <code>vgradient</code> , <code>hgradient</code> , <code>doublevgradient</code> , <code>doublehgradient</code>
<code>gradmidpoint=&lt;r&gt;</code>	Set the <code>&lt;midpoint&gt;</code> parameter of double gradient rules to <code>&lt;r&gt;</code> . Default: <code>\bgndgradmidpoint</code> Used by: <code>doublevgradient</code> , <code>doublehgradient</code>
<code>firstgradprogression=&lt;i&gt;</code>	Set the first gradient progression of gradient rules to <code>&lt;i&gt;</code> . Default: <code>\bgndfirstgradprogression</code> Used by: <code>vgradient</code> , <code>hgradient</code> , <code>doublevgradient</code> , <code>doublehgradient</code>
<code>secondgradprogression=&lt;i&gt;</code>	Set the second gradient progression of double gradient rules to <code>&lt;i&gt;</code> . Default: <code>\bgndsecondgradprogression</code> Used by: <code>doublevgradient</code> , <code>doublehgradient</code>
<code>startcolor=&lt;c&gt;</code>	Set the <code>&lt;startcolor&gt;</code> parameter of gradient rules to <code>&lt;c&gt;</code> . Default: If neither <code>startcolor</code> nor <code>startcolordef</code> is given, the color <code>bgndstartcolor</code> is used as <code>&lt;startcolor&gt;</code> . Used by: <code>vgradient</code> , <code>hgradient</code> , <code>doublevgradient</code> , <code>doublehgradient</code> Overwrites: <code>startcolordef</code>
<code>startcolordef={&lt;cm&gt;}{&lt;cd&gt;}</code>	Set the <code>&lt;startcolor&gt;</code> parameter of gradient rules to color <code>foo</code> , which is obtained by <code>\definecolor{foo}{&lt;cm&gt;}{&lt;cd&gt;}</code> . Note that the two pairs of curly braces are mandatory. Default: If neither <code>startcolor</code> nor <code>startcolordef</code> is given, the color <code>bgndstartcolor</code> is used as <code>&lt;startcolor&gt;</code> . Used by: <code>vgradient</code> , <code>hgradient</code> , <code>doublevgradient</code> , <code>doublehgradient</code> Overwrites: <code>startcolor</code>
<code>endcolor=&lt;c&gt;</code>	Set the <code>&lt;endcolor&gt;</code> parameter of gradient rules to <code>&lt;c&gt;</code> . Default: If neither <code>endcolor</code> nor <code>endcolordef</code> is given, the color <code>bgndendcolor</code> is used as <code>&lt;endcolor&gt;</code> . Used by: <code>vgradient</code> , <code>hgradient</code> , <code>doublevgradient</code> , <code>doublehgradient</code> Overwrites: <code>endcolordef</code>

`endcolordef={<cm>}{<cd>}`

Set the `<endcolor>` parameter of gradient rules to color `foo`, which is obtained by `\definecolor{foo}{<cm>}{<cd>}`. Note that the two pairs of curly braces are mandatory. Default: If neither `endcolor` nor `endcolordef` is given, the color `bgndendcolor` is used as `<endcolor>`. Used by: `vgradient`, `hgradient`, `doublevgradient`, `doublehgradient` Overwrites: `endcolor`

`midcolor=<c>`

Set the `<midcolor>` parameter of double gradient rules to `<c>`. Default: If neither `midcolor` nor `midcolordef` is given, the color `bgndmidcolor` is used as `<midcolor>`. Used by: `doublevgradient`, `doublehgradient` Overwrites: `midcolordef`

`midcolordef={<cm>}{<cd>}`

Set the `<midcolor>` parameter of double gradient rules to color `foo`, which is obtained by `\definecolor{foo}{<cm>}{<cd>}`. Note that the two pairs of curly braces are mandatory. Default: If neither `midcolor` nor `midcolordef` is given, the color `bgndmidcolor` is used as `<midcolor>`. Used by: `doublevgradient`, `doublehgradient` Overwrites: `midcolor`

`hpanels=<t>`

Specifies the ‘direction’ of panels produced. `hpanels=true` means the top and bottom panel span the full width of the screen. In the space left in the middle, the left panel, the background itself, and the right panel are displayed. `hpanels=false` means the left and right panel span the full height of the screen. In the space left in the middle, the top panel, the background itself, and the bottom panel are displayed. Default: `hpanels=true` is the default for `plain`, `hgradient` and `doublehgradient`. `hpanels=false` is the default for `vgradient` and `doublevgradient`. Used by: `plain`, `vgradient`, `hgradient`, `doublevgradient`, `doublehgradient`

`autopanelwidth=<t>`

Specifies whether the default values of the parameters `toppanelheight`, `bottompanelheight`, `leftpanelwidth`, `rightpanelwidth` should be calculated automatically from the contents of declared panels. The automatism used is analogous to that of `\DeclarePanel*`. Note that for panel arrangement, both the width and the height of all declared panels are overwritten. If you don't want this, calculate the panel parameters yourself and set `autopanelwidth=false`. In this case, the current panel dimensions of declared panels are used as defaults for `toppanelheight`, `bottompanelheight`, `leftpanelwidth`, `rightpanelwidth`. Default: true. Used by: `plain`, `vgradient`, `hgradient`, `doublevgradient`, `doublehgradient`

`toppanelheight=<l>`

`bottompanelheight=<l>`

`leftpanelwidth=<l>`

`rightpanelwidth=<l>`

Set the height/width of the space left for the top/bottom/left/right panel to  $l_i$ . Note that the remaining dimensions of panels, for instance the width of the top panel, are always calculated automatically, depending on the setting of the `hpanels` parameter. Default: If a respective panel has been defined using `\DeclarePanel`, the default used depends on the setting of the `autopanel` parameter. If `autopanel=true`, the correct dimension is calculated from the contents of the panel. The respective one of `\toppanelheight`, `\bottompanelheight`, `\leftpanelwidth`, `\rightpanelwidth` is overwritten with the result. If `autopanel=false`, then the respective setting of `\toppanelheight`, `\bottompanelheight`, `\leftpanelwidth`, `\rightpanelwidth` is taken as the default. If a panel has not been declared, the appropriate one of `\bgndtoppanelheight`, `\bgndbottompanelheight`, `\bgndleftpanelwidth`, `\bgndrightpanelwidth` is used as default. Used by: `plain`, `vgradient`, `hgradient`, `doublevgradient`, `doublehgradient`

`toppanelcolor=<c>`  
`bottompanelcolor=<c>`  
`leftpanelcolor=<c>`  
`rightpanelcolor=<c>`

Set the color of the space left for the top/bottom/left/right panel to  $c_i$ . Default: The standard colors `toppanelcolor`, `bottompanelcolor`, `leftpanelcolor`, `rightpanelcolor` are used as defaults. Used by: `plain`, `vgradient`, `hgradient`, `doublevgradient`, `doublehgradient` Overwrites: `toppanelcolordef` `bottompanelcolordef` `leftpanelcolordef` `rightpanelcolordef`

`toppanelcolordef={<cm>}{<cd>}`  
`bottompanelcolordef={<cm>}{<cd>}`  
`leftpanelcolordef={<cm>}{<cd>}`

`rightpanelcolordef={<cm>}{<cd>}` Set the color of the space left for the top/bottom/left/right panel to color foo, which is obtained by `\definecolor{foo}{<cm>}{<cd>}`. Note that the two pairs of curly braces are mandatory. Default: See the description of top/bottom/left/rightpanelcolor. Used by: plain, vgradient, hgradient, doublevgradient, doublehgradient Overwrites: toppanelcolor bottompanelcolor leftpanelcolor rightpanelcolor

```

1364 \newcommand{\backgroundstyle}[2] []
1365 {%
1366   \replacecolor{startcolor@TP}{bgndstartcolor}%           Initialize the intern
1367   \replacecolor{midcolor@TP}{bgndmidcolor}%               of parameters to thei
1368   \replacecolor{endcolor@TP}{bgndendcolor}%
1369   \replacecolor{bgndtoppanelcolor@TP}{toppanelcolor}%
1370   \replacecolor{bgndbottompanelcolor@TP}{bottompanelcolor}%
1371   \replacecolor{bgndleftpanelcolor@TP}{leftpanelcolor}%
1372   \replacecolor{bgndrightpanelcolor@TP}{rightpanelcolor}%
1373   \let\firstgradprogression@TP=\bgndfirstgradprogression
1374   \let\secondgradprogression@TP=\bgndsecondgradprogression%
1375   \setcounter{stripes@TP}{\bgndstripes}%
1376   \let\gradmidpoint@TP=\bgndgradmidpoint
1377   \let\bgndtoppanelheight@TP=\empty%                       The panel dimensions depend on other param
1378   \let\bgndbottompanelheight@TP=\empty%                   set directly, defaults are calculated after
1379   \let\bgndleftpanelwidth@TP=\empty%                       parameters.
1380   \let\bgndrightpanelwidth@TP=\empty
1381   \let\hpanelsvalue@TP=\empty
1382   \setboolean{autopanel@TP}{true}%
1383   \csname set#2bgnd@TP\endcsname{#1}%                       Execute the style-specific command which d
1384 }

```

Background-specific default values.

Default number of stripes for gradient page backgrounds.

```
1385 \newcommand{\bgndstripes}{10}
```

Default position of the ‘middle’ color of a double gradient.

```
1386 \newcommand{\bgndgradmidpoint}{.5}
```

Default gradient progression for page backgrounds (single gradients or first part of double gradients).

```
1387 \newcommand{\bgndfirstgradprogression}{1}
```

Default gradient progression for page backgrounds (second part of double gradients).

```
1388 \newcommand{\bgndsecondgradprogression}{1}
```

Default height/width of the space left for the top/bottom/left/right panel, in case no panel in the respective position has been declared. Otherwise, the defaults are taken from `\toppanelheight`, `\bottompanelheight`, `\leftpanelwidth`, `\rightpanelwidth` or calculated automatically, depending on the setting of the `autopanelwidth` parameter. Note that the remaining dimensions of panels, for instance the width of the top panel, are always calculated automatically, depending on the setting of the `hpanels` parameter.

```
1389 \newcommand{\bgndtoppanelheight}{Opt}
1390
1391 \newcommand{\bgndbottompanelheight}{Opt}
1392
1393 \newcommand{\bgndleftpanelwidth}{Opt}
1394
1395 \newcommand{\bgndrightpanelwidth}{Opt}
```

Internal names for parameter values.

```
1396 \newcommand{\bgndtoppanelheight@TP}{Opt}
1397
1398 \newcommand{\bgndtoppanelwidth@TP}{Opt}
1399
1400 \newcommand{\bgndbottompanelheight@TP}{Opt}
1401
1402 \newcommand{\bgndbottompanelwidth@TP}{Opt}
1403
1404 \newcommand{\bgndleftpanelheight@TP}{Opt}
1405
1406 \newcommand{\bgndleftpanelwidth@TP}{Opt}
1407
1408 \newcommand{\bgndrightpanelheight@TP}{Opt}
1409
1410 \newcommand{\bgndrightpanelwidth@TP}{Opt}
1411
1412 \newboolean{hpanels@TP}
1413
1414 \newboolean{autopanelwidth@TP}
```

The following commands define the keys for setting the parameters using the `keyval` package.

```
1415 \define@key{bgnd@TP}{stripes}{\setcounter{stripes@TP}{#1}}
1416
1417
1418 \define@key{bgnd@TP}{startcolor}{\replacecolor{startcolor@TP}{#1}}
1419
1420 \define@key{bgnd@TP}{startcolordef}{\definecolor{startcolor@TP}{#1}}
1421
1422
1423 \define@key{bgnd@TP}{midcolor}{\replacecolor{midcolor@TP}{#1}}
1424
1425 \define@key{bgnd@TP}{midcolordef}{\definecolor{midcolor@TP}{#1}}
1426
```

```

1427
1428 \define@key{bgnd@TP}{endcolor}{\replacecolor{endcolor@TP}{#1}}
1429
1430 \define@key{bgnd@TP}{endcolordef}{\definecolor{endcolor@TP}{#1}}
1431
1432
1433 \define@key{bgnd@TP}{gradmidpoint}{\edef\gradmidpoint@TP{#1}}
1434
1435
1436 \define@key{bgnd@TP}{firstgradprogression}{\def\firstgradprogression@TP{#1}}
1437
1438 \define@key{bgnd@TP}{secondgradprogression}{\def\secondgradprogression@TP{#1}}
1439
1440
1441 \define@key{bgnd@TP}{hpanels}[true]{\def\hpanelsvalue@TP{#1}}
1442
1443 \define@key{bgnd@TP}{autopanel}[true]{\setboolean{autopanel@TP}{#1}}
1444
1445
1446 \define@key{bgnd@TP}{toppanelcolor}{\replacecolor{bgndtoppanelcolor@TP}{#1}}
1447
1448 \define@key{bgnd@TP}{toppanelcolordef}{\definecolor{bgndtoppanelcolor@TP}{#1}}
1449
1450
1451 \define@key{bgnd@TP}{bottompanelcolor}{\replacecolor{bgndbottompanelcolor@TP}{#1}}
1452
1453 \define@key{bgnd@TP}{bottompanelcolordef}{\definecolor{bgndbottompanelcolor@TP}{#1}}
1454
1455
1456 \define@key{bgnd@TP}{leftpanelcolor}{\replacecolor{bgndleftpanelcolor@TP}{#1}}
1457
1458 \define@key{bgnd@TP}{leftpanelcolordef}{\definecolor{bgndleftpanelcolor@TP}{#1}}
1459
1460
1461 \define@key{bgnd@TP}{rightpanelcolor}{\replacecolor{bgndrightpanelcolor@TP}{#1}}
1462
1463 \define@key{bgnd@TP}{rightpanelcolordef}{\definecolor{bgndrightpanelcolor@TP}{#1}}
1464
1465
1466 \define@key{bgnd@TP}{toppanelheight}{\mklength@TP{\bgndtoppanelheight@TP}{#1}}
1467
1468 \define@key{bgnd@TP}{bottompanelheight}{\mklength@TP{\bgndbottompanelheight@TP}{#1}}
1469
1470 \define@key{bgnd@TP}{leftpanelwidth}{\mklength@TP{\bgndleftpanelwidth@TP}{#1}}
1471
1472 \define@key{bgnd@TP}{rightpanelwidth}{\mklength@TP{\bgndrightpanelwidth@TP}{#1}}

```



### 3.4.4 Implementation of `\backgroundstyle`

In this box, the constructed background is stored. This box is placed behind every page at `\shipout` time by the kernel (see below).

1473 `\newbox\bgndbox@TP`

1474 `\setbox\bgndbox@TP\null%` Default: Empty.

`\mkpanels@TP{<command>}` adds the panels to the main page background. The main page background should be produced by the command `<command>`, which is given the width and height of the central area as arguments.

1475 `\newcommand{\mkpanels@TP}[1]`

1476 `{%`

1477 `\ifthenelse{\boolean{hpanels@TP}}%` 'horizontal' panels?

1478 `{% Yes. Vertically align top panel, center area with left and right panels, and bottom panel.`

1479 `\vbox%` A vbox with `\offinterline`

1480 `{%` horizontal panels with

1481 `\offinterlineskip`

1482 `\ifthenelse{\lengthtest{\bgndtoppanelheight@TP=0pt}}%` Should top panel be cre

1483 `{}% No.`

1484 `{%`

1485 `\hbox{%`

1486 `\setcolor@TP{bgndtoppanelcolor@TP}%`

1487 `\rule{bgndtoppanelwidth@TP}{bgndtoppanelheight@TP}%` Make horizontal colored

1488 `}}%`

1489 `}%`

1490 `\hbox%` Make 'background center

1491 `{%`

1492 `\ifthenelse{\lengthtest{\bgndleftpanelwidth@TP=0pt}}%` Should left panel be cr

1493 `{}% No.`

1494 `{{%`

1495 `\setcolor@TP{bgndleftpanelcolor@TP}%`

1496 `\rule{bgndleftpanelwidth@TP}{bgndleftpanelheight@TP}%` Make vertical colored a

1497 `}}%`

1498 `#1%` Make main background ob

1499 `{\bgndtoppanelwidth@TP-\bgndleftpanelwidth@TP-\bgndrightpanelwidth@TP}% Calculate remaini`

1500 `{\bgndleftpanelheight@TP}%`

1501 `\ifthenelse{\lengthtest{\bgndrightpanelwidth@TP=0pt}}%` Should right panel be c

1502 `{}% No.`

1503 `{{%`

1504 `\setcolor@TP{bgndrightpanelcolor@TP}%`

1505 `\rule{bgndrightpanelwidth@TP}{bgndrightpanelheight@TP}%` Make vertical colored a

1506 `}}%`

1507 `}% matches \hbox{%`

1508 `\ifthenelse{\lengthtest{\bgndbottompanelheight@TP=0pt}}%` Should bottom panel be

1509 `{}% No.`

1510 `{%`

1511 `\hbox`

1512 `{%`

1513 `\setcolor@TP{bgndbottompanelcolor@TP}%`

1514 `\rule{bgndbottompanelwidth@TP}{bgndbottompanelheight@TP}%` Make horizontal colored

```

1515         }}%
1516     }%
1517     }% matches \vbox{%
1518 }% matches \ifthenelse{\boolean{hpanels@TP}}{%
1519 {% No. Horizontally align left panel, center area with top and bottom panels, and right panel
1520 \ifthenelse{\lengthtest{\bgndleftpanelwidth@TP=0pt}}% Should left panel be cr
1521 }% No.
1522 {%
1523     \setcolor@TP{bgndleftpanelcolor@TP}%
1524     \rule{\bgndleftpanelwidth@TP}{\bgndleftpanelheight@TP}% Make vertical colored a
1525 }}%
1526 \vbox% A vbox with \offinterli
1527 {% the horizontal panels w
1528     \offinterlineskip
1529     \ifthenelse{\lengthtest{\bgndtoppanelheight@TP=0pt}}% Should top panel be cre
1530 }% No.
1531 {%
1532     \hbox%
1533     {%
1534         \setcolor@TP{bgndtoppanelcolor@TP}%
1535         \rule{\bgndtoppanelwidth@TP}{\bgndtoppanelheight@TP}% Make horizontal colored
1536     }}%
1537     }%
1538     \hbox% Make main background ob
1539     {%
1540         #1%
1541         {\bgndtoppanelwidth@TP}%
1542         {\bgndleftpanelheight@TP-\bgndtoppanelheight@TP-\bgndbottompanelheight@TP}% Calculate s
1543     }%
1544     \ifthenelse{\lengthtest{\bgndbottompanelheight@TP=0pt}}% Should bottom panel be
1545 }% No.
1546 {%
1547     \hbox%
1548     {%
1549         \setcolor@TP{bgndbottompanelcolor@TP}%
1550         \rule{\bgndbottompanelwidth@TP}{\bgndbottompanelheight@TP}% Make horizontal colored
1551     }}%
1552     }%
1553 }% matches \vbox{%
1554 \ifthenelse{\lengthtest{\bgndrightpanelwidth@TP=0pt}}% Should right panel be c
1555 }% No.
1556 {%
1557     \setcolor@TP{bgndrightpanelcolor@TP}%
1558     \rule{\bgndrightpanelwidth@TP}{\bgndrightpanelheight@TP}% Make vertical colored area.
1559 }}%
1560 }% matches second argument of \ifthenelse{\boolean{hpanels@TP}}
1561 }% matches \newcommand{\mkpanels@TP}[1]{%

```

For those background styles which use panels, `\initpanels@TP{<hpanels>}` sets all panel-related parameters depending on the options and defaults. `jpanelsi`

gives the background style dependent default of the hpanels option.

```

1562 \newcommand{\initpanels@TP}[1]%
1563 {%
1564   \ifx\hpanelsvalue@TP\empty%
1565     \setboolean{hpanels@TP}{#1}%
1566   \else
1567     \setboolean{hpanels@TP}{\hpanelsvalue@TP}%
1568   \fi
1569   \ifthenelse{\boolean{hpanels@TP}}%
1570   {% Yes. Horizontal panels are 'outer', vertical panels are 'inner'.
1571     \let\bgndtoppanelwidth@TP=\TPpagewidth%
1572     \let\bgndbottompanelwidth@TP=\TPpagewidth%
1573     \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}%
1574     {% No. Guess default.
1575       \ifx\toppanelcontents@TP\empty%
1576         \mklength@TP{\bgndtoppanelheight@TP}{\bgndtoppanelheight}% Use background-specific default.
1577       \else
1578         \ifthenelse{\boolean{autopanel@TP}}%
1579         {% Yes.
1580           \calcvdimen@TP{\bgndtoppanelheight@TP}{\bgndtoppanelwidth@TP}% Measure the height of
1581           {\toppanelcontents@TP}%
1582           \let\toppanelheight=\bgndtoppanelheight@TP% Overwrite panel settings.
1583           \let\toppanelwidth=\bgndtoppanelwidth@TP%
1584           \def\toppanelshift{Opt}%
1585         }
1586         {% No
1587           \mklength@TP{\bgndtoppanelheight@TP}{\toppanelheight}% Use panel-specific default.
1588         }%
1589       \fi
1590     }% matches \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}
1591   {% Yes.
1592     \let\toppanelheight=\bgndtoppanelheight@TP%
1593   }
1594   \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}%
1595   {% No. Guess default.
1596     \ifx\bottompanelcontents@TP\empty%
1597       \mklength@TP{\bgndbottompanelheight@TP}{\bgndbottompanelheight}% Use background-specific default.
1598     \else
1599       \ifthenelse{\boolean{autopanel@TP}}%
1600       {% Yes.
1601         \calcvdimen@TP{\bgndbottompanelheight@TP}{\bgndbottompanelwidth@TP}% Measure the height of
1602         {\bottompanelcontents@TP}%
1603         \let\bottompanelheight=\bgndbottompanelheight@TP% Overwrite panel settings.
1604         \let\bottompanelwidth=\bgndbottompanelwidth@TP%
1605         \def\bottompanelshift{Opt}%
1606       }
1607       {% No
1608         \mklength@TP{\bgndbottompanelheight@TP}{\bottompanelheight}% Use panel-specific default.
1609       }%

```

```

1610     \fi
1611     }% matches \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}
1612     {% Yes.
1613     \let\bottompanelheight=\bgndbottompanelheight@TP%   Overwrite panel settings - use user-s
1614     }
1615     \mklength@TP{\bgndleftpanelheight@TP}%               Calculate remaining space in the cent
1616     {\TPpageheight-\bgndtoppanelheight@TP-\bgndbottompanelheight@TP}%
1617     \let\bgndrightpanelheight@TP=\bgndleftpanelheight@TP% Height of left and right panels is eq
1618     \ifthenelse{\equal{\bgndleftpanelwidth@TP}{}}%       Has the left panel width been set?
1619     {% No. Guess default.
1620     \ifx\leftpanelcontents@TP\empty%                     Is the panel defined?
1621     \mklength@TP{\bgndleftpanelwidth@TP}{\bgndleftpanelwidth}% Use background-specific defa
1622     \else
1623     \ifthenelse{\boolean{autopanel@TP}}%                 Calculate panel dimensions?
1624     {% Yes.
1625     \calchdimen@TP{\bgndleftpanelwidth@TP}{\bgndleftpanelheight@TP}% Measure the 'optimal'
1626     {\leftpanelcontents@TP}%
1627     \let\leftpanelheight=\bgndleftpanelheight@TP%   Overwrite panel settings.
1628     \let\leftpanelwidth=\bgndleftpanelwidth@TP%
1629     \let\leftpanelraise=\bgndbottompanelheight@TP% Left panel is raised above bottom pan
1630     }
1631     {% No
1632     \mklength@TP{\bgndleftpanelwidth@TP}{\leftpanelwidth}% Use panel-specific default.
1633     }%
1634     \fi
1635     }% matches \ifthenelse{\equal{\bgndleftpanelwidth@TP}{}}
1636     {% Yes.
1637     \let\leftpanelwidth=\bgndleftpanelwidth@TP%       Overwrite panel settings - use user-s
1638     }
1639     \ifthenelse{\equal{\bgndrightpanelwidth@TP}{}}%     Has the right panel width been set?
1640     {% No. Guess default.
1641     \ifx\rightpanelcontents@TP\empty%                 Is the panel defined?
1642     \mklength@TP{\bgndrightpanelwidth@TP}{\bgndrightpanelwidth}% Use background-specific de
1643     \else
1644     \ifthenelse{\boolean{autopanel@TP}}%               Calculate panel dimensions?
1645     {% Yes.
1646     \calchdimen@TP{\bgndrightpanelwidth@TP}{\bgndrightpanelheight@TP}% Measure 'optimal'
1647     {\rightpanelcontents@TP}%
1648     \let\rightpanelheight=\bgndrightpanelheight@TP% Overwrite panel settings.
1649     \let\rightpanelwidth=\bgndrightpanelwidth@TP%
1650     \let\rightpanelraise=\bgndbottompanelheight@TP% Right panel is raised above bottom pa
1651     }
1652     {% No
1653     \mklength@TP{\bgndrightpanelwidth@TP}{\rightpanelwidth}% Use panel-specific default
1654     }%
1655     \fi
1656     }% matches \ifthenelse{\equal{\bgndrightpanelwidth@TP}{}}
1657     {% Yes.
1658     \let\rightpanelwidth=\bgndrightpanelwidth@TP%     Overwrite panel settings - use user-s
1659     }

```

```

1660     }% matches \ifthenelse{\boolean{hpanels@TP}}
1661     {% No. Vertical panels are 'outer', horizontal panels are 'inner'.
1662     \let\bgnleftpanelheight@TP=\TPpageheight%           Full height for vertical panels.
1663     \let\bgnrightpanelheight@TP=\TPpageheight%
1664     \ifthenelse{\equal{\bgnleftpanelwidth@TP}{}}%      Has the left panel width been set?
1665     {% No. Guess default.
1666     \ifx\leftpanelcontents@TP\empty%                   Is the panel defined?
1667     \mklength@TP{\bgnleftpanelwidth@TP}{\bgnleftpanelwidth}% Use background-specific default
1668     \else
1669     \ifthenelse{\boolean{autopanelwidth@TP}}%          Calculate panel dimensions?
1670     {% Yes.
1671     \calchdimen@TP{\bgnleftpanelwidth@TP}{\bgnleftpanelheight@TP}% Measure the 'optimal'
1672     {\leftpanelcontents@TP}%
1673     \let\leftpanelheight=\bgnleftpanelheight@TP% Overwrite panel settings.
1674     \let\leftpanelwidth=\bgnleftpanelwidth@TP%
1675     \def\leftpanelraise{0pt}%                           Left panel spans the whole left part of
1676     }
1677     {% No
1678     \mklength@TP{\bgnleftpanelwidth@TP}{\leftpanelwidth}% Use panel-specific default.
1679     }%
1680     \fi
1681     }% matches \ifthenelse{\equal{\bgnleftpanelwidth@TP}{}}
1682     {% Yes.
1683     \let\leftpanelwidth=\bgnleftpanelwidth@TP%           Overwrite panel settings - use user-sup
1684     }
1685     \ifthenelse{\equal{\bgnrightpanelwidth@TP}{}}%      Has the right panel width been set?
1686     {% No. Guess default.
1687     \ifx\rightpanelcontents@TP\empty%                   Is the panel defined?
1688     \mklength@TP{\bgnrightpanelwidth@TP}{\bgnrightpanelwidth}% Use background-specific default
1689     \else
1690     \ifthenelse{\boolean{autopanelwidth@TP}}%          Calculate panel dimensions?
1691     {% Yes.
1692     \calchdimen@TP{\bgnrightpanelwidth@TP}{\bgnrightpanelheight@TP}% Measure 'optimal'
1693     {\rightpanelcontents@TP}%
1694     \let\rightpanelheight=\bgnrightpanelheight@TP% Overwrite panel settings.
1695     \let\rightpanelwidth=\bgnrightpanelwidth@TP%
1696     \def\rightpanelraise{0pt}%                           Right panel spans the whole left part
1697     }
1698     {% No
1699     \mklength@TP{\bgnrightpanelwidth@TP}{\rightpanelwidth}% Use panel-specific default
1700     }%
1701     \fi
1702     }% matches \ifthenelse{\equal{\bgnrightpanelwidth@TP}{}}
1703     {% Yes.
1704     \let\rightpanelwidth=\bgnrightpanelwidth@TP%           Overwrite panel settings - use user-s
1705     }
1706     \mklength@TP{\bgndtoppanelwidth@TP}%               Calculate remaining space in the center
1707     {\TPpagewidth-\bgnleftpanelwidth@TP-\bgnrightpanelwidth@TP}%
1708     \let\bgndbottompanelwidth@TP=\bgndtoppanelwidth@TP% Width of top and bottom panels is equal
1709     \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}%      Has the top panel height been set?

```

```

1710     {% No. Guess default.
1711     \ifx\toppanelcontents@TP\empty%                Is the panel defined?
1712         \mklength@TP{\bgndtoppanelheight@TP}{\bgndtoppanelheight}% Use background-specific defa
1713     \else
1714     \ifthenelse{\boolean{autopanel@TP}}%           Calculate panel dimensions?
1715     {% Yes.
1716         \calcvdimen@TP{\bgndtoppanelheight@TP}{\bgndtoppanelwidth@TP}% Measure the height of
1717         {\toppanelcontents@TP}%
1718         \let\toppanelheight=\bgndtoppanelheight@TP% Overwrite panel settings.
1719         \let\toppanelwidth=\bgndtoppanelwidth@TP%
1720         \let\toppanelshift=\bgndleftpanelwidth@TP% Shift top panel to the right of left pa
1721     }
1722     {% No
1723         \mklength@TP{\bgndtoppanelheight@TP}{\toppanelheight}% Use panel-specific default.
1724     }%
1725     \fi
1726     }% matches \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}
1727     {% Yes.
1728         \let\toppanelheight=\bgndtoppanelheight@TP% Overwrite panel settings - use user-sup
1729     }
1730     \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}% Has the bottom panel height been se
1731     {% No. Guess default.
1732     \ifx\bottompanelcontents@TP\empty%            Is the panel defined?
1733         \mklength@TP{\bgndbottompanelheight@TP}{\bgndbottompanelheight}% Use background-specific
1734     \else
1735     \ifthenelse{\boolean{autopanel@TP}}%           Calculate panel dimensions?
1736     {% Yes.
1737         \calcvdimen@TP{\bgndbottompanelheight@TP}{\bgndbottompanelwidth@TP}% Measure the heig
1738         {\bottompanelcontents@TP}%
1739         \let\bottompanelheight=\bgndbottompanelheight@TP% Overwrite panel settings.
1740         \let\bottompanelwidth=\bgndbottompanelwidth@TP%
1741         \let\bottompanelshift=\bgndleftpanelwidth@TP% Shift bottom panel to the right of
1742     }
1743     {% No
1744         \mklength@TP{\bgndbottompanelheight@TP}{\bottompanelheight}% Use panel-specific def
1745     }%
1746     \fi
1747     }% matches \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}
1748     {% Yes.
1749         \let\bottompanelheight=\bgndbottompanelheight@TP% Overwrite panel settings - use user
1750     }
1751     }% matches second argument of \ifthenelse{\boolean{hpanel@TP}}
1752     }% matches \newcommand{\initpanels@TP}[1]{

```

Make an ‘invisible’ rule.

```

1753 \newcommand{\phantomrule@TP}[2]{\rule{0pt}{#2}\rule{#1}{0pt}}

```

Implementations of individual background styles.

```

1754 \newcommand{\setnonebgnd@TP}[1]%                 Implementation of the background style ‘none’.
1755 {\global\setbox\bgndbox@TP=\null}%             Just produce an empty box.
1756

```

```

1757
1758 \newcommand{\setplainbgnd@TP}[1]%           Implementation of the background style ‘plain’.
1759 {%
1760   \setkeys{bgnd@TP}{#1}%                     Evaluate parameters.
1761   \initpanels@TP{true}%                       Initialize panel parameters.
1762   \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\phantomrule@TP}}% Make panels only.
1763   }%
1764
1765
1766 \newcommand{\setvgradientbgnd@TP}[1]%       Implementation of the background style ‘vgradient’.
1767 {%
1768   \setkeys{bgnd@TP}{#1}%                     Evaluate parameters.
1769   \initpanels@TP{false}%                     Initialize panel parameters.
1770   \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@vgradrule@TP}}% Make background box.
1771   }%
1772
1773 \newcommand{\sethgradientbgnd@TP}[1]%       Implementation of the background style ‘hgradient’.
1774 {%
1775   \setkeys{bgnd@TP}{#1}%                     Evaluate parameters.
1776   \initpanels@TP{true}%                     Initialize panel parameters.
1777   \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@hgradrule@TP}}% Make background box.
1778   }%
1779
1780
1781 \newcommand{\setdoublevgradientbgnd@TP}[1]% Implementation of the background style ‘doublevgradient’.
1782 {%
1783   \setkeys{bgnd@TP}{#1}%                     Evaluate parameters.
1784   \initpanels@TP{false}%                     Initialize panel parameters.
1785   \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@dblvggradrule@TP}}% Make background box.
1786   }%
1787
1788
1789 \newcommand{\setdoublehgradientbgnd@TP}[1]% Implementation of the background style ‘doublehgradient’.
1790 {%
1791   \setkeys{bgnd@TP}{#1}%                     Evaluate parameters.
1792   \initpanels@TP{true}%                     Initialize panel parameters.
1793   \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@dblhgradrule@TP}}% Make background box.
1794   }%

```

`\hpagecolor[<start>]{<end>}` is provided here for compatibility with background.sty from PPower4. It sets a horizontal gradient background. See the PPower4 documentation on the meaning of the arguments (which is quite confusing).

```

1795 \providecommand{\hpagecolor}[2] []
1796 {%
1797   \ifthenelse{\equal{#1}{}}
1798   {\colorbetween{pendcolor}{#2}{white}\backgroundstyle[startcolor=#2,endcolor=pendcolor]{hgradient}}
1799   {\backgroundstyle[startcolor=#1,endcolor=#2]{hgradient}}%
1800   }%

```

`\vpagecolor[<start>]{<end>}` is provided here for compatibility with `background.sty` from PPower4. It sets a vertical gradient background. See the PPower4 documentation on the meaning of the arguments (which is quite confusing).

```
1801 \providecommand{\vpagecolor}[2] []
1802 {%
1803   \ifthenelse{\equal{#1}{}}
1804     {\colorbetween{ppendcolor}{#2}{white}\backgroundstyle[startcolor=#2,endcolor=ppendcolor]{vgra
1805     {\backgroundstyle[startcolor=#1,endcolor=#2]{vgradient}}%
1806   }
```

### 3.5 Panels

The following code is rather preliminary and provides only the very basics for constructing panels. If you're using a document class or package which allows to do this or know how to achieve it using fancy headers, don't even consider using the following.

Some configurable panel parameters.

Margin around panels (space [on all sides] between beginning of background and panel contents).

```
1807 \mklength@TP{\panelmargin}{\fboxsep}
```

Dimensions of top panel. Note that parts or all of these dimensions might be overwritten by `\DeclarePanel*` or using `\backgroundstyle` with specific settings.

```
1808 \newcommand{\toppanelwidth}{\TPpagewidth}%      Width.
1809 \newcommand{\toppanelheight}{\TPpageheight/5}% Height.
1810 \newcommand{\toppanelshift}{0pt}%              Space between left screen edge and left edge of
```

Dimensions of bottom panel.

```
1811 \newcommand{\bottompanelwidth}{\TPpagewidth}%  Width.
1812 \newcommand{\bottompanelheight}{\TPpageheight/5}% Height.
1813 \newcommand{\bottompanelshift}{0pt}%           Space between left screen edge and left edge
```

Dimensions of left panel.

```
1814 \newcommand{\leftpanelwidth}{\TPpagewidth/5}% Width.
1815 \newcommand{\leftpanelheight}{\TPpageheight}% Height.
1816 \newcommand{\leftpanelraise}{0pt}%            Space between bottom screen edge and bottom edge
```

Dimensions of right panel.

```
1817 \newcommand{\rightpanelwidth}{\TPpagewidth/5}% Width.
1818 \newcommand{\rightpanelheight}{\TPpageheight}% Height.
1819 \newcommand{\rightpanelraise}{0pt}%           Space between bottom screen edge and bottom edge
```

Some internal panel parameters.

Storage for panel contents.

```
1820 \newcommand*{\toppanelcontents@TP}{}
1821 \newcommand*{\bottompanelcontents@TP}{}
1822 \newcommand*{\leftpanelcontents@TP}{}
1823 \newcommand*{\rightpanelcontents@TP}{}

```



### 3.5.1 Panel-specific user level commands

`\DeclarePanel` `\DeclarePanel[<name>]{<pos>}{<contents>}` declares the contents `<contents>` of the panel at position `<pos>`. Afterwards, on every page the panel contents are set in a parbox of dimensions and position specified by `<pos>panelwidth`, `<pos>panelheight`, `\panelmargin` and `<pos>panelshift` for top and bottom panels and `<pos>panelraise` for left and right panels. The parbox is constructed anew on every page, so all changes influencing panel contents or parameters (like a `\thepage` in the panel contents) are respected.

The panel contents are set in color `<pos>paneltextcolor`. There is another standard color `<pos>panelcolor`, which is however not activated by `\DeclarePanel` but by selecting an appropriate background style. Note that pages are constructed as follows: first the page background, then the panels, and then the page contents. Hence, panels overwrite the background and the page contents overwrite the panels. The user is supposed to make sure themselves that there is enough space left on the page for the panels (document class specific settings). The panel declaration is global. A panel can be ‘undeclared’ by using `\DeclarePanel{<pos>}{}`.

There is a starred version which will (try to) automatically calculate the ‘flexible’ dimension of each panel. For top and bottom panels this is the height, for left and right panels this is the width. Make sure the panel contents are ‘valid’ at the time `\DeclarePanel*` is called so the calculation can be carried out in a meaningful way. While the automatic calculation of the height of top and bottom panels is trivial (using `\settoheight`), there is a sophisticated procedure for calculating a ‘good’ width for the parbox containing the panel. Owing to limitations set by TeX, there are certain limits to the sophistication of the procedure. For instance, any ‘whatsits’ (specials (like color changes), file accesses (like `\label`), or hyper anchors) or rules which are inserted directly in the vertical list of the parbox ‘block’ the analysis, so the procedure can’t ‘see’ past them (starting at the bottom of the box) when analysing the contents of the parbox. The user should make sure such items are set in horizontal mode (by using `\leavevmode` or enclosing stuff in boxes). Furthermore, only overfull and underfull hboxes which occur while setting the parbox are considered when judging which width is ‘best’. This will reliably make the width large enough to contain ‘wide’ objects like tabulars, logos and buttons, but might not give optimal results for justified text. vboxes occurring directly in the vbox are ignored. Note further that hboxes with fixed width (made by `\hbox to...`) which occur directly in the vbox may disturb the procedure, because the fixed width cannot be recovered. These hboxes will be reformatted with the width of the vbox, generating an extremely large badness, unsettling the calculation of maximum badness. To avoid this such hboxes should be either contained in a vbox or set in horizontal mode with appropriate glue at the end.

If the optional argument `[name]` is given, the panel contents and (calculated) size will also be stored under the given name, to be restored later with `\restorepanels`. This is nice for switching between different sets of panels.

1824 `\newcommand{\DeclarePanel}{\ifstar\auto@declarepanel@TP\declarepanel@TP}`

```

1825
1826 \newcommand{\declarepanel@TP}[3][ ]% Non-starred version of \DeclarePan
1827 {%
1828 \expandafter\gdef\csname #2panelcontents@TP\endcsname{#3}% Just store panel contents.
1829 \ifthenelse{equal{#1}{}}{\csname store#2panel@TP\endcsname{#1}}% If <name> was given, stor
1830 }%
1831
1832 % Store away top panel parameters.
1833 \newcommand{\storetoppanel@TP}[1]
1834 {%
1835 \expandafter\global\expandafter\let\csname toppanelcontents@TP@#1\endcsname\toppanelcontents@
1836 \expandafter\global\expandafter\let\csname toppanelwidth@TP@#1\endcsname\toppanelwidth@TP%
1837 \expandafter\global\expandafter\let\csname toppanelheight@TP@#1\endcsname\toppanelheight@TP%
1838 \expandafter\global\expandafter\let\csname toppanelshift@TP@#1\endcsname\toppanelshift@TP%
1839 }
1840
1841 % Store away bottom panel parameters.
1842 \newcommand{\storebottompanel@TP}[1]
1843 {%
1844 \expandafter\global\expandafter\let\csname bottompanelcontents@TP@#1\endcsname\bottompanelcon
1845 \expandafter\global\expandafter\let\csname bottompanelwidth@TP@#1\endcsname\bottompanelwidth@
1846 \expandafter\global\expandafter\let\csname bottompanelheight@TP@#1\endcsname\bottompanelheigh
1847 \expandafter\global\expandafter\let\csname bottompanelshift@TP@#1\endcsname\bottompanelshift@
1848 }
1849
1850 % Store away left panel parameters.
1851 \newcommand{\storeleftpanel@TP}[1]
1852 {%
1853 \expandafter\global\expandafter\let\csname leftpanelcontents@TP@#1\endcsname\leftpanelcontent
1854 \expandafter\global\expandafter\let\csname leftpanelwidth@TP@#1\endcsname\leftpanelwidth@TP%
1855 \expandafter\global\expandafter\let\csname leftpanelheight@TP@#1\endcsname\leftpanelheight@TP%
1856 \expandafter\global\expandafter\let\csname leftpanelraise@TP@#1\endcsname\leftpanelraise@TP%
1857 }
1858
1859 % Store away right panel parameters.
1860 \newcommand{\storerightpanel@TP}[1]
1861 {%
1862 \expandafter\global\expandafter\let\csname rightpanelcontents@TP@#1\endcsname\rightpanelcon
1863 \expandafter\global\expandafter\let\csname rightpanelwidth@TP@#1\endcsname\rightpanelwidth@TP
1864 \expandafter\global\expandafter\let\csname rightpanelheight@TP@#1\endcsname\rightpanelheight@
1865 \expandafter\global\expandafter\let\csname rightpanelraise@TP@#1\endcsname\rightpanelraise@TP
1866 }
1867
1868
1869 \newcommand{\auto@declarepanel@TP}[2]% Starred version of \DeclarePan
1870 {\csname calc#1dimen@TP\endcsname{#2}\declarepanel@TP{#1}{#2}} % Calculate 'optimal' dimension
1871
1872 % Restore panel parameters stored away under a given name.
1873 \newcommand{\restorepanels}[1]
1874 {%

```

```

1875 \@ifundefined{toppanelcontents@TP@#1}{\global\let\toppanelcontents@TP\empty}
1876 {%
1877   \expandafter\global\expandafter\let\expandafter\toppanelcontents@TP\csname toppanelcontents
1878   \expandafter\global\expandafter\let\expandafter\toppanelwidth@TP\csname toppanelwidth@TP@#1
1879   \expandafter\global\expandafter\let\expandafter\toppanelheight@TP\csname toppanelheight@TP@
1880   \expandafter\global\expandafter\let\expandafter\toppanelshift@TP\csname toppanelshift@TP@#1
1881   }%
1882 \@ifundefined{bottompanelcontents@TP@#1}{\global\let\bottompanelcontents@TP\empty}
1883 {%
1884   \expandafter\global\expandafter\let\expandafter\bottompanelcontents@TP\csname bottompanelco
1885   \expandafter\global\expandafter\let\expandafter\bottompanelwidth@TP\csname bottompanelwidth
1886   \expandafter\global\expandafter\let\expandafter\bottompanelheight@TP\csname bottompanelheig
1887   \expandafter\global\expandafter\let\expandafter\bottompanelshift@TP\csname bottompanelshift
1888   }%
1889 \@ifundefined{leftpanelcontents@TP@#1}{\global\let\leftpanelcontents@TP\empty}
1890 {%
1891   \expandafter\global\expandafter\let\expandafter\leftpanelcontents@TP\csname leftpanelconten
1892   \expandafter\global\expandafter\let\expandafter\leftpanelwidth@TP\csname leftpanelwidth@TP@
1893   \expandafter\global\expandafter\let\expandafter\leftpanelheight@TP\csname leftpanelheight@T
1894   \expandafter\global\expandafter\let\expandafter\leftpanelraise@TP\csname leftpanelraise@TP@
1895   }%
1896 \@ifundefined{rightpanelcontents@TP@#1}{\global\let\rightpanelcontents@TP\empty}
1897 {%
1898   \expandafter\global\expandafter\let\expandafter\rightpanelcontents@TP\csname rightpanelcont
1899   \expandafter\global\expandafter\let\expandafter\rightpanelwidth@TP\csname rightpanelwidth@T
1900   \expandafter\global\expandafter\let\expandafter\rightpanelheight@TP\csname rightpanelheight
1901   \expandafter\global\expandafter\let\expandafter\rightpanelraise@TP\csname rightpanelraise@T
1902   }%
1903 }

```

### 3.5.2 Implementation of automatic dimension calculation

Interface to the horizontal and vertical calculation procedures. The first argument is being recalculated, the second and third ones are parameters.

```

1904 \newcommand{\calctopdimen@TP}[1]{\calcvdimen@TP{\toppanelheight}{\toppanelwidth}{#1}}
1905 \newcommand{\calcbottomdimen@TP}[1]{\calcvdimen@TP{\bottompanelheight}{\bottompanelwidth}{#1}}
1906 \newcommand{\calcleftdimen@TP}[1]{\calchdimen@TP{\leftpanelwidth}{\leftpanelheight}{#1}}%
1907 \newcommand{\calcrightdimen@TP}[1]{\calchdimen@TP{\rightpanelwidth}{\rightpanelheight}{#1}}%

```

Remove any contents which could mess up the box analysis.

```

1908 \newcommand{\panel@sanitize@TP}
1909 {%
1910   \let\hyperlink=\@secondoftwo
1911   \let\Acrobatmenu=\@secondoftwo
1912 }

```

Calculate height of 'horizontal' panel.

```

1913 \newcommand{\calcvdimen@TP}[3]
1914 {%
1915   \setbox\tempbox@TP=\hbox{\panel@sanitize@TP\@mk@panel@TP{#2}{\toppaneltextcolor}{#3}}% Set

```

```

1916 \mklength@TP{#1}{\ht\tempbox@TP+\dp\tempbox@TP}% % Meas
1917 }
    Calculate 'optimal' width of 'vertical panel.
1918 \newcommand{\calchdimen@TP}[3]
1919 {%
1920 \optwidth@TP{#1}{#2-\panelmargin*2}% Calculate 'optimal' width of a parbox. Pane
1921 {.5\textwidth-\panelmargin*2}{\panelalignment#3}% .5\textwidth is the hardcoded absolute maxi
1922 \mklength@TP{#1}{#1+\panelmargin*2}%
1923 }
    User-configurable: Which 'resolution' should be used when searching for 'best'
    width?
1924 \newcommand{\optwidthsteps}{100}
    User-configurable: Which badness should be tolerated as 'perfect' (stopping the
    search for a better one).
1925 \newcommand{\optwidthlinetolerance}{200}
    Internal parameter: Badness of the parbox currently under consideration.
1926 \let\maxbadness@TP=\@tempcnta
    A hook to disable some commands which would be in the way while measuring
    things.
1927 \def\optwidthdisablecommands@TP
1928 {%
1929 \let\Hy@colorlink\@firstofone
1930 \let\Hy@endcolorlink\relax
1931 }
    Calculate 'best' width of a parbox. The current algorithm will set the textual
    contents into parboxes of increasing width, starting from Opt and ending with the
    maximum width given, in \optwidthsteps steps. The 'badness' of every parbox
    is measured. If it is below the threshold defined by \optwidthlinetolerance,
    the process is stopped and the found width accepted. If this doesn't happen, the
    width of the parbox with the least badness is returned.
1932 \newcommand{\optwidth@TP}[4]
1933 {%
1934 \setcounter{tmpcnta@TP}{0}% Initialize 'probe counter' for b
1935 \let\best@cnt@TP=\empty% Initialize number of best 'probe
1936 \def\bestbadness@TP{1000000}% Initialize badness of best 'probe
1937 \setboolean{carryon@TP}{true}% Flag for breaking out of loop.
1938 \setlength{\tempdimb@TP}{#2}% Store maximal box height.
1939 \whiledo
1940 {\value{tmpcnta@TP}<\optwidthsteps\and\boolean{carryon@TP}}% Probes done or break of loop?
1941 {%
1942 \stepcounter{tmpcnta@TP}% Start next probe.
1943 \setbox\@tempboxa=\hbox% The trick with vbox/lastbox is t
1944 {% produced by \parbox 'immediately
1945 \optwidthdisablecommands@TP% Turn off some nasties not needed
1946 \parbox[b]{(##3)/\optwidthsteps*\value{tmpcnta@TP}}% Make the next parbox.

```

```

1947     {\hfuzz\maxdimen\hbadness\@M\relax#4}%
1948     \global\setbox\tempbox@TP=\lastbox%           ... and assign \tempbox@TP to it
1949     }%
1950 \setlength{\tempdima@TP}{\ht\tempbox@TP+\dp\tempbox@TP}% Measure total height.
1951 \ifthenelse{\lengthtest{\tempdima@TP>\tempdimb@TP}}% If it exceeds the maximum height
1952 {}%                                               acceptable anyway.
1953 {%
1954     \calcmxbadness@TP{\maxbadness@TP}{\tempbox@TP}% Calculate 'worst badness' of any
1955     \ifthenelse{\not\maxbadness@TP>\optwidthlinetolerance}% Below Threshold?
1956     {% Yes. Accept this width.
1957         \edef\best@cnt@TP{\thetmpcnta@TP}%           Store this probe number.
1958         \setboolean{carryon@TP}{false}%           Break loop.
1959     }
1960     {% No. Carry on.
1961         \ifthenelse{\maxbadness@TP<\bestbadness@TP}%           Below lowest badness found so far
1962         {% Yes. Store probe number.
1963             \edef\bestbadness@TP{\number\maxbadness@TP}%           Store badness value.
1964             \edef\best@cnt@TP{\thetmpcnta@TP}%           Store probe number.
1965         }
1966         }% No. Try next probe.
1967     }% matches second argument of \ifthenelse{\not\maxbadness@TP>\optwidthlinetolerance}%
1968 }% matches second argument of \ifthenelse{\lengthtest{\tempdima@TP>\tempdimb@TP}}%
1969 }% matches \whiledo{\value{tmpcnta@TP}<\optwidthsteps\and\boolean{carryon@TP}}{%
1970 \ifx\best@cnt@TP\empty%                               Was _any_ badness below the init
1971     \mklength@TP{#1}{#3}% No; return max width.
1972     \else
1973     \mklength@TP{#1}{(#3)/\optwidthsteps*\best@cnt@TP}% Yes; return width of best probe.
1974 \fi
1975 }% matches \newcommand{\optwidth@TP}[4]{%
    Calculate maximal badness of any hbox occurring in a vbox.
1976 \newcommand{\calcmxbadness@TP}[2]
1977 {%
1978     \let\@resultcnt@TP=#1%           Here we store the result.
1979     \global\@resultcnt@TP=\z@\relax% Just in case no hbox occurs...
1980     \setlength{\@tempdima}{\wd#2}%   This is the width to which every hbox is stretched for finding
1981     \setbox\@tempboxa=\vbox%         A dummy vbox for recursively analysing the vbox contents using
1982     {%
1983         \hfuzz\maxdimen\hbadness\@M
1984         \unvbox#2%                   'free' the contents of the vbox.
1985         \measureboxes@TP%           Analyse 'tail to head' using \lastbox.
1986     }%
1987 }
    Recursively analyse vertical list using \lastbox, to find maximum badness of
    any contained hbox.
1988 \newcommand{\measureboxes@TP}%
1989 {%
1990     \unskip\unpenalty\unkern%       This is a kluge for TeX, because there is no certain way of finding
1991     \unskip\unpenalty\unkern%       penalty, glue or kern on the vertical list. \lastpenalty will give
1992     \unskip\unpenalty\unkern%       value of 0 might mean there was none or there was one of value 0

```

```

1993 \unskip\unpenalty\unkern% This is different in eTeX. I might make a switch to a smarter so
1994 \unskip\unpenalty\unkern
1995 \unskip\unpenalty\unkern
1996 \unskip\unpenalty\unkern
1997 \unskip\unpenalty\unkern
1998 \unskip\unpenalty\unkern
1999 \unskip\unpenalty\unkern
2000 \setbox\@tempboxa=\lastbox% Grab last box.
2001 \ifhbox\@tempboxa% Was this an hbox?
2002 \setbox0=\hb@xt@\@tempdima{\unhbox\@tempboxa}% Yes. Reformat with given width.
2003 \ifnum\badness>\@resultcnt@TP% Badness larger than largest recorded badness
2004 \global\@resultcnt@TP=\badness% Yes. Memorize.
2005 \fi
2006 \expandafter\measureboxes@TP% Recursive call.
2007 \else
2008 \ifvbox\@tempboxa% Was this a vbox?
2009 \expandafter\expandafter\expandafter\measureboxes@TP% Ignore, but execute recursive call.
2010 \fi
2011 \fi
2012 }

```

### 3.5.3 Actually typeset panels

```

2013 \newcommand{\mk@toppanel@TP}% top panel
2014 {%
2015 \ifx\toppanelcontents@TP\empty% top panel specified?
2016 \else% Yes; create box with appropriate dimensions, backgr
2017 \@mk@panel@TP{\toppanelwidth}{\toppanelheight}{\toppaneltextcolor}{\toppanelcontents@TP}%
2018 \fi
2019 }
2020
2021 \newcommand{\mk@bottompanel@TP}% bottom panel
2022 {%
2023 \ifx\bottompanelcontents@TP\empty% bottom panel specified?
2024 \else% Yes; create box with appropriate dimensions, backgr
2025 \@mk@panel@TP{\bottompanelwidth}{\bottompanelheight}{\bottompaneltextcolor}{\bottompanelcont
2026 \fi
2027 }
2028
2029 \newcommand{\mk@leftpanel@TP}% left panel
2030 {%
2031 \ifx\leftpanelcontents@TP\empty% left panel specified?
2032 \else% Yes; create box with appropriate dimensions, backgr
2033 \@mk@panel@TP{\leftpanelwidth}{\leftpanelheight}{\leftpaneltextcolor}{\leftpanelcontents@TP}%
2034 \fi
2035 }
2036
2037 \newcommand{\mk@rightpanel@TP}% right panel
2038 {%
2039 \ifx\rightpanelcontents@TP\empty% right panel specified?

```

```

2040 \else%                               Yes; create box with appropriate dimensions, text c
2041 \mk@panel@TP{\rightpanelwidth}{\rightpanelheight}{rightpaneltextcolor}{\rightpanelcontents
2042 \fi
2043 }
2044
2045
2046 \newcommand{\mk@panel@TP}[2]% Generate 'standard' parbox parameters for panels.
2047 {%
2048 \mk@panel@TP{#1}{[#2-\panelmargin*2][s]}%
2049 }
2050
2051
2052 \ifclassloaded{powersem}
2053 {%
2054 \newcommand{\panelalignment}{\sem@ptsize{\slide@ptsize}\large\normalsize}%
2055 }
2056 {%
2057 \newcommand{\panelalignment}%           Justification for panels. This setting allows a cert
2058 {\setlength{\rightskip}{0pt plus 20pt}}% 'right-raggedness'. Leave empty for standard parbox
2059 }
2060
2061 % Make a panel box.
2062 \newcommand{\mk@panel@TP}[4]
2063 {%
2064 \vbox
2065 {%
2066 \offinterlineskip
2067 \kern\panelmargin% Top margin.
2068 \hbox
2069 {{{%
2070 \ifthenelse{\boolean{instepwise@TP}}% Inside \stepwise, colors mat have been dimmed, l
2071 {\usecolorset{stwcolors}}{}% Restore them, just in case.
2072 \color{#3}% Set panel text color.
2073 \kern\panelmargin% Left margin.
2074 \parbox[b]{#2{#1-\panelmargin*2}}% The parbox with the main panel contents.
2075 {%
2076 \normalfont
2077 \panelalignment#4%
2078 \hrule\@height\z@% The hrule makes sure the total height of this box can b
2079 }%
2080 \kern\panelmargin% Right margin.
2081 }}%
2082 \kern\panelmargin% Bottom margin.
2083 }%
2084 }

```

### 3.6 Navigation helpers

The following code is rather preliminary and provides only the very basics for making navigation buttons and such. If you're using a package which allows to do

this, don't even consider using the following.

Some configurable button parameters.

Space between button label and border.

2085 `\newcommand{\buttonsep}{\fboxsep}`

Width of button frame.

2086 `\newcommand{\buttonrule}{0pt}`

Horizontal displacement of button shadow.

2087 `\newcommand{\buttonshadowshift}{.3\fboxsep}`

Vertical displacement of button shadow.

2088 `\newcommand{\buttonshadowvshift}{-.3\fboxsep}`

Button-specific user level commands.

`\button` `\button[<width>][<height>][<depth>][<alignment>]{<navcommand>}{<text>}`

creates a button labelled `<text>` which executes `<navcommand>` when pressed.

`<navcommand>` can be for instance `\Acrobatmenu{<command>}` or `\hyperlink{<target>}`

(note that `navcommand` should take one (more) argument specifying the sensitive area which is provided by `\button`). If given, the optional parameters

`<width>`, `<height>`, and `<depth>` give the width, height and depth, respectively, of the framed area comprising the button (excluding the shadow, but including the frame). Default are the 'real' width, height and depth, respectively, of `<text>`,

plus allowance for the frame. If given, the optional parameter `<alignment>` (one of l,c,r) gives the alignment of `<text>` inside the button box (makes sense only if `<width>` is given).

2089 `\newcommand{\button}[1][Opt]%` Collect first optional parameter.

2090 `{%`

2091 `\mklength@TP\bt@width@TP{#1}%` Store optional argument.

2092 `\button@TP`

2093 `}`

2094

2095 `\newcommand{\button@TP}[1][Opt]%` Collect second optional parameter.

2096 `{%`

2097 `\mklength@TP\bt@height@TP{#1}%` Store optional argument.

2098 `\@button@TP`

2099 `}`

2100

2101 `\newcommand{\@button@TP}[1][Opt]%` Collect third optional parameter.

2102 `{%`

2103 `\mklength@TP\bt@depth@TP{#1}%` Store optional argument.

2104 `\@@button@TP`

2105 `}`

2106

2107 `\newcommand{\@@button@TP}[3][c]%` Collect fourth optional and two mandatory parameters and pro

2108 `{%`

2109 `\ifthenelse{\lengthtest{\bt@width@TP=Opt}}% <width> given?`

2110 `{\mklength@TP{\bt@width@TP}{\widthof{#3}}}% No. Calculate width of <text>.`

2111 `{\mklength@TP{\bt@width@TP}{\bt@width@TP-\buttonsep*2-\buttonrule*2}}% Yes. Calculate area le`



```

2112 \ifthenelse{\lengthtest{\bt@height@TP=0pt}}%      <height> given?
2113 {\mklength@TP{\bt@height@TP}{\heightof{#3}}}%      No. Calculate height of <text>.
2114 {\mklength@TP{\bt@height@TP}{\bt@height@TP-\buttonsep-\buttonrule}}% Yes. Calculate area left
2115 \ifthenelse{\lengthtest{\bt@depth@TP=0pt}}%      <depth> given?
2116 {\mklength@TP{\bt@depth@TP}{\depthof{#3}}}%      No. Calculate depth of <text>.
2117 {\mklength@TP{\bt@depth@TP}{\bt@depth@TP-\buttonsep-\buttonrule}}% Yes. Calculate area left f
2118 \leavevmode% \rlap creates a 'raw' hbox. So we get into horizontal mode.
2119 \rlap%      Make shadow.
2120 {%
2121   \hspace*{\buttonshadowshift}% Horizontal displacement.
2122   \raisebox{\buttonshadowvshift}% Vertical displacement.
2123   {%
2124     {% Inner group for correct color handling.
2125       \setcolor@TP{buttonshadowcolor}% Button shadow color.
2126       \rule%      Create colored rectangular patch of appropriate dimens
2127       [-\bt@depth@TP-\buttonsep-\buttonrule]
2128       {\bt@width@TP+\buttonsep*2+\buttonrule*2}
2129       {\bt@height@TP+\bt@depth@TP+\buttonsep*2+\buttonrule*2}%
2130       }% matches inner group
2131     }% matches \raisebox{\buttonshadowshift}{%
2132   }% matches \rlap{%
2133   \edef\o@fboxrule@TP{\the\fboxrule}%      Preserve original definitions of \fbox parameters.
2134   \edef\o@fboxsep@TP{\the\fboxsep}%
2135   #2%      Execute <navcommand>.
2136   {%
2137     \setlength{\fboxrule}{\buttonrule}%      Set \fbox parameters for button frame.
2138     \setlength{\fboxsep}{\buttonsep}%
2139     \fcolorbox{buttonframecolor}{buttoncolor}% Create button frame with the right colors.
2140     {%
2141       \makebox[\bt@width@TP][#1]%      Create box of correct width to contain <text>.
2142       {%
2143         \raisebox{0pt}[\bt@height@TP][\bt@depth@TP]% Create box of correct height and depth.
2144         {%
2145           \setlength{\fboxrule}{\o@fboxrule@TP}\setlength{\fboxsep}{\o@fboxsep@TP}% Restore fbo
2146           \setcolor@TP{buttontextcolor}#3%      Produce <text>.
2147           }% matches \raisebox{0pt}[\bt@height@TP][\bt@depth@TP]{%
2148           }% matches \makebox[\bt@width@TP][#1]{%
2149           }% matches \fcolorbox{buttonframecolor}{buttoncolor}{%
2150           }% matches argument of <navcommand>.
2151           }% matches \newcommand{\@@@button@TP}[3][c]{%

```

Some predefined buttons.

Size of predefined button symbols.

```
2152 \newcommand{\buttonssymbolsize}{\footnotesize}
```

Define predefined button symbols.

```
2153 \@ifpackageloaded{amssymb}%      AMS symbols available?
```

```
2154 {% Yes. Use 'black' symbols.
```

```
2155 \newcommand{\buttonleftarrowsymbol}{\buttonssymbolsize\boldmath\origmath{\blacktriangleleft}}
```

```
2156 \newcommand{\buttonrightarrowsymbol}{\buttonssymbolsize\boldmath\origmath{\blacktriangleright}}
```

```

2157 \newcommand{\buttonbackarrowsymbol}{\buttonssymbolsize\boldmath\origmath{\vartriangleleft}}%
2158 }
2159 {% No. Use replacements from standard set.
2160 \newcommand{\buttonleftarrowsymbol}{\buttonssymbolsize\boldmath\origmath{\lhd}}% 'left
2161 \newcommand{\buttonrightarrowsymbol}{\buttonssymbolsize\boldmath\origmath{\rhd}}% 'right
2162 \newcommand{\buttonbackarrowsymbol}{\buttonssymbolsize\boldmath\origmath{\leftarrow}}% 'back
2163 }

```

Width of predefined buttons.

```

2164 \newcommand{\stdbuttonwidth}{\widthof{\buttonrightarrowsymbol\buttonrightarrowsymbol\buttonrightarrowsymbol}}
Customizeable default: How to find the number of the current page?
2165 \newcommand{\currentpagevalue}{\value{page}}

```

Predefined button: last subpage of previous page.

```

2166 \newcommand{\backpagebutton}[1][\stdbuttonwidth]
2167 {%
2168 \button[#1]{\setcounter{tmpcnta@TP}{\currentpagevalue-1}\hyperlink{page.\thetmpcnta@TP}}
2169 {\buttonleftarrowsymbol\buttonleftarrowsymbol}%
2170 }

```

Predefined button: previous step.

```

2171 \newcommand{\backstepbutton}[1][\stdbuttonwidth]
2172 {%
2173 \button[#1]{\Acrobatmenu{PrevPage}}
2174 {\buttonleftarrowsymbol}%
2175 }

```

Predefined button: ‘undo action’ (go back to whatever was before last action).

```

2176 \newcommand{\gobackbutton}[1][\stdbuttonwidth]
2177 {%
2178 \button[#1]{\Acrobatmenu{GoBack}}
2179 {\buttonbackarrowsymbol}%
2180 }

```

Predefined button: next step.

```

2181 \newcommand{\nextstepbutton}[1][\stdbuttonwidth]
2182 {%
2183 \button[#1]{\Acrobatmenu{NextPage}}
2184 {\buttonrightarrowsymbol}%
2185 }

```

Predefined button: first subpage of next page.

```

2186 \newcommand{\nextpagebutton}[1][\stdbuttonwidth]
2187 {%
2188 \button[#1]{\setcounter{tmpcnta@TP}{\currentpagevalue+1}\hyperlink{firstpage.\thetmpcnta@TP}}
2189 {\buttonrightarrowsymbol\buttonrightarrowsymbol}%
2190 }

```

Predefined button: last subpage of next page.

```

2191 \newcommand{\nextfullpagebutton}[1][\stdbuttonwidth]
2192 {%
2193 \button[#1]{\setcounter{tmpcnta@TP}{\currentpagevalue+1}\hyperlink{page.\thetmpcnta@TP}}

```

```

2194 {\buttonrightarrowsymbol\buttonrightarrowsymbol\buttonrightarrowsymbol}%
2195 }
    Predefined button: toggle fullscreen mode.
2196 \newcommand{\fullscreenbutton}[1][\stdbuttonwidth]
2197 {%
2198   \button[#1]{\Acrobatmenu{FullScreen}}
2199   {\buttonrightarrowsymbol\buttonleftarrowsymbol}%
2200 }

```

### 3.7 Set acrobat reader's page transition mode

Most of the following is snarfed from an email message of Marc van Dongen to the ppower4 mailing list on Thu, 7 Oct 1999. Thanks to Marc for his permission to include his code into this package.

```

\pagetransition \pagetransition{<setting>} is a generic command for setting the page tran-
sition with hyperref's \hypersetup. The command is defined to a noop in case
hyperref is not loaded.
2201 \ifthenelse{\boolean{display}}%           Are dynamic features enabled?
2202 {% Yes.
2203   \newcommand{\pagetransition}[1]%         Definition for preamble.
2204   {%
2205     \@ifpackageloaded{hyperref}%         Can we use \hypersetup?
2206     {% Yes.
2207       \hypersetup{pdfpagetransition={#1}}% Set page transition with \hypersetup.
2208     }
2209     {% No. In this case, we can't set the page transition.
2210       \PackageWarning{texpower}
2211       {Package hyperref not loaded.\MessageBreak Page transition not set}%
2212     }%
2213     }% matches \newcommand{\pagetransition}[1]{%
2214   \AtBeginDocument%                       Definition for document body.
2215   {%
2216     \@ifpackageloaded{hyperref}%         Can we use \hypersetup?
2217     {% Yes.
2218       \hypersetup{pdfpagetransition={}}% Make pagetransition setting (consistently) local to gr
2219       \renewcommand{\pagetransition}[1]{\hypersetup{pdfpagetransition={#1}}}%
2220     }
2221     {% No. Disable page transitions.
2222       \PackageWarning{texpower}
2223       {Package hyperref not loaded.\MessageBreak Page transitions disabled}%
2224       \let\pagetransition=\@gobble
2225     }%
2226     }% matches \AtBeginDocument{%
2227   }% matches \ifthenelse{\boolean{display}}{%
2228   {\let\pagetransition=\@gobble}% No. Disable page transitions.

```

Some standard page transitions.

```

2229 \newcommand{\pageTransitionSplitH0}{\pagetransition{Split /Dm /H /M /O}}% Split Horizontally to

```

2230  
2231 `\newcommand{\pageTransitionSplitHI}{\pagetransition{Split /Dm /H /M /I}}`% Split Horizontally to  
2232  
2233 `\newcommand{\pageTransitionSplitVO}{\pagetransition{Split /Dm /V /M /O}}`% Split Vertically to t  
2234  
2235 `\newcommand{\pageTransitionSplitVI}{\pagetransition{Split /Dm /V /M /I}}`% Split Vertically to t  
2236  
2237 `\newcommand{\pageTransitionBlindsH}{\pagetransition{Blinds /Dm /H}}`% Horizontal Blinds.  
2238  
2239 `\newcommand{\pageTransitionBlindsV}{\pagetransition{Blinds /Dm /V}}`% Vertical Blinds.  
2240  
2241 `\newcommand{\pageTransitionBoxO}{\pagetransition{Box /M /O}}`% Growing Box.  
2242  
2243 `\newcommand{\pageTransitionBoxI}{\pagetransition{Box /M /I}}`% Shrinking Box.  
2244  
2245 % argument must be number fom 0 to 360  
2246 `\newcommand{\pageTransitionWipe}[1]{\pagetransition{Wipe /Di #1}}`% Wipe from one edge of the pa  
2247  
2248 `\newcommand{\pageTransitionDissolve}{\pagetransition{Dissolve}}`% Dissolve.  
2249  
2250 % argument must be number fom 0 to 360  
2251 `\newcommand{\pageTransitionGlitter}[1]{\pagetransition{Glitter /Di #1}}`% Glitter from one edge  
2252  
2253 `\newcommand{\pageTransitionReplace}{\pagetransition{Replace}}`% Simple Replace (the default).

### 3.8 Set acrobat reader's automatic page advancing feature

`\pageDuration{<d>}` will force pages to be advanced automatically after <d> seconds when in full screen mode. The effect starts on the current page and is undone by a group end or the command `\stopAdvancing`. <d> should be a (fixed-point) number.

Note a very strange behaviour of acrobat and acroread v4.05 and onwards: Automatic advancing is disabled unless explicitly enabled by the “advance every n seconds” setting in the full screen dialogue. But in this case, all pages not having any page duration setting will be advanced after <n> seconds. As another trap, at least in some versions of acroread and acrobat, the maximum for <n> when set in the dialogue is 60 seconds. To make matters even worse, there seems to be a (quite arbitrary) even for duration setting of pages: 546 seconds (about nine minutes). This leaves you with the following options if you want to use automated advancing in your presentations (for animation effects, say):

1. You're using a version of acrobat or acroread predating 4.05 (or any other version where this misbehaviour miraculously isn't present): All is well, nothing to do.
2. You're using a 'buggy' version where more than 60 seconds can be set as a value for <n> in the full screen dialogue (I'm not sure whether such a beast exists; maybe both 'features' coincide): Just activate automatic advancing

in the full screen dialogue and set an ‘infinite’ number of seconds (3600, say). This value is masked by an explicit setting using `\pageDuration`, so animations will still work.

3. You’re using a ‘buggy’ version with a maximum dialogue setting of 60, but you’re comfortable with your presentation always advancing automatically at least every 60 seconds: Set `<n>` to 60 and proceed as above.
4. You’re using a ‘buggy’ version with a maximum dialogue setting of 60 and you’re comfortable with your presentation always advancing automatically at least every 546 seconds (9.1 minutes): Give the package option “ninemminutes” to `texpower`. This will put an explicit duration setting of 546 seconds on every page (overridden by explicit settings using `\pageDuration`). Set the advancing time in the full screen dialogue to anything; it will be ignored because every page has an explicit setting.
5. You’re using a ‘buggy’ version with a maximum dialogue setting of 60, you’d like your presentation to advance more slowly than every 546 seconds and your version of `acroread` or `acrobat` miraculously doesn’t have the upper bound of 546 seconds (which I’ve empirically discovered with `acrobat 5.0` on Win NT): Put `\renewcommand{\infinitepageduration}{3600}` in your preamble and see what happens. If the value is too high, `acroread` will go into fast forward mode or do other strange things. Otherwise: Lucky you.
6. You’re using a ‘buggy’ version with a maximum dialogue setting of 60, you’d like your presentation to advance more slowly than every 546 seconds and your version of `acroread` or `acrobat` also has the upper bound of 546 seconds: In this case you’re out of luck. Of course you can try to get the guys at adobe to clean up this whole mess. Good luck with that!

If `\infinitepageduration` is set to empty, then a page duration setting will appear only where `\pageDuration` was used. Otherwise, every page without explicit setting gets a page duration of `\infinitepageduration`.

```
2254 \newcommand{\infinitepageduration}{}%           Default: No page duration setting on every page.
2255
2256 \ifthenelse{\boolean{nineminutes@TP}}%           If the option nineminutes is given, every page w
2257 {\renewcommand{\infinitepageduration}{546}}{}% gets a setting of 546 seconds (9.1 minutes).
```

Implementation of `\pageDuration`.

```
2258 \ifthenelse{\boolean{display}}%                 Are dynamic features enabled?
2259 {% Yes.
2260 \newcommand{\pageDuration}[1]%                 Definition for preamble.
2261 {%
2262 \ifpackageloaded{hyperref}%                     Can we use \hypersetup?
2263 {% Yes.
2264 \hypersetup{pdfpageduration={#1}}%             Set page duration with \hypersetup.
2265 }%
2266 {% No. In this case, we can't set the page duration.
2267 \PackageWarning{texpower}
```

```

2268     {Package hyperref not loaded.\MessageBreak Page duration not set}%
2269     }%
2270   }% matches \newcommand{\pageDuration}[1]{%
2271 \AtBeginDocument%           Make sure this also works if texpower is loaded before
2272 {%
2273   \@ifpackageloaded{hyperref}%           Can we use \hypersetup?
2274   {% Yes.
2275     \edef\next{\noexpand\hypersetup{pdfpageduration={\infinitepageduration}}}%
2276     \next%           Set default page duration.
2277     \renewcommand{\pageDuration}[1]{\hypersetup{pdfpageduration={#1}}}% Set page duration with
2278     }%
2279   {% No. In this case, we can't set the page duration.
2280     \PackageWarning{texpower}
2281     {Package hyperref not loaded.\MessageBreak Page duration disabled}%
2282     \let\pageDuration=\@gobble
2283     }%
2284   }% matches \AtBeginDocument{%
2285   }% matches \ifthenelse{\boolean{display}}{%
2286 {\let\pageDuration=\@gobble}% No. Disable page duration.
2287   \stopAdvancing undoes any setting effected by \pageDuration.
2287 \newcommand{\stopAdvancing}
2288 {\edef\next{\noexpand\pageDuration{\infinitepageduration}}\next}% Reset to default page duration

```

### 3.9 TeXPower kernel

This area contains the ‘low level’ implementation of TeXPowers central functions. Mainly, (La)TeX’s `\shipout` and `\output` routines are overloaded, adding some necessary functionality (duplication of page contents for incremental page building; display of backgrounds and panels). Also, the user command `\pause` is implemented here, using the kernel functions for saving and restoring page contents.

The code for overloading the output routine is derived from Klaus Guntermanns `texpause` package which can be obtained from the PPower4 Web site

<http://www-sp.iti.informatik.tu-darmstadt.de/software/ppower4/>

Thanks to Heiko Oberdiek for his suggestion how `\leaders` can be used to disable processing of whatsits in the duplicated text.

The code for overloading `\shipout` is derived from `everyshi.sty` with permission by Martin Schroeder.

#### 3.9.1 Overload `\shipout`

Overloading is done at the beginning of the document just in case some other package messes with `\shipout`.

```

2289 \AtBeginDocument{\shipoutinit@TP}%
2290   Replace \shipout by own definition.
2290 \newcommand*{\shipoutinit@TP}
2291 {%
2292   \let\o@shipout@TP=\shipout% Memorize previous definition.

```

```

2293 \let\shipout\shipout@TP%    Replace by own one.
2294 }%

```

Our own version of `\shipout` takes the offered box and passes it to another routine for further inspection.

```

2295 \newcommand{\shipout@TP}
2296 {%
2297   \afterassignment\shipout@test@TP
2298   \global\setbox\@cclv= %
2299 }%

```

If `\shipout` is called with an argument of the form `\box<n>`, then the box will have indeed been assigned to `\@cclv` at the time `\shipout@test@TP` is called. In this case, `\shipout@output@TP` is called immediately. If `\shipout` is called with an argument of the form `\vbox{...}`, then `\shipout@test@TP` is called at the beginning of the box definition, **before** the box is assigned (this is a feature of `\afterassignment`). In this case, the call of `\shipout@output@TP` is moved **after** the box definition using `\afetrgroup`, so that `\@cclv` is correctly defined at the time `\shipout@output@TP` is executed.

```

2300 \newcommand{\shipout@test@TP}
2301 {%
2302   \ifvoid\@cclv\relax%           Is the box assigned yet?
2303     \aftergroup\shipout@output@TP% No; defer execution of \shipout@output@TP.
2304   \else
2305     \shipout@output@TP%         Yes; execute \shipout@output@TP immediately.
2306   \fi%
2307 }%

```

The main part of our redefinition of `\shipout`.

```

2308 \newcommand{\shipout@output@TP}
2309 {%
2310   \shipout@hook@TP%             Here, our own stuff is executed, manipulating \@cclv.
2311   \o@shipout@TP\box\@cclv%    Execute original shipout routine.
2312 }%

```

### 3.9.2 The kernel functions to be executed at `\shipout`

`\AtShipout{<code>}` will save `<code>` in a special place where it is executed at the time of the next `\shipout` (and then deleted). Needless to say it should not produce any output. This is mainly for placing anchors in a controlled way even inside incremental builds.

```

2313 \newcommand{\AtShipout}[1]
2314 {\expandafter\gdef\expandafter\at@shipout@once@TP\expandafter{\at@shipout@once@TP#1}}% Add argu

```

The container for `<code>` stored away by `\AtShipout`. This is placed on the page by the next `\shipout` and then deleted.

```

2315 \newcommand{\at@shipout@once@TP}{}

```

Save the original definition of `\hyper@@anchor`.

```

2316 \AtBeginDocument{\global\let\o@hyper@@anchor\hyper@@anchor}

```

The following makes sure that a hyper target `firstpage.<n>` is placed on the **first** subpage of every page.

```

2317 \ifpackageloaded{hyperref}%                               Is hyperref loaded?
2318 {% Yes. Prepare hook.
2319 \newcommand{\do@insert@firstanchor@TP}%                   This is executed on the first subpage o
2320 {\hypertarget{firstpage.\number\currentpagevalue}{}}% Create target.
2321 }
2322 {\newcommand{\do@insert@firstanchor@TP}{}}% No. Leave hook empty.
    This is executed on every subpage which is not the first one.
2323 \newcommand{\dont@insert@firstanchor@TP}{}%
    This is the hook itself which is executed on every page.
2324 \newcommand{\insert@firstanchor@TP}{\do@insert@firstanchor@TP}%
    The hook executed at every call of \shipout. It executes the following tasks:
    1. Filter out whatsits on duplicate pages.
    2. Put page background at the ‘lowest’ layer.
    3. Put panels at the ‘second lowest’ layer.
    4. Execute \AtShipout code and place hypertarget firstpage.<n>.
    5. Put \box\@cclv at ‘top’ layer.
2325 \newcommand{\shipout@hook@TP}
2326 {%
2327 \filterpage@TP%                               Filter out whatsits on duplicate pages.
2328 \setbox\@cclv=%                               Create ‘real’ page box (which is later shipped out).
2329 \hbox{%
2330 \set@typeset@protect
2331 \raise\ht\@cclv\rlap%                           Place background box.
2332 {%
2333 \vtop to \TPpageheight
2334 {%
2335 \offinterlineskip
2336 \hrule\@height\z@\relax
2337 \kern -1truein\relax
2338 \kern -\voffset\relax
2339 \moveleft 1truein\hbox{\kern-\hoffset\copy\bgndbox@TP}%
2340 \vss
2341 \kern 1truein\relax
2342 \kern\voffset\relax
2343 }%
2344 }%
2345 \raise\ht\@cclv\rlap%                           Place ‘vertical’ panels.
2346 {%
2347 \vtop to \TPpageheight
2348 {%
2349 \offinterlineskip

```



```

2350     \hrule\@height\z@\relax
2351     \kern -1truein\relax
2352     \kern -\voffset\relax
2353     \moveleft 1truein\hb@xt@\TPpagewidth
2354     {%
2355         \kern-\hoffset\raise\leftpanelraise\hbox{\mk@leftpanel@TP}%
2356         \hfill
2357         \raise\rightpanelraise\hbox{\mk@rightpanel@TP}%
2358         \kern\hoffset
2359     }%
2360     \vss
2361     \kern 1truein\relax
2362     \kern\voffset\relax
2363     }%
2364 }%
2365 \raise\ht\@cclv\rlap%                Place ‘horizontal’ panels.
2366 {%
2367     \vtop to \TPpageheight
2368     {%
2369         \offinterlineskip
2370         \hrule\@height\z@\relax
2371         \kern -1truein\relax
2372         \kern -\voffset\relax
2373         \moveleft 1truein\hbox{\kern-\hoffset\kern\toppanelshift\mk@toppanel@TP}%
2374         \vfill
2375         \moveleft 1truein\hbox{\kern-\hoffset\kern\bottompanelshift\mk@bottompanel@TP}%
2376         \kern 1truein\relax
2377         \kern\voffset\relax
2378     }%
2379 }%
2380 \raise\ht\@cclv\rlap
2381 {%
2382     \let\hyper@@anchor\o@hyper@@anchor%    Reactivate hyper anchors.
2383     \insert@firstanchor@TP%                Execute hook for creating target firstpage.<n>
2384     \ifshippingduplicate%                  Will further subpages follow?
2385     \global\let\insert@firstanchor@TP=\dont@insert@firstanchor@TP% Deactivate hook for next
2386     \else
2387     \global\let\insert@firstanchor@TP=\do@insert@firstanchor@TP%   Reactivate hook for next
2388     \fi
2389     \at@shipout@once@TP%                    Execute code stored away by \AtShipout.
2390     }%
2391     \global\let\at@shipout@once@TP=\empty%   Clear \AtShipout container.
2392     \box\@cclv
2393     }%
2394 }

```

### 3.9.3 Implementation of ‘fixcolorstack’ option

The problem is this: dvips (and some other tools) maintains a color stack when converting dvi to ps. Its function is to always be able to correctly switch back to

the previously used color even if there is a page break (and according typesetting of headers etc) inbetween switching to another color and switching back. This has many advantages (pdftex, for instance, which doesn't maintain a color stack, always has problems to restore the correct text color after a page break). In connection with TeXPower, however, this leads to problems. For controlling the color stack, push and pop commands are inserted into the dvi using specials. At typesetting time, these specials are stored in the vertical list. When page contents are copied, it might be that the copy contains an unequal number of push and pop commands, which will make dvips's color stack go out of sync. To remedy this, texpower maintains a 'counter stack' which should contain all color stack commands issued on the current page. Whenever a copied page is shipped out, appropriate color stack correction commands are issued to balance the stack. As this is necessary only for drivers implementing a color stack, the option is disabled for some drivers.

```

2395 \def\colorcorrections@TP{\relax}%           The 'counter stack' of color correction codes, reset
2396
2397 \AtBeginDocument%                           Make sure to catch color.sty even if it's loaded aft
2398 {%
2399 \ifthenelse{\boolean{fixcolorstack@TP}}% Was the fixcolorstack option given?
2400 {%
2401 \ifundefined{VTeXversion}%                   Using vtex?
2402 {% No. Go on.
2403 \ifthenelse{\boolean{pdf}}%                 Producing pdf with pdftex?
2404 {% Yes. Color stack fixing unnecessary.
2405 \PackageWarning{texpower}
2406 {pdftex doesn't need color stack correction.\MessageBreak Option fixcolorstack disabled
2407 }
2408 {% No. We need to add corrections for the color stack...
2409 \@ifpackageloaded{color}%                   ... but only if color.sty is loaded at all.
2410 {%
2411 \expandafter\def\expandafter\shipout@hook@TP\expandafter% Extend \shipout hook ...
2412 {\shipout@hook@TP\clearcolorcorrections@TP}% ... by a command to clear c
2413 %
2414 \newcommand{\clearcolorcorrections@TP}%     The counter stack is
2415 {%                                         ... on all 'final' s
2416 \ifshippingduplicate
2417 \else
2418 \gdef\colorcorrections@TP{\relax}%
2419 \fi
2420 }

```

Pray to god all drivers will support the following hacks...

Save 'original' definition of `\set@color`. Our redefinition of `\reset@color` needs to know which color is being reset, so we add this as an argument. As it would be inconvenient to add a lot of tokens with `\aftergroup`, we wrap everything in a control sequence.

```

2421 \expandafter\def\expandafter\o@set@color@TP\expandafter%
2422 {%

```

```

2423         \set@color%                               This is the ‘real’ origi
2424         \expandafter\aftergroup\csname\current@color\endcsname%   Add definition of \curre
2425         }%
2426
2427         \let\o@reset@color@TP=\reset@color%         Save original definition

```

We need one command to ‘just push’ a color on the stack. Pushing is done by `\set@color` in a driver-specific way. But `\set@color` also creates an instance of `\reset@color` using `\aftergroup`. This instance is gobbled by this hack, hopefully leaving only the driver-specific code to push a color. Of course, this will break spectacularly if `\set@color` doesn’t have the form `{<do something>\aftergroup\reset@color}`.

```

2428         \def\remove@resetcolor@TP#1\aftergroup\reset@color%
2429         {\def\pushcolor@TP##1{\def\current@color{##1}##1}}%
2430         %
2431         \expandafter\remove@resetcolor@TP\set@color

```

`\reset@color` gets the color definition in the form of a control sequence (because of `\aftergroup`, see above). We have to ‘unwrap’ it before pushing.

```

2432         \def\pushcolorname@TP#1%
2433         {%
2434         \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup% Get the coded tokens
2435         \expandafter\pushcolor@TP\expandafter{\@gtempa}%           ... and execute push
2436         }%

```

Our own definition of `\set@color` adds a `\reset@color` command for the color just set to `\colorcorrections@TP`.

```

2437         \def\set@color%
2438         {%
2439         \o@set@color@TP%                               ‘Original’ definition
2440         \expandafter\@temptokena\expandafter{\colorcorrections@TP}% Add \o@reset@color@TP
2441         \xdef\colorcorrections@TP{\noexpand\o@reset@color@TP\the\@temptokena}% ... of \col
2442         }
2443
2444         \def\reset@color#1%                             \reset@color now recei
2445         {%
2446         \expandafter\addpushtocorr@TP\colorcorrections@TP\@nil{#1}% Add a push command for
2447         % of \colorcorrections@T
2448         \o@reset@color@TP%                               ‘Original’ definition
2449         }%
2450
2451         \def\addpushtocorr@TP#1#2\@nil#3%             Add a push command for some color to the front
2452         {%
2453         \ifx\o@reset@color@TP#1%                       For efficiency, a push immediately followed by
2454         \gdef\colorcorrections@TP{#2}%
2455         \else
2456         \gdef\colorcorrections@TP{\pushcolorname@TP{#3}#1#2}% Otherwise, add the push to
2457         \fi
2458         }%
2459         }% matches \ifpackageloaded{color}

```

```

2460     {}% No changes needed if color.sty is not loaded.
2461     }% matches second argument of \ifthenelse{\boolean{pdf}}
2462     }% matches \@ifundefined{VTeXversion}
2463     {% Yes. Color stack fixing unnecessary.
2464     \PackageWarning{texpower}
2465     {vtex doesn't need color stack correction.\MessageBreak Option fixcolorstack disabled.}%
2466     }%
2467     }% matches \ifthenelse{\boolean{fixcolorstack@TP}}
2468     {}%
2469     }% matches \AtBeginDocument

```

### 3.9.4 Kernel functions for overloading \output

This is the ‘inner kernel’ which lies behind all dynamic effects.

Some user level parameters.

This flag can be evaluated at \output (resp. \shipout) time and tells whether the page being shipped out is a duplicate page.

```
2470 \newif\ifshippingduplicate
```

The command used to shipout a duplicate page.

```
2471 \providecommand{\TProject}{\newpage}
```

Some internal registers to store away things.

Contents of the page so far. These have to be duplicated on every subpage.

```
2472 \newbox\mempageconts@TP
```

Footnotes on the page being duplicated. These have to be duplicated also.

```
2473 \newinsert\memfootins@TP
```

Original definition of output routine.

```
2474 \newtoks\memoutput@TP
```

Save current page contents to a box.

Which counters are to be restored to their original value after \pause?

```
2475 \def\cl@@ckptpause@TP{\@elt{page}}
```

Save values of counters from \cl@@ckptpause@TP so that they can be restored with \restorepausecounters@TP.

```
2476 \def\savepausecounters@TP{%
```

```
2477   \begingroup
```

```
2478     \def\@elt##1{\global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
```

```
2479     \xdef\restorepausecounters@TP{\cl@@ckptpause@TP}%
```

```
2480   \endgroup
```

```
2481   }
```

User-level command to add a counter name to \cl@@ckptpause@TP.

```
2482 \newcommand{\pausesafecounter}[1]%
```

```
2483 {\expandafter\def\expandafter\cl@@ckptpause@TP\expandafter{\cl@@ckptpause@TP\@elt{#1}}}
```

Setting \pausesafecounter for common classes

```
2484 \@ifclassloaded{seminar}{\pausesafecounter{slide}}{}
```

Making some commands stepwise-aware (if we are in display modus) so we avoid duplicates when not using the old aggressive/robust filtering of whatsits.

```

2485 \ifthenelse{\boolean{display}}{%
2486 \AtBeginDocument{\@ifpackageloaded{hyperref}{%
2487 \let\hyper@anchor@TP=\hyper@anchor
2488 \long\def\hyper@anchor#1#2{%
2489   %% \ifthenelse{\boolean{instepwise@TP}}%
2490   \ifthenelse{\boolean{instepwise@TP} \AND \NOT \boolean{oldfiltering@TP}}%
2491   {\ifthenelse{\NOT \boolean{instep@TP}}%
2492     {\ifthenelse{\value{step}=\value{firststep}}{\@hyper@anchor#1\relax#2\relax}{}}%
2493     {\ifthenelse{\boolean{active} \AND \boolean{firstactivation}}%
2494       {\@hyper@anchor#1\relax#2\relax}{}}% End \ifthenelse{\boolean{active} ...
2495   }% End \ifthenelse{\NOT \boolean{instep@TP}}
2496   {\@hyper@anchor#1\relax#2\relax}% End \ifthenelse{\boolean{instepwise@TP}}
2497 }
2498 \let\hyper@anchorstart@TP=\hyper@anchorstart
2499 \def\hyper@anchorstart#1{%
2500   \ifthenelse{\boolean{instepwise@TP} \AND \NOT \boolean{oldfiltering@TP}}%
2501   {\ifthenelse{\NOT \boolean{instep@TP}}%
2502     {\ifthenelse{\value{step}=\value{firststep}}{\hyper@anchorstart@TP{#1}}{}}%
2503     {\ifthenelse{\boolean{active} \AND \boolean{firstactivation}}%
2504       {\hyper@anchorstart@TP{#1}}{}}% End \ifthenelse{\boolean{active} ...
2505   }% End \ifthenelse{\NOT \boolean{instep@TP}}
2506   {\hyper@anchorstart@TP{#1}}% End \ifthenelse{\boolean{instepwise@TP}}
2507 }
2508 }{}% End \AtBeginDocument{\@ifpackageloaded{hyperref}{
2509 % Changing \protected@write to avoid duplicates in aux-file
2510 \let\protected@write@TP=\protected@write
2511 \long\def\protected@write#1#2#3{%
2512   \ifthenelse{\boolean{instepwise@TP} \AND \NOT \boolean{oldfiltering@TP}}%
2513   {\ifthenelse{\NOT \boolean{instep@TP}}%
2514     {\ifthenelse{\value{step}=\value{firststep}}{\protected@write@TP{#1}{#2}{#3}}{}}%
2515     {\ifthenelse{\boolean{active} \AND \boolean{firstactivation}}%
2516       {\protected@write@TP{#1}{#2}{#3}}{}}% End \ifthenelse{\boolean{active} ...
2517   }% End \ifthenelse{\NOT \boolean{instep@TP}}
2518   {\protected@write@TP{#1}{#2}{#3}}% End \ifthenelse{\boolean{instepwise@TP}}
2519 }%
2520 }{}% End \ifthenelse{\boolean{display}}

```

Save the current page contents to the box `\mempageconts@TP` by overloading and triggering `\output`. Footnotes are also saved. The saved page contents are used by `\pause` and all `\stepwise` variants for making duplicates of page contents.

```

2521 \def\save@TP
2522 {%
2523   \par%                               Always end current paragraph.
2524   \global\setbox\mempageconts@TP=\copy\voidb@x%   Initialise \mempageconts@TP (suggested b
2525   \savepausecounters@TP%             Save counters.
2526   \memoutput@TP=\output%             Make backup copy of output routine.
2527   \output={\global\setbox\mempageconts@TP=\box@cclv}% Copy current page contents.
2528   \eject%                             Trigger now.

```

```

2529 \global\setbox\memfootins@TP=\copy\footins%           Save footnotes.
2530 \global\skip\memfootins@TP=\skip\footins
2531 \global\count\memfootins@TP=\count\footins
2532 \global\dimen\memfootins@TP=\dimen\footins
2533 \output=\memoutput@TP%                               Restore output routine.
2534 }

```

### 3.9.5 Kernel functions for re-inserting page contents

Used by `\pause` and also by variants of `\stepwise`.

Filter file and anchor whatsits out of a duplicate page. Called by `\shipout@hook@TP`.

Does nothing by default.

```
2535 \let\filterpage@TP\relax
```

Interpretation of `\filterpage@TP` for duplicate pages. Assigned by `\outputduplicate@TP`.

```

2536 \newcommand{\filterwhatsits@TP}{%
2537   \ifthenelse{\boolean{oldfiltering@TP}}{%
2538     {\global\setbox\@cclv=\hbox{\leaders\copy\@cclv\hskip\wd\@cclv}}% Old aggressive/robust fil
2539     }% No filtering - handled by \insertfilterwhatsits@TP and stepwise aware commands.
2540 }
2541 \newcommand{\insertfilterwhatsits@TP}{%
2542   \global\setbox\tempbox@TP=\hbox{\leaders\copy\tempbox@TP\hskip\wd\tempbox@TP}%
2543 }

```

Insert saved page contents for the first time.

```

2544 \newcommand{\insertfirstduplicate@TP}
2545 {\unvcopy\mempageconts@TP}%           Just output the vbox's contents.

```

Execute color correction stack accumulated when the `fixcolor` option is given.

```

2546 \newcommand{\do@colorcorrections@TP}
2547 {%
2548   {%
2549     \colorcorrections@TP%           Execute color correction stack.
2550     \gdef\colorcorrections@TP{\relax}% Re-initialize for next round.
2551   }%
2552 }

```

Insert saved page contents for the second time (and all further times).

```

2553 \newcommand{\insertsecondduplicate@TP}
2554 {%
2555   \do@colorcorrections@TP%           Execute color correction stack.
2556   \global\setbox\footins=\copy\memfootins@TP%       Restore footnotes.
2557   \global\skip\footins=\skip\memfootins@TP
2558   \global\count\footins=\count\memfootins@TP
2559   \global\dimen\footins=\dimen\memfootins@TP
2560   \ifthenelse{\boolean{oldfiltering@TP}}{%
2561     {\unvcopy\mempageconts@TP}%
2562     {\setbox\tempbox@TP=\vbox{\unvcopy\mempageconts@TP}%
2563     \insertfilterwhatsits@TP
2564     \copy\tempbox@TP%
2565   }

```

```

2566 }
      Ship out a duplicated page.
2567 \newcommand{\outputduplicate@TP}
2568 {%
2569   \shippingduplicatetrue%           This switch can be evaluated in panels or headed
2570   \global\let\filterpage@TP\filterwhatsits@TP% Filter out file whatsits when shipping out.
2571   \global\let\o@hyper@@anchor@TP\hyper@@anchor% Save definition of hyperref command for hyper a
2572   \global\let\hyper@@anchor@gobble% Disable hyper anchors on duplicate pages to avo
2573   \Tpeject%                         Shipout page.
2574   \global\let\hyper@@anchor\o@hyper@@anchor@TP% Restore definition of hyperref command for hype
2575   \global\let\filterpage@TP\relax%   Disable whatsit filtering.
2576   \shippingduplicatefalse%          Unset switch.
2577 }%

```

### 3.9.6 Implementation of `\pause`

`\pause` ends the current paragraph, ships out the current page, starts a new page and copies whatever was on the current page onto the new page, where typesetting is resumed. This will create the effect of a ‘pause’ in the presentation, i.e. the presentation stops because the current page ends at the point where the `\pause` command occurred and is resumed at this point when the presenter switches to the next page.

```

2578 \providecommand\pause% If pause.sty is loaded, the existing definition of \pause is not overwri
2579 {%
2580   \save@TP%                         Save contents of the page...
2581   \insertfirstduplicate@TP%         ... and insert again.
2582   \ifthenelse{\boolean{display}}}% Are dynamic features enabled?
2583   {% Yes.
2584     \outputduplicate@TP%           Output page.
2585     \restorepausecounters@TP%      Restore counters (page number).
2586     \insertsecondduplicate@TP%     Reinsert saved contents.
2587   }
2588 }%
2589 }%

```

### 3.9.7 Implementing `\stepwise` and all functions surrounding it

General usage: `\stepwise{<contents>}`

As of itself, `\stepwise` doesn’t do very much. If `<contents>` contains one or more constructs of the form `\step{<stepcontents>}`, the following happens:

1. The current paragraph is ended.
2. The current contents of the page are saved (as with `\pause`).
3. As many pages as there are `\step` commands in `<contents>` are produced. Every page starts with what was on the current page when `\stepwise` started. The first page also contains everything in `<contents>` which is

not in `<stepcontents>` for any `\step` command. The second page additionally contains the `<stepcontents>` for the first `\step` command, and so on, until all `<stepcontents>` are displayed.

4. When all `<stepcontents>` are displayed, `\stepwise` ends and typesetting is resumed (still on the current page).

This will create the effect that the `\step` commands are executed ‘step by step’.

For a more detailed description of `\stepwise`, `\step` and their options, see below.

Most of the registers, macros and environments defined in the following are part of the user interface, so no `@s`.

### 3.9.8 Command administration

```

2590 \newcommand{\@onlyinstepwise@TP}[1]
2591 {%
2592   \providecommand#1%
2593   {%
2594     \PackageError{texpower}%
2595     {Command \string#1 can be used only inside \string\stepwise}
2596     {%
2597       Commands like \string\step, \string\switch,
2598       \string\multistep\space or \string\overlays\MessageBreak
2599       can be used only inside the argument of a \string\stepwise\space variant.
2600     }%
2601   }%
2602 }
```

### 3.9.9 Registers

The total number of `\step` commands occurring in `<contents>`.

```
2603 \newcounter{totalsteps}
```

The number at which the counter step starts counting. Can be set in the optional argument of `\stepwise`.

```
2604 \newcounter{firststep}
```

The number of the step currently being performed.

```
2605 \newcounter{step}
```

The number of the current `\step` command (only useful inside `<stepcontents>`).

```
2606 \newcounter{stepcommand}
```

The total number of `\step` commands which have been activated so far (this can differ from `\value{step}` if the order of `\step` commands is changed via the optional argument of `\step`).

```
2607 \newcounter{stepsperformed}
```



Is this `\step` command currently active for the first time? (only useful inside `<stepcontents>`).

```
2608 \newboolean{firstactivation}
```

Is this `\step` command currently active? (only useful inside `<stepcontents>`).

```
2609 \newboolean{active}
```

### 3.9.10 Custom commands for displaying step contents

Some of them are selected by the switches `\boxedsteps` and `\nonboxedsteps`.

Display `<stepcontents>` in a box.

```
2610 \newcommand{\displayboxed}
```

```
2611 {%
```

```
2612   \ifmmode % We need to distinguish between math and
```

```
2613     \expandafter\mathpalette\expandafter\math@db@TP % In math mode, the style has to be respec
```

```
2614   \else
```

```
2615     \expandafter\text@db@TP
```

```
2616   \fi
```

```
2617 }
```

```
2618 \newcommand{\text@db@TP}[1]{\mbox{#1}} % In text mode, we just use an \mbox.
```

```
2619 \newcommand{\math@db@TP}[2]{\mbox{${\m@th#1}{#2}$}} % In math mode, the style (#1) is inserted by
```

Display `<stepcontents>` ‘as is’.

```
2620 \let\displayidentical=\@iden
```

### 3.9.11 Custom commands for ‘hiding’ stepcontents at the time the corresponding `\step` is not active.

Hiding stuff is not as easy as it seems... Often, it is desirable that an appropriate amount of ‘space’ is left where something is hidden, in case something visible follows or the hidden stuff is part of an alignment structure. Even if this is not the case, completely ignoring hidden text containing further `\step`’s can disturb the accounting of `\stepwise`, because `\step` numbers become unaligned. On the other hand, a lot of things which might be hidden (solitary `&`’s if the hidden text is part of an alignment structure, for instance) execute an implicit group closing and don’t like it at all to be enclosed in boxes, for instance. Because of these conflicting constraints, several methods for hiding content are provided. It is up to the user to select the one most appropriate for each type of content, or to use the suggestions below as inspiration for own definitions.

Hide `<stepcontents>`, but display an appropriate amount of white space in the form of an appropriately dimensioned, empty box.

```
2621 \newcommand{\hidephantom}[1]
```

```
2622 {%
```

```
2623   {% a group makes redefinitions local
```

```
2624     \leavevmode\phantom{#1}%
```

`\phantom` normally produces an `\hbox`. `\leavevmode` makes it behave like `\mbox`.

```
2625   }%
```

```
2626 }
```

Ignore `<stepcontents>` completely.

```
2627 \newcommand{\hideignore}[1]{}
```

Sometimes, ignoring `<stepcontents>` completely can lead to confusion of `\stepwise`'s accounting when `<stepcontents>` contains further `\step` commands. `\hidesmartignore` produces no output, but executes `<stepcontents>` (in a box). Note that this will lead to errors if for instance `<stepcontents>` contains a tabular character `&` from an alignment structure.

```
2628 \newcommand{\hidesmartignore}[1]{\setbox\tempbox@TP=\vbox{#1}}
```

The command `\hidetext` makes its argument 'invisible', but without putting it into a box, thus preserving line breaks. `\hidetext` needs the soul package to work. If it is not loaded, `\hidetext` is defined to be equal to `\hideboxed`. Because of restrictions implied by the soul package, `\hidetext` is quite picky about the 'regularity' of its argument. That is, a lot of things will break when in the argument of `\hidetext`. See the documentation of the soul package for hints how to prevent this.

To allow soul to be loaded after `texpower`, we use `\AtBeginDocument`.

```
2629 \AtBeginDocument%
```

```
2630 {%
```

```
2631   \ifpackageloaded{soul}{%
```

```
2632     \ifpackagelater{soul}{2002/05/28}{%
```

```
2633       \DeclareRobustCommand*\hidetext{%
```

To prevent `\phantom` from inserting an hbox into the vertical list.

```
2634     \leavevmode
```

```
2635     \SOUL@setup
```

Make current token 'invisible'.

```
2636     \def\SOUL@everytoken{\phantom{\the\SOUL@token\SOUL@setkern\SOUL@charkern}}
```

```
2637     \def\SOUL@everyhyphen
```

```
2638     {%
```

```
2639       \discretionary
```

```
2640       {%
```

```
2641         \unkern
```

Output an 'invisible' hyphen if needed.

```
2642         \phantom{\SOUL@setkern\SOUL@hyphkern\char\hyphenchar\font}%
```

```
2643       }{}{ }%
```

```
2644     }%
```

```
2645     \SOUL@%
```

```
2646   }%
```

```
2647 }{
```

Too old soul package - encourage people to update.

```
2648   \PackageInfo{texpower}
```

```
2649   {Package soul too old.\MessageBreak Command \string\hidetext\space disabled}%
```

Using `\hidephantom` which is a sorry excuse for `\hidetext`.

```
2650   \let\hidetext=\hidephantom
```

```
2651 }%
```

```

2652 }{
2653   \PackageInfo{texpower}
2654   {Package soul not loaded.\MessageBreak Command \string\hidetext\space disabled}%
2655   \let\hidetext=\hidephantom
2656 }%
2657 }

```

Helper command to switch to ‘dimmed’ textcolor or mathcolor (if in math mode and colormath option is set).

```

2658 \ifthenelse{\boolean{colormath@TP}}{%
2659   \newcommand{\commitcolor@TP}

```

Switch to mathcolor if in math mode, to textcolor otherwise.

```

2660   {\textcolor{\ifmmode mathcolor\else textcolor\fi}}
2661 }{
2662   \newcommand{\commitcolor@TP}{\textcolor{textcolor}}
2663 }

```

The command `\hidedimmed` doesn’t really make its argument ‘invisible’. Instead, it dims all colors so the argument ‘fades’ into the background.

```

2664 \ifthenelse{\boolean{TPcolor}}{
2665   \newcommand{\hidedimmed}[1]{%

```

Adding a group to make the color changes local.

```

2666   {

```

Just in case we currently are in dimmed mode.

```

2667     \ifthenelse{\boolean{instepwise@TP}}{\usecolorset{stwcolors}}{}%
2668     \dimcolors
2669     \commitcolor@TP{#1}
2670   }
2671 }
2672 }{
2673   \let\hidedimmed=\displayidentical
2674 }

```

The command `\hidevanish` makes its argument ‘invisible’ by switching to the background color. Of course, this only works if the background is uniformly colored. If we don’t use colors, we just use `\hidephantom`.

```

2675 \ifthenelse{\boolean{TPcolor}}{
2676   \newcommand{\hidevanish}[1]{\textcolor{vanishcolor}{vanishcolors#1}}
2677 }{
2678   \let\hidevanish=\hidephantom
2679 }

```

### 3.9.12 Displaying and hiding of step contents

The displaying and hiding of `<stepcontents>` is controlled by the commands `\displaystepcontents` and `\hidestepcontents`. The following switches define these to be either the boxed or the ‘as is’ versions defined above.

```

2680 \newcommand{\boxedsteps}           % Use boxed versions.

```

```

2681 {\let\displaystepcontents=\displayboxed\let\hidestepcontents=\hidephantom}
2682
2683 \newcommand{\nonboxedsteps}          % Use nonboxed versions.
2684 {\let\displaystepcontents=\displayidentical\let\hidestepcontents=\hideignore}

  The default is to use the nonboxed versions. This can be changed in the optional
  argument of \stepwise.
2685 \nonboxedsteps

  There is another command named \activatestep which controls what hap-
  pens when a \step command is activated for the first time. This is defined to do
  nothing by default.
2686 \let\activatestep=\displayidentical

  The length \highlightboxsep gives the width of the frame around the box
  created by \highlightboxed.
2687 \newlength{\highlightboxsep}
2688 \setlength{\highlightboxsep}{.5\fbboxsep}

  \highlightboxed{<text>} puts <text> into an \mbox with coloured back-
  ground if the colorhighlight option is set, and into an \fbox otherwise. As this
  is meant as an interpretation of \activatestep, it is made sure that the result-
  ing box has the same dimensions as the argument (the outer frame may overlap
  surrounding text).
2689 \DeclareRobustCommand{\highlightboxed}
2690 {%
2691   \ifmmode%                               Check for math mode.
2692     \expandafter\mathpalette\expandafter\math@hb@TP% Math mode version needs to respect curren
2693   \else
2694     \expandafter\text@hb@TP%               Text mode version.
2695   \fi
2696 }

  Math mode version of \highlightboxed.
2697 \newcommand{\math@hb@TP}[2]{\text@hb@TP{$\m@th#1{#2}$}}

  The text mode version of \highlightboxed does the 'real' work.
2698 \ifthenelse{\boolean{colorhighlight@TP}}%   Color highlighting enabled?
2699 {% Yes; use a box with colored background.
2700   \newcommand{\text@hb@TP}[1]
2701   {%
2702     \makebox[\width-2\highlightboxsep]%     Make the frame stick out at the side
2703     {%
2704       \setlength{\fbboxsep}{\highlightboxsep}% Set frame size.
2705       \raisebox{0pt}[\height-\fbboxsep][\depth-\fbboxsep]% Make the frame stick out above and b
2706       {\colorbox{highlightcolor}{#1}}%     Make colored box containing <text>.
2707     }%
2708   }%
2709 }% matches \ifthenelse{\boolean{colorhighlight@TP}}
2710 {% No; use an \fbox.
2711   \newcommand{\text@hb@TP}[1]

```

```

2712  {%
2713    \makebox[\width-2\highlightboxsep-2\fbboxrule]%           Make the frame stick out at the side
2714    {%
2715      \setlength{\fbboxsep}{\highlightboxsep}%               Set frame size.
2716      \raisebox{0pt}[\height-\fbboxsep-\fbboxrule][\depth-\fbboxsep-\fbboxrule]% Make the frame sti
2717      {\fbbox{#1}}}%%                                         Make an fbox containing <text>.
2718    }%
2719  }%
2720  }% matches second argument of \ifthenelse{\boolean{colorhighlight@TP}}

```

`\highlighttext` is the counterpart of `\highlightboxed` for arbitrary text. It puts its argument on a colored background without putting it into a box (i.e. line breaks and hyphenation still work) if the `colorhighlight` option is set, and underlines otherwise. As this is meant as an interpretation of `\activatestep`, it is made sure that the resulting text has the same dimensions as the argument (the outer frame may overlap surrounding text). `\highlighttext` needs the `soul` package to work. If it is not loaded, `\highlighttext` is defined to do nothing. Because of restrictions implied by the `soul` package, `\highlighttext` is quite picky about the ‘regularity’ of its argument. That is, a lot of things will break when in the argument of `\highlighttext`. See the documentation of the `soul` package for hints how to prevent this.

To allow `soul` to be loaded after `texpower`, we use `\AtBeginDocument`.

```

2721 \AtBeginDocument%
2722 {%
2723   \@ifpackageloaded{soul}%                                   Can we use the soul package?
2724   {%
2725     \@ifpackagelater{soul}{2002/05/28}%                     Correct version?
2726     {% Yes. Let’s define the necessary macros.
2727       \ifthenelse{\boolean{colorhighlight@TP}}%             Color highlighting enabled?
2728       {% Yes; use a colored background.
2729         % This is implemented as an application of soul (modifying the code for underline). Se
2730         % soul package for details on soul.
2731         \newlength{\SOUL@boxheight@TP}%                     Height of colored patch.
2732         \newlength{\SOUL@boxtotalheight@TP}%               Total height of colored patch.
2733         \newlength{\SOUL@boxdepth@TP}%                      Depth of colored patch.
2734         \DeclareRobustCommand*\highlighttext
2735         {%
2736           \leavevmode%                                       To prevent \smash from inserting an hb
2737           \SOUL@ulsetup%                                       Underline initialization.
2738           \def\SOUL@preamble
2739           {%
2740             \setlength{\SOUL@boxdepth@TP}%                   \SOUL@uldepth is below the depth o
2741             {\SOUL@uldepth+\highlightboxsep}%
2742             \def\SOUL@uldepth{-\SOUL@boxheight@TP}%         For correctly positioning the rule
2743             \setlength{\SOUL@boxheight@TP}{\heightof{/}+\highlightboxsep}%   Calcula
2744             \setlength{\SOUL@boxtotalheight@TP}{\SOUL@boxdepth@TP+\SOUL@boxheight@TP}% Calcula
2745             \def\SOUL@ulthickness{\SOUL@boxtotalheight@TP}% The thickness of the rule is the t
2746             \smash%                                           Make the left border of the colore
2747           }%

```

```

2748         \llap{\color{highlightcolor}\rule[-\SOUL@boxdepth@TP]{\highlightboxsep}{\SOUL@bo
2749     }%
2750     \SOUL@ulpreamble%                               Underline preamble.
2751 }%
2752 \def\SOUL@everytoken
2753 {%
2754     {%
2755         \setbox\tempbox@TP\hbox{\the\SOUL@token\SOUL@setkern\SOUL@charkern}%
2756         \dimen@ii\wd\tempbox@TP
2757         \smash{\rlap{\color{highlightcolor}\SOUL@ulleaders\hskip\dimen@ii}}%
2758         \unhbox\tempbox@TP%
2759         \smash{\rlap{\color{highlightcolor}\rule[-\SOUL@boxdepth@TP]{\highlightboxsep}{\
2760     }%
2761 }%
2762 \def\SOUL@everyspace
2763 {%
2764     \cleaders\hbox{\smash{\color{highlightcolor}\rule[-\SOUL@boxdepth@TP]{1pt}{\SOUL@b
2765     \hskip\spaceskip
2766     \smash{\llap{\color{highlightcolor}\rule[-\SOUL@boxdepth@TP]{\highlightboxsep}{\SO
2767 }%
2768 \def\SOUL@everyhyphen
2769 {%
2770     \discretionary
2771     {\unkern
2772     \setbox4\hbox{\SOUL@setkern\SOUL@hyphkern\char\hyphenchar\font}%
2773     \smash{\rlap{\color{highlightcolor}\SOUL@ulleaders\hskip\wd4}}%
2774     \box4%
2775     \smash{\rlap{\color{highlightcolor}\rule[-\SOUL@boxdepth@TP]{\highlightboxsep}{\SO
2776     }%
2777     {\smash{\llap{\color{highlightcolor}\rule[-\SOUL@boxdepth@TP]{\highlightboxsep}{\SO
2778     }%
2779     }%
2780     \let\SOUL@everysyllable\empty
2781 \SOUL@%
2782 }%
2783 }% matches \ifthenelse{\boolean{colorhighlight@TP}}%
2784 {% No. Underline.
2785 \DeclareRobustCommand*\highlighttext
2786 {%
2787     \SOUL@ulsetup%                               We modify SOUL's standard definition of underlin
2788     \def\SOUL@everysyllable%                       the result uses no more space than the non-un
2789     {%
2790         {%
2791             \let\o@rlap@TP=\rlap
2792             \def\rlap####1{\setbox\@tempboxa\box\z@\smash{\o@rlap@TP{####1}}\setbox\z@\box\@te
2793             \SOUL@uleverysyllable
2794         }%
2795     }%
2796 \def\SOUL@everyspace
2797 {\cleaders\hbox{\smash{\vrule\@depth\SOUL@uldp\@height\SOUL@ulht\@width.5pt}}\hskip\sp

```

```

2798     \def\SOUL@everyhyphen{\discretionary
2799         {\unkern
2800         \setbox4\hbox{\SOUL@setkern\SOUL@hyphkern\char\hyphenchar\font}%
2801         \smash{\rlap{\SOUL@ulleaders\hskip\wd4}\box4}}{}}%
2802     }%
2803     \SOUL@%
2804     }%
2805     }% matches second argument of \ifthenelse{\boolean{colorhighlight@TP}}%
2806     }%
2807     {% No. Encourage people to update.
2808     \PackageInfo{texpower}
2809     {Package soul too old.\MessageBreak Command \string\highlighttext\space disabled}%
2810     \let\highlighttext=@iden%
2811     }%
2812     }% matches \@ifpackageloaded{soul}
2813     {% No. In this case, there is no useful definition for \highlighttext.
2814     \PackageInfo{texpower}
2815     {Package soul not loaded.\MessageBreak Command \string\highlighttext\space disabled}%
2816     \let\highlighttext=@iden
2817     }% matches second argument of \@ifpackageloaded{soul}
2818     }% matches \AtBeginDocument%

    The command \highlightenhanced enhances all colors so the argument
    ‘stands out’.

2819 \ifthenelse{\boolean{TPcolor}}% Can we use colors at all?
2820 {% Yes.
2821     \newcommand{\highlightenhanced}[1]%           Make argument appear in ‘enhanced’ colors.
2822     {%
2823         {% A group makes the color changes local.
2824         \ifthenelse{\boolean{instepwise@TP}}{\usecolorset{stwcolors}}{}% Just in case we currentl
2825         \enhancecolors%           Enhance colors.
2826         \commitcolor@TP{#1}%     Switch on enhanced color.
2827         }%
2828     }%
2829     }
2830 {\let\highlightenhanced=\displayidentical}% No. Disable this command.

```

### 3.9.13 Implementation of \step, \switch and relatives

\step takes two optional arguments for influencing the mode of activation, like this:

```
\step[<activatefirst>][<whenactive>]{<stepcontents>}
```

Both <activatefirst> and <whenactive> should be conditions in the syntax of the \ifthenelse command.

<activatefirst> checks whether this \step is to be activated for the first time. The default value is \value{step}=\value{stepcommand}. By using \value{step}=<n>, this \step can be forced to appear as the n’t h one.

<whenactive> checks whether this \step is to be considered active at all. The default behaviour is to check whether this \step has been activated before (this

is saved internally for every step).

Both optional arguments allow two syntactical forms:

1. enclosed in square brackets [...] like explained above.
2. enclosed in braces (...). In this case, <activatefirst> and <whenactive> are not treated as conditions in the sense of `\ifthenelse`, but as conditionals like those used internally by L<sup>A</sup>T<sub>E</sub>X. That means, <activatefirst> (when enclosed in braces) can contain arbitrary T<sub>E</sub>Xcode which then takes two arguments and expands to one of them, depending on whether the condition is fulfilled or not fulfilled. For instance, `\step[<activatefirst>]{<stepcontents>}` could be replaced by `\step{\ifthenelse{<activatefirst>}}{<stepcontents>}`.

Internally, the default for the treatment of <whenactive> is (`\if@first@TP@true`), where `\if@first@TP@true` is an internal condition checking whether this `\step` has been activated before.

If you wish to give the second optional argument but not the first, just write `\step[] [<whenactive>] ...`

There are the following variants of `\step`. In all cases, the treatment of optional arguments for controlling activation are the same as for `\step`:

`\bstep[<activatefirst>] [<whenactive>]{<stepcontents>}` Like `\step`, but is always boxed.

`\switch[<activatefirst>] [<whenactive>]{<from>}{<to>}` Instead of hiding and displaying <stepcontents>, switch from <from> to <to>.

`\vstep[<activatefirst>] [<whenactive>]` Doesn't take an argument, but switches to 'invisible' color.

`\dstep[<activatefirst>] [<whenactive>]` Doesn't take an argument, but switches to dimmed colors.

`\steponce[<activatefirst>]{<stepcontents>}` Like `\step`, but goes inactive again in the subsequent step.

The following variants act like their counterparts, but appear at the same time as the previous `\step` (or variant).

`\restep[<activatefirst>] [<whenactive>]{<stepcontents>}`

`\rebstep[<activatefirst>] [<whenactive>]{<stepcontents>}`

`\reswitch[<activatefirst>] [<whenactive>]{<from>}{<to>}`

`\revstep[<activatefirst>] [<whenactive>]`

`\redstep[<activatefirst>] [<whenactive>]`

Optional argument handling.

`\pickup@s@optargs@TP` reads the optional arguments of `\step` (or `\switch`, or relatives) and then calls `\do@s@TP`.

```
2831 \newcommand{\pickup@s@optargs@TP}
2832 {%
2833   \ifnextchar[%]           Check for first optional argument in [...] syntax.
2834   {\f@brackstep@TP}%
2835   {%
2836     \ifnextchar(%)         Check for first optional argument in (...) syntax.
2837     {\f@parenstep@TP}
```



```

2838   {\f@brackstep@TP[]}%      No optional argument given; call with empty argument in [...] synt
2839   }%
2840 }
2841
2842 \def\f@brackstep@TP[#1]%      First optional argument was given in [...] syntax.
2843 {%
2844   \def\tmp@TP{#1}%
2845   \ifx\tmp@TP\empty%          Optional argument empty?
2846     \def\f@step@TPcheck{\ifthenelse{\value{step}=\value{stepcommand}}}% Yes; use default.
2847   \else
2848     \def\f@step@TPcheck{\ifthenelse{#1}}% No; condition is defined via \ifthenelse.
2849   \fi
2850   \f@step@TP%                  Go on and check for second optional argument.
2851 }
2852
2853 \def\f@parenstep@TP(#1)%      First optional argument was given in (...) syntax.
2854 {%
2855   \def\f@step@TPcheck{#1}%    Save condition (given as argument).
2856   \f@step@TP%                  Go on and check for second optional argument.
2857 }
2858
2859 \newcommand{\f@step@TP}%      Pick up the second optional argument.
2860 {%
2861   \@ifnextchar[%]             Check for second optional argument in [...] syntax.
2862   {\s@brackstep@TP}
2863   {%
2864     \@ifnextchar(%)           Check for second optional argument in (...) syntax.
2865     {\s@parenstep@TP}
2866     {\s@parenstep@TP(\if@first@TP@true)}% No second optional argument given;
2867     }%                          call with \if@first@TP@true in (...) syntax (default)
2868 }
2869
2870 \def\s@brackstep@TP[#1]%      Second optional argument was given in [...] syntax.
2871 {%
2872   \def\s@step@TPcheck{\ifthenelse{#1}}% Condition is defined via \ifthenelse.
2873   \@do@s@TP%                  Go on.
2874 }
2875
2876 \def\s@parenstep@TP(#1)%      Second optional argument was given in (...) syntax.
2877 {%
2878   \def\s@step@TPcheck{#1}%    Save condition (given as argument).
2879   \@do@s@TP%                  Go on.
2880 }

```

The following are needed to switch between the ‘switch behaviour’ and the ‘step behaviour’ of `\@@switch@TP`, which implements the functionality of both `\switch` and `\step`.

```

2881 \newcommand{\deactivate@inner@TP}% \switch behaviour.
2882 {%

```

Both `\inner@display@TP` and `\inner@hide@TP` just expand to their argument.

```
2883 \let\inner@display@TP=\displayidentical%
2884 \let\inner@hide@TP=\displayidentical%
2885 }
2886
2887 \newcommand{\activate@inner@TP}% \step behaviour.
2888 {%
```

Use the user interface macros `\displaystepcontents` and `\hidestepcontents`.

```
2889 \let\inner@display@TP=\displaystepcontents%
2890 \let\inner@hide@TP=\hidestepcontents%
2891 }
```

Implementation of `\step`.

```
2892 \@onlyinstepwise@TP\step
```

`\proper@step@TP` is the ‘real’ implementation of `\step`. Most of the time, `\step` is defined to execute `\proper@step@TP`.

```
2893 \newcommand{\proper@step@TP}{\let\do@s@TP=\@step@TP\pickup@s@optargs@TP}
\@step@TP implements the functionality of \step by calling \@switch@TP, which
points to \@switch@TP most of the time.
```

```
2894 \newcommand{\@step@TP}[1]{\activate@inner@TP\@switch@TP{#1}{#1}}
```

Inside, `\@step@TP` executes `\@switch@TP`, which is the implementation of the `\switch` command (see below).

Implementation of `\switch`.

```
2895 \@onlyinstepwise@TP\switch
```

`\switch` works exactly like `\step`, but it takes **two** mandatory arguments and selects the first if ‘not active’, the second if ‘active’.

`\proper@switch@TP` is the ‘real’ implementation of `\switch`. Most of the time, `\switch` is defined to execute `\proper@switch@TP`.

```
2896 \newcommand{\proper@switch@TP}{\deactivate@inner@TP\let\do@s@TP=\@switch@TP\pickup@s@optargs@TP}
```

`\if@first@TP@true` checks whether the `\switch` command number `\value{stepcommand}` has already been activated in this `\stepwise` session and selects one of its arguments accordingly.

```
2897 \newcommand{\if@first@TP@true}[2]%
2898 {%
2899   \expandafter                % \first@TP<n> is set to \undefined if \switch command number
2900   \ifx\csname first@TP@\the\c@stepcommand\endcsname\empty%          activated
2901     #1%
2902   \else
2903     #2%
2904   \fi
2905 }
```

`\switch` shouldn’t change the status quo in AMSLaTeX’s measuring pass in typesetting aligned formulae. To guarantee this, we check whether AMSLaTeX is

measuring with AMSLaTeX's \ifmeasuring@. When AMSLaTeX is not loaded, we provide this check ourselves.

```

2906 \ifundefined{ifmeasuring@}{\newif\ifmeasuring@}{}
    \@switch@TP implements the functionality of \switch. Most of the time,
    \@switch@TP (which is called after checking for optional arguments) is defined
    to execute \@switch@TP.
2907 \newcommand{\@switch@TP}[2]
2908 {%
2909   \global\advance\c@stepcommand by 1\relax% This execution of \switch is counted.
2910   \setboolean{instep@TP}{true}%           Set indicator.
2911   %
2912   % If the verbose option is set, type out some accounting information which can be used for de
2913   \ifthenelse{\boolean{verbose@TP}}
2914   {%
2915     \PackageInfo{texpower}
2916     {Step: \the\c@step, Stepcommand: \the\c@stepcommand,\MessageBreak Stepsperformed: \the\c@st
2917     }
2918   }%
2919   %
2920   \f@step@TPcheck% Is this step to be activated? \f@step@TPcheck is defined by the first option
2921   {% Yes.
2922     \if@first@TP@true{}% For non-unique conditions given as optional argument or when \reswitch
2923     {%       that \first@TP@value{stepcommand} is already set. In this case, don't advance the
2924     \ifmeasuring@\else% Do nothing in AMSLaTeX's measuring pass for aligned equations.
2925       \global\expandafter\let\csname first@TP@\the\c@stepcommand\endcsname=\empty% Set \first
2926       \fi
2927       \global\advance\c@stepsperformed by 1\relax% Advance the counter for 'real' first activat
2928       \gdef\current@step@TP{#2}%
2929       }%
2930       \setboolean{firstactivation}{true}% This switch can be tested in <stepcontents>, but also i
2931     }
2932   {% No.
2933     \setboolean{firstactivation}{false}% This switch can be tested in <stepcontents>, but also
2934     }% End of \f@step@TPcheck
2935   %
2936   \let\o@afterstep@TP=\afterstep% We need to save the current definition of \afterstep.
2937   %
2938   \s@step@TPcheck% Is this step active? \s@step@TPcheck is defined by the second optional argum
2939   {% Yes.
2940     \setboolean{active}{true}%           Make this fact known to the user.
2941     \ifthenelse{\boolean{firstactivation}}
2942     {\inner@display@TP{\activatstep{#2}}}% 'First' display of <stepcontents>.
2943     {\inner@display@TP{#2}}}%           Display <stepcontents>.
2944     }
2945   {% No.
2946     \setboolean{active}{false}%           Make this fact known to the user.
2947     \let\afterstep=\@gobble%           Don't execute \afterstep here.
2948     \ifthenelse{\boolean{firstactivation}}
2949     {\inner@hide@TP{\activatstep{#1}}}%   Hide <stepcontents>, but with 'first activation'.

```

```

2950     {\inner@hide@TP{#1}}%           Hide <stepcontents>.
2951   }% End of \s@step@TPcheck
2952   %
2953   \let\afterstep=\o@afterstep@TP%   Restore the definition of \afterstep.
2954   \setboolean{instep@TP}{false}%     Set indicator.
2955   }% End of the definition of \@switch@TP.

\restep is identical with \step, but is displayed at the same time as the
previous \step.
2956 \@onlyinstepwise@TP\restep
2957
2958 \newcommand{\proper@restep@TP}
2959 {%
2960   \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
2961   \proper@step@TP%                   Go on with \step.
2962   }

\reswitch is identical with \switch, but is displayed at the same time as the
previous \switch.
2963 \@onlyinstepwise@TP\reswitch
2964
2965 \newcommand{\proper@reswitch@TP}
2966 {%
2967   \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
2968   \proper@switch@TP%                 Go on with \switch.
2969   }

\bstep is a variant of \step which is always boxed.
2970 \@onlyinstepwise@TP\bstep
2971
2972 \newcommand{\proper@bstep@TP}{\let\@do@s@TP=\@bstep@TP\pickup@s@optargs@TP}
\@bstep@TP implements the functionality of \bstep by calling \boxedsteps and
\@step@TP.
2973 \newcommand{\@bstep@TP}[1]{\{\boxedsteps\@step@TP{#1}\}}

\rebstep is identical with \bstep, but is displayed at the same time as the
previous \bstep.
2974 \@onlyinstepwise@TP\rebstep
2975
2976 \newcommand{\proper@rebstep@TP}
2977 {%
2978   \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
2979   \proper@bstep@TP%                 Go on with \bstep.
2980   }

\dstep is a variant of \step which takes no argument, but switches colors to
‘dimmed’.
Helper command to switch to ‘dimmed’ textcolor or mathcolor (if in math
mode and colormath option is set).
2981 \ifthenelse{\boolean{colormath@TP}}% Should we color math?

```

```

2982 {% Yes.
2983   \newcommand{\commitcolors@TP}
2984   {\color{\ifmmode mathcolor\else textcolor\fi}}% Switch to mathcolor if in math mode, to textc
2985   }
2986 {% No.
2987   \newcommand{\commitcolors@TP}{\color{textcolor}}% Switch to textcolor.
2988   }
2989
2990 \@onlyinstepwise@TP\dstep
2991
2992 \newcommand{\proper@dstep@TP}{\deactivate@inner@TP\let\do@s@TP=\@dstep@TP\pickup@s@optargs@TP}
2993
2994 \ifthenelse{\boolean{TPcolor}}% Can we use colors at all?
2995 {% Yes.
2996   \newcommand{\@dstep@TP}{\@@switch@TP{\dimcolors\commitcolors@TP}{\set@color}}
2997   }
2998 {\newcommand{\@dstep@TP}{\@@switch@TP{}{}}}% No. Disable this command.
    \redstep is identical with \dstep, but is displayed at the same time as the
    previous \dstep.
2999 \@onlyinstepwise@TP\redstep
3000
3001 \newcommand{\proper@redstep@TP}
3002 {%
3003   \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
3004   \proper@dstep@TP%           Go on with \dstep.
3005   }
    \vstep is a variant of \step which takes no argument, but switches all colors
    to \vanishcolor.
3006 \@onlyinstepwise@TP\vstep
3007
3008 \newcommand{\proper@vstep@TP}{\deactivate@inner@TP\let\do@s@TP=\@vstep@TP\pickup@s@optargs@TP}
3009
3010 \ifthenelse{\boolean{TPcolor}}% Can we use colors at all?
3011 {% Yes.
3012   \newcommand{\@vstep@TP}{\@@switch@TP{\vanishcolors\color{textcolor}}{\set@color}}
3013   }
3014 {\newcommand{\@vstep@TP}{\@@switch@TP{}{}}}% No. Disable this command.
    \revstep is identical with \vstep, but is displayed at the same time as the
    previous \vstep.
3015 \@onlyinstepwise@TP\revstep
3016
3017 \newcommand{\proper@revstep@TP}
3018 {%
3019   \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
3020   \proper@vstep@TP%           Go on with \vstep.
3021   }
    \steponce[<activatefirst>]{<stepcontents>} is a variant of \step which

```

is active only at the time of activation and goes inactive again in the subsequent step.

```

3022 \@onlyinstepwise@TP\steponce
3023
3024 \newcommand{\proper@steponce@TP}
3025 {\@ifnextchar[{\brack@steponce@TP}{\@steponce@TP}}% Optional argument in square brackets?
3026
3027 \newcommand{\@steponce@TP}
3028 {%
3029   \@ifnextchar(%%           Optional argument in parentheses?
3030   {\paren@steponce@TP}%
3031   {\brack@steponce@TP[]}% [] is the default if no optional argument is given.
3032   }
3033
3034 \def\brack@steponce@TP[#1]%
3035 {%
3036   \def\optarg@so@TP{[#1]}% Store optional argument for later re-insertion.
3037   \@@steponce@TP%         Proceed.
3038   }%
3039
3040 \def\paren@steponce@TP(#1){\def\optarg@so@TP{(#1)}\@@steponce@TP}

```

Main body of \steponce.

```

3041 \newcommand{\@@@steponce@TP}[1]{\expandafter\step\optarg@so@TP[\boolean{firstactivation}]{#1}}

```

\multistep\* [<activatefirst>]{<n>}{<stepcontents>} is a shorthand macro for executing several steps successively. In fact, it would better be called \multiswitch, because its functionality is based on \switch, it only acts like a (simplified) \step command which is executed ‘several times’. \multistep [<activatefirst>]{<n>}{<stepcontents>} expands to a sequence of <n> commands of the form \switch [<activatefirst>][\boolean{firstactivation}] with the effect that <stepcontents> is executed <n> times at different iterations of \stepwise. Note that [<activatefirst>] can also have the form (<activatefirst>), as usual for \switch. Because of the second optional argument [\boolean{firstactivation}], only one instance of <stepcontents> is displayed at a time. Inside <stepcontents>, a counter substep can be evaluated which tells the number of the current instance. In the starred form, the optional argument [\boolean{firstactivation}] is left out for the very last instance, so the last instance of <stepcontents> stays visible.

New counter for the number of the current substep.

```

3042 \newcounter{substep}

```

User interface for \multistep.

```

3043 \@onlyinstepwise@TP\multistep
3044
3045 \newcommand{\proper@multistep@TP}
3046 {%
3047   \let\ns@ms@TP=\normalstep@ms@TP% Placeholder for ‘every step but the last one’.
3048   \let\nshook@ms@TP=\relax%         These hooks are used by \movie.
3049   \let\lshook@ms@TP=\relax

```

```

3050 \@ifstar%                               Starred version?
3051 {%
3052   \let\ls@ms@TP=\laststep@ms@TP% Last step acts differently.
3053   \multistep@TP%                           Collect optional argument and proceed.
3054   }
3055 {%
3056   \let\ls@ms@TP=\normalstep@ms@TP% Last step acts like all other steps.
3057   \multistep@TP%                           Collect optional argument and proceed.
3058   }%
3059 }
3060
3061 \newcommand{\multistep@TP}% Collect optional argument.
3062 {\@ifnextchar[{\brack@multistep@TP}{\@multistep@TP}}% Optional argument in square brackets?
3063
3064 \newcommand{\@multistep@TP}
3065 {%
3066   \@ifnextchar(%%                           Optional argument in parentheses?
3067   {\paren@multistep@TP}%
3068   {\brack@multistep@TP[]}% [] is the default if no optional argument is given.
3069   }
3070
3071 \def\brack@multistep@TP[#1]%
3072 {%
3073   \def\optarg@ms@TP{[#1]}% Store optional argument for later re-insertion.
3074   \@@multistep@TP%                           Proceed.
3075   }%
3076
3077 \def\paren@multistep@TP(#1){\def\optarg@ms@TP{(#1)}\@@multistep@TP}
3078
3079 Execute one step.
3079 \newcommand{\normalstep@ms@TP}[1]
3080 {%
3081   \expandafter\switch\optarg@ms@TP% Re-insert optional argument.
3082   [\boolean{firstactivation}]{#1}% 'normal' steps appear only once.
3083   }
3084
3085 \newcommand{\laststep@ms@TP}[1]
3086 {\expandafter\switch\optarg@ms@TP{#1}}% In the starred version, the last step doesn't disappear.
3087 Main body of \multistep.
3087 \newcommand{\@@multistep@TP}[2]
3088 {%
3089   \setcounter{substep}{0}% Initialize substep counter.
3090   \whiledo{\value{substep}<#1}% Iterate <n> times.
3091   {%
3092     \stepcounter{substep}%
3093     \ifthenelse{\value{substep}=#1}% Last step?
3094     {\ls@ms@TP{\lshook@ms@TP#2}}% Execute single step (together with \movie hooks).
3095     {\ns@ms@TP{\nshook@ms@TP#2}}%
3096     }%

```

```

3097 }
    \movie* [<activatefirst>]{<n>}{<dur>} [<stop>]{<stepcontents>} works
    like \multistep, but between \steps, pages are advanced automatically every
    <dur> seconds. The additional optional argument <stop> gives the code (default:
    \stopAdvancing) which stops the animation. User interface for \movie.
3098 \@onlyinstepwise@TP\movie
3099
3100 \newcommand{\proper@movie@TP}
3101 {%
3102   \let\ns@ms@TP=\normalstep@ms@TP% Placeholder for 'every step but the last one'.
3103   \def\nshook@ms@TP{\afterstep{\pageDuration{\dur@ms@TP}}}% Page duration to be used between st
3104   \def\lshook@ms@TP{\afterstep{\end@ms@TP}}% Page duration setting after last st
3105   \ifstar% Starred version?
3106   {%
3107     \let\ls@ms@TP=\laststep@ms@TP% Last step acts differently.
3108     \movie@TP% Collect optional argument and proceed.
3109   }
3110   {%
3111     \let\ls@ms@TP=\normalstep@ms@TP% Last step acts like all other steps.
3112     \movie@TP% Collect optional argument and proceed.
3113   }%
3114 }
3115
3116 \newcommand{\movie@TP}% Collect optional argument.
3117 {\@ifnextchar[{\brack@movie@TP}{\@movie@TP}}% Optional argument in square brackets?
3118
3119 \newcommand{\@movie@TP}
3120 {\@ifnextchar[{\paren@movie@TP}{\brack@movie@TP[]}}% Optional argument in parentheses?
3121
3122 \def\brack@movie@TP[#1]{\def\optarg@ms@TP{[#1]}\@movie@TP}% Store optional argument and procee
3123 \def\paren@movie@TP(#1){\def\optarg@ms@TP{(#1)}\@movie@TP}
3124
3125 \newcommand{\@@movie@TP}[2]% Collect <n> and <dur> arguments.
3126 {\gdef\dur@ms@TP{#2}\gdef\nosteps@ms@TP{#1}\@@@movie@TP}
3127
3128 \newcommand{\@@@movie@TP}[1][\stopAdvancing]% Collect second optional argument and call body of
3129 {\gdef\end@ms@TP{#1}\@@multistep@TP{\nosteps@ms@TP}}

```

`\overlays [<activatefirst>]{<n>}{<stepcontents>}` is another shorthand macro for executing several steps successively. In contrast to `\multistep`, it doesn't print things **after** each other, but **over** each other. Obviously, there is no need for a starred version. `\overlays [<activatefirst>]{<n>}{<stepcontents>}` expands to a sequence of `<n>` commands of the form `\switch [<activatefirst>]{\rlap{<stepcontents>}}` with the effect that `<stepcontents>` is executed `<n>` times at different iterations of `\stepwise`, and all results are overlaid over each other. Note that `[<activatefirst>]` can also have the form `(<activatefirst>)`, as usual for `\switch`. Inside `<stepcontents>`, a counter substep can be evaluated which tells the number of the current instance. User interface for `\overlays`.



```

3130 \@onlyinstepwise@TP\overlays
3131
3132 \providecommand{\proper@overlays@TP}
3133 {\@ifnextchar[{\brack@overlays@TP}{\@overlays@TP}}% Optional argument in square brackets?
3134
3135 \newcommand{\@overlays@TP}
3136 {%
3137   \@ifnextchar(%%           Optional argument in parentheses?
3138   {\paren@overlays@TP}%
3139   {\brack@overlays@TP[]}% [] is the default if no optional argument is given.
3140   }
3141
3142 \def\brack@overlays@TP[#1]%
3143 {%
3144   \def\optarg@ov@TP{#1}% Store optional argument for later re-insertion.
3145   \@@overlays@TP%       Proceed.
3146   }%
3147
3148 \def\paren@overlays@TP(#1){\def\optarg@ov@TP{#1}\@@overlays@TP}
3149
3150 Main body of \overlays.
3149 \newcommand{\@@overlays@TP}[2]
3150 {%
3151   \setcounter{substep}{1}%           Initialize substep counter.
3152   \leavevmode%                       Make sure that \rlap doesn't insert anything in the vertical l
3153   \whiledo{\value{substep}<#1}% Iterate <n-1> times (since we start at 1).
3154   {%
3155     \expandafter\switch\optarg@ov@TP{}{\ifthenelse{\boolean{firstactivation}}{\mbox{#2}}{\rlap{
3156     \stepcounter{substep}%
3157     }}%
3158     \expandafter\switch\optarg@ov@TP{}{\mbox{#2}}% Always using \mbox for last overlay.
3159   }

```

### 3.9.14 Implementation of \stepwise

Every variant of `\stepwise` takes an optional argument, like this

```
\stepwise[<settings>]{<contents>}
```

`<settings>` will be placed right before the internal loop which produces the sequence of pages. It can contain settings of parameters which modify the behaviour of `\stepwise` or `\step`. `<settings>` is placed inside a group so all changes are local to this call of `\stepwise`.

Usually, the first page of a sequence produced contains only material which is not part of any `<stepcontents>`. The first `<stepcontents>` are displayed on the second page of the sequence.

For special effects, it might be desirable to have the first `<stepcontents>` active even on the first page of the sequence.

All variants of `\stepwise` have a starred version (e. g. `\stepwise*`) which does exactly that.

When `\stepwise` is executed, for every page of the sequence generated, `<contents>` is wrapped in the environment `stepcapsule` (but not grouped by default). This is empty by default for minimum intrusion. Redefine `stepcapsule` in the optional argument of `\stepwise` to change this (as is done for instance by `\liststepwise`).

```
3160 \newenvironment{stepcapsule}{}{}
```

Because `\step` commands usually occur deep in some nested structure, it is difficult to set local parameters (like page transitions) only for certain steps (local settings executed in `<stepcontents>` would be undone by closing groups).

`\afterstep{<setting>}` has the effect that `<setting>` will be performed **after** the current execution of `<contents>`, right before the page break for this page of the sequence generated.

```
3161 \newcommand{\afterstep}[1]
```

```
3162 {%
```

```
3163   \gdef\@afterstep@TP{#1}% The argument is simply stored in \@afterstep@TP, which is executed i
```

```
3164   }
```

One new counter for saving the value of `firststep`.

```
3165 \newcounter{o@fs@TP}
```

`\stepwise` user interface.

```
3166 \newcommand{\stepwise}
```

```
3167 {%
```

```
3168   \global\c@o@fs@TP=\c@firststep\relax%           Save the default value of counter firststep.
```

```
3169   \ifstar%                                         Using the starred version?
```

```
3170   {% Yes.
```

```
3171     \c@firststep=1\relax%                           Start with counter step at number 1.
```

```
3172     \@stepwise@TP%                                   Collect optional argument and proceed.
```

```
3173   }
```

```
3174   {% No.
```

```
3175     \@stepwise@TP%                                   Use the default.
```

```
3176   }% End of \ifstar.
```

```
3177   }%
```

Sometimes, it might happen that vertical spacing is different on every page of a sequence generated by `\stepwise`, making lines ‘wobble’. There are two custom versions `\liststepwise`, `\parstepwise` of `\stepwise` which produce better vertical spacing by putting an invisible rule before `<contents>`. This will almost certainly lead to ‘consistent’ spacing which might nevertheless be different from the spacing if `\liststepwise` wasn’t present.

`\liststepwise{<contents>}` works exactly like `\stepwise`, but `<contents>` is delimited by a rule of height zero. Use for list environments and aligned equations.

`\parstepwise{<contents>}` works like `\liststepwise`, but `\boxedsteps` is turned on by default. Use for texts where steps are to be filled into blank spaces.

Command to activate special `stepcapsule` for `\liststepwise`.

```
3178 \newcommand{\liststepcapsule}
```

```
3179 {%
```

```

3180 \renewenvironment{stepcapsule}%           stepcapsule is to put an invisible rule on
3181 {\vspace*{\parskip}\hrule \@height\z@\relax}{ }
3182 }%
3183
3184 \let\par@stepcapsule=\list@stepcapsule%   Identical for \parstepwise.
    User interface for \liststepwise.
3185 \newcommand{\liststepwise}
3186 {%
3187   \@ifstar%                               Starred version?
3188   {\def\star@TP{*}\@liststepwise@TP}% Save star in \star@TP, collect optional argument and proceed
3189   {\def\star@TP{}\@liststepwise@TP}% Collect optional argument and proceed (non-starred version)
3190   }
3191
3192 \newcommand{\@liststepwise@TP}[1] []% Collect optional argument.
3193 {%
3194   \expandafter\stepwise\star@TP%          Re-insert the star (if given).
3195   [%
3196     \list@stepcapsule%                    Activate special stepcapsule.
3197     #1%                                    Insert optional argument of \liststepwise
3198   ]%
3199   }
    User interface for \parstepwise.
3200 \newcommand{\parstepwise}
3201 {%
3202   \@ifstar%                               Starred version?
3203   {\def\star@TP{*}\@parstepwise@TP}% Save star in \star@TP, collect optional argument and proceed
3204   {\def\star@TP{}\@parstepwise@TP}% Collect optional argument and proceed (non-starred version)
3205   }
3206
3207 \newcommand{\@parstepwise@TP}[1] []% Collect optional argument.
3208 {%
3209   \expandafter\stepwise\star@TP%          Re-insert the star (if given).
3210   [%
3211     \boxedsteps%                          Activate \boxedsteps.
3212     \par@stepcapsule%                     Activate special stepcapsule.
3213     #1%                                    Insert optional argument of \parstepwise.
3214   ]%
3215   }
    \count@em@TP is used by \stepwise as a redefinition of \@switch@TP for
    counting \step commands.
3216 \newcommand{\count@em@TP}[2]
3217 {%
3218   \global\advance\c@stepcommand by 1\relax% We simply advance the number of \step commands...
3219   #2%                                       ... and execute the second argument (to find nested u
3220   }
    \savecounters@TP saves the values of all counters that have ever been defined
    by \newcounter in the macro \restorecounters@TP, which can be used later to

```

restore the saved values. The code is snarfed from amsmath.sty. During execution of `\stepwise`, this is used to restore the values of all counters between steps so that things counted in the argument of `\stepwise` (like equation numbers) do not ‘run wild’.

```

3221 \def\nb@TPfalse{\global\let@if@nb@TP\iffalse}
3222 \def\nb@TPtrue{\global\let@if@nb@TP\iftrue}
3223 \newtoks\ep@TP
3224 \def\savecounters@TP{%
3225   \begingroup
3226     \def@elt##1{\global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
3227     \xdef\restorecounters@TP{\cl@@ckpt}%
3228   \endgroup
3229   \if@nobreak\nb@TPtrue\else\nb@TPfalse\fi
3230   \global\ep@TP\everypar
3231 }

```

`\saveTPcounters@TP` saves the values of all ‘stepwise-specific’ counters in the macro `\restoreTPcounters@TP`. This is used to ‘counteract’ `\restorecounters@TP`, leaving the values of ‘stepwise-specific’ counters intact.

```

3232 \def\saveTPcounters@TP{%
3233   \begingroup
3234     \def@elt##1{\global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
3235     \xdef\restoreTPcounters@TP{\cl@@ckpt@TP}%
3236   \endgroup
3237 }

```

This list gives the names of all ‘stepwise-specific’ counters.

```

3238 \def\cl@@ckpt@TP{\@elt{totalsteps}\@elt{firststep}\@elt{step}\@elt{stepcommand}\@elt{stepsperfo

```

`\releasecounter{<name>}` inserts `<name>` into the list `\cl@@ckpt@TP`. This way, the counter `<name>` is not restored between steps.

```

3239 \newcommand{\releasecounter}[1]%
3240 {\expandafter\def\expandafter\cl@@ckpt@TP\expandafter{\cl@@ckpt@TP\@elt{#1}}}

```

`\disable@counting@TP` is executed when counting `\step` commands. Everything the execution of which would be harmful during counting, or which needs much computing resources, can be disabled here.

```

3241 \newcommand{\disable@counting@TP}
3242 {%
3243   \let\afterstep=\gobble
3244   \renewcommand{\backgroundstyle}[2] [] {}%
3245   \renewcommand{\@vgradrule@TP}[3] [Opt] {}%
3246   \renewcommand{\@@@dblgradrule@TP}[3] [Opt] {}%
3247   \renewcommand{\@hgradrule@TP}[3] [Opt] {}%
3248   \renewcommand{\@@@dblhgradrule@TP}[3] [Opt] {}%
3249 }

```

The boolean `instepwise@TP` indicates whether the execution of `\stepwise` is currently going on.

```

3250 \newboolean{instepwise@TP}

```

The boolean `instep@TP` indicates whether we are inside a `\step` command.

```

3251 \newboolean{instep@TP}
      \@stepwise@TP implements the functionality of \stepwise. It is called by
      \stepwise after checking for the star.
3252 \newcommand{\@stepwise@TP}[2] []
3253 {%
3254   \save@TP%           Save the current contents of the page.
3255   \savecounters@TP%  Save the values of all counters.
3256   \dumpcolorset{stwcolors}% Make a copy of all color definitions.
3257   \begin@group%      A group makes redefinitions local.
3258   \setboolean{instepwise@TP}{true}% Set indicator.
3259   \let\step\proper@step@TP%      By default, \step executes \proper@step@TP.
3260   \let\restep\proper@restep@TP%
3261   \let\bstep\proper@bstep@TP%
3262   \let\restep\proper@restep@TP%
3263   \let\dstep\proper@dstep@TP%
3264   \let\redstep\proper@redstep@TP%
3265   \let\vstep\proper@vstep@TP%
3266   \let\revstep\proper@revstep@TP%
3267   \let\steponce\proper@steponce@TP%
3268   \let\multistep\proper@multistep@TP%
3269   \let\movie\proper@movie@TP%
3270   \let\overlays\proper@overlays@TP%
3271   \let\switch\proper@switch@TP%      By default, \switch executes \proper@switch@TP.
3272   \let\reswitch\proper@reswitch@TP%

```

One big problem with `math` is that `\mathchoice` typesets its argument four times. If `\step` commands are inside the argument of `\mathchoice`, counters (which are advanced globally by `\step`) go astray. So far, I don't know any remedy for this apart from (locally) hacking `\mathchoice`. I know this is a very fragile and non-recommended method, but it works for the examples and will hopefully do until someone helps me find a better solution.

```

3273   \let\orig@mathchoice@TP=\mathchoice% Save the current definition of \mathchoice...
3274   \def\mathchoice##1##2##3##4%      ... and redefine.
3275   {%
3276     \orig@mathchoice@TP%           The original definition of \mathchoice is called...
3277     {##1}%                         ... with the first argument untouched and in all other
3278     {\let\step\restep\let\bstep\restep\let\dstep\restep\let\vstep\restep\let\switch\restep
3279     {\let\step\restep\let\bstep\restep\let\dstep\restep\let\vstep\restep\let\switch\restep
3280     {\let\step\restep\let\bstep\restep\let\dstep\restep\let\vstep\restep\let\switch\restep
3281     }%
3282   %
3283   \c@stepcommand=0\relax%          Initialize the counter for \step commands.
3284   \let\@switch@TP=\count@em@TP%    Next, we count the \step commands in <contents>...
3285   \setbox\tempbox@TP%              ... by putting <contents> into a \vbox (which is the
3286   =\vbox
3287   {%
3288     \hfuzz\maxdimen\hbadness\@M\relax% No bogus 'overfull \hbox' warnings.
3289     \disable@counting@TP#2%        Inside the \vbox, some commands are redefined for saf

```

```

3290 }%
3291 \c@totalsteps=\c@stepcommand%           Now, we know the total number of \step commands.
3292 %
3293 % Next, we have to reset \first@TP@<n> for <n>=0...\value{totalsteps}.
3294 \c@step=0\relax
3295 \loop
3296 \ifnum\c@step<\c@totalsteps
3297 \advance\c@step by 1\relax
3298 \expandafter\let\csname first@TP@\the\c@step\endcsname=\@undefined% \first@TP@<n>=\@undef
3299 % hasn't yet been activ
3300 \repeat

Next, the optional argument of \stepwise is executed. At this point, \value{totalsteps}
already has its final value and \value{step} has not yet been set to \value{firststep},
so both totalsteps and firststep can meaningfully be modified in <settings>.

3301 #1%
3302 %
3303 \c@step=\c@firststep%                   Set the counter for the current step to its startin
3304 \c@stepsperformed=0\relax%             ... and also the counter for the \step commands whi
3305 \let\@switch@TP=\@switch@TP%         \step will now act normally.
3306 \ifthenelse{\boolean{verbose@TP}}%    Some accounting info (if verbose option is set).
3307 {\PackageInfo{texpower}{Total number of step commands: \the\c@totalsteps}}
3308 {}%
3309 %
3310 \ifthenelse{\boolean{display}}%       Are dynamic features enabled?
3311 {}% Yes.
3312 {% No. Do only one loop.
3313 \c@stepsperformed=\c@totalsteps%     Set everything up for the last loop.
3314 \c@step=0\relax%                     Set \first@TP@<n> for <n>=0...\value{totalsteps}.
3315 \loop
3316 \ifnum\c@step<\c@totalsteps
3317 \advance\c@step by 1\relax
3318 \expandafter\let\csname first@TP@\the\c@step\endcsname=\@empty% \first@TP@<n>=\@empty m
3319 % has already been activa
3320 \repeat
3321 \advance\c@step by 1\relax%           This way, the last step won't think it's 'first act
3322 }%
3323 \let\insertdup@TP=\insertfirstduplicate@TP% Setup command to restore page contents for the
3324 \loop%                                 This is the central loop.
3325 \c@stepcommand=0\relax%               Initialize the counter for the current \step comman
3326 \saveTPcounters@TP%                   Save the 'stepwise-specific' counters.
3327 \restorecounters@TP%                   Restore the 'original' values of all counters...
3328 \restoreTPcounters@TP%                 ... and the current values of the 'stepwise-specifi
3329 \let\@afterstep@TP=\relax%            Reset the container for \afterstep.
3330 %
3331 \insertdup@TP%                           Insert saved page contents.
3332 \begin{stepcapsule}%                     This is usually empty, but may start a minipage (or
3333 \usecolorset{stwcolors}%                 Restore colors to state at the beginning of \stepwi
3334 \if@nb@TP\@nbreaktrue\else\@nbreakfalse\fi
3335 \global\everypar\ep@TP

```

```

3336         #2%                               Execute <contents>
3337     \end{stepcapsule}%
3338     %
3339     \@afterstep@TP%                         Whatever has been saved with \afterstep is now executed
3340     \ifnum\c@stepsperformed<\c@totalsteps% Doing one more round?
3341     \outputduplicate@TP%                     Shipout this page and round we go again.
3342     \let\insertdup@TP=\insertsecondduplicate@TP% Setup command to restore page contents for step
3343     \advance\c@step by 1\relax%             Round we go again
3344     \repeat
3345     \endgroup
3346     \global\c@firststep=\c@o@fs@TP\relax%   Restore default value of counter firststep.
3347 }

```

### 3.9.15 Implementation of the fragilesteps environment

Defining fragilesteps - an environment for fragile/verbatim stuff. The code is contributed by Till Tantau, the author of the excellent presentation class beamer.

```

3348 \newenvironment{fragilesteps}{%
3349     \def\texpower@verbatimfilename{\jobname-texpower.vrb}%
3350     \immediate\openout\texpower@verbatimfileout=\texpower@verbatimfilename%
3351     \texpower@verbatimreadframe%
3352 }
3353 {%
3354     \immediate\closeout\texpower@verbatimfileout%
3355     \stepwise{\input{\texpower@verbatimfilename}}%
3356 }

```

Internals used in the fragilesteps environment.

```

3357 \newwrite\texpower@verbatimfileout
3358
3359 \def\texpower@verbatimreadframe{%
3360     \begingroup%
3361     \let\do\@makeother\dospecials%
3362     \count@=127%
3363     \@whilenum\count@<255 \do{%
3364         \advance\count@ by 1%
3365         \catcode\count@=11%
3366     }%
3367     \@makeother\^^L% and whatever other special cases
3368     \endlinechar'\^^M \catcode'\^^M=12 \texpower@processframefirstline}
3369
3370 {\catcode'\^^M=12\endlinechar=-1%
3371     \long\gdef\texpower@processframefirstline#1^^M{%
3372         \def\texpower@test{#1}%
3373         \ifx\texpower@test\texpower@stopframefirst%
3374             \let\next=\texpower@endfragilesteps%
3375         \else
3376             \ifx\texpower@test\@empty%
3377                 \else%

```

```

3378     \@temptokena={#1}%
3379     \immediate\write\texpower@verbatimfileout{\the\@temptokena}%
3380     \fi%
3381     \let\next=\texpower@processframeline%
3382     \fi%
3383     \next%
3384 }
3385 \long\gdef\texpower@processframeline#1^^M{%
3386   \def\texpower@test{#1}%
3387   \ifx\texpower@test\texpower@stopframe%
3388     \let\next=\texpower@endfragilesteps%
3389   \else
3390     \immediate\write\texpower@verbatimfileout{#1}%
3391     \fi%
3392     \next%
3393 }
3394 }
3395
3396 {
3397   \escapechar=-1\relax%
3398   \xdef\texpower@stopframe{\string\end\string\{fragilesteps\string\}}
3399   \xdef\texpower@stopframefirst{\noexpand\end\string\{fragilesteps\string\}}
3400 }
3401
3402 \def\texpower@endfragilesteps{\endgroup\end{fragilesteps}}

```

### 3.9.16 Input system-specific settings

If file exists.

```
3403 \InputIfFileExists{tpsettings.cfg}{-}{-}
```

## Change History

v0.0.1		incompatible versions of hyper-
General: First pre-alpha version. . .	2	ref (spotted by Marc van Don-
v0.0.2		gen). (Apr 14: this code no
General: Squashing a bug... . . . . .	2	longer exists) . . . . . 2
v0.0.3		v0.0.6
General: Tidying up command syn-		General: Added papersize settings.
tax; adding some in-line docu-		(Mar 28: these are now in
mentation. . . . .	2	fixseminar.sty) . . . . . 2
v0.0.4		v0.0.7
General: In-line documentation for		General: Removed dependency on
the first pre-alpha version com-		hyperref; added support for
pleted. . . . .	2	color emphasis; respect the dis-
v0.0.5		play option; now loading tpset-
General: Fixed some problems with		tings.cfg and tpoptions.cfg;



	added <code>\bstep</code> , <code>\switch</code> , <code>\rebstep</code> , <code>\reswitch</code> . . . . .	2	
v0.0.7a	General: <code>\pause</code> and <code>\stepwise</code> now use <code>\leaders</code> for insert- ing duplicated parts of pages. This way, processing of what- sits is turned off in the dupli- cates so that table of contents entries are no longer duplicated when a section occurs on a page where <code>\pause</code> or <code>\stepwise</code> is used (spotted by heiner richter). Thanks to Heiko Oberdiek for his suggestion how <code>\leaders</code> can be successfully applied for this purpose. <code>\stepwise</code> now does the right thing if no <code>\step</code> command occurs in contents. There was a bug in <code>\save@TP</code> which would become apparent if <code>\stepwise</code> was the first thing on a page. Spotted and fixed by Ross Moore (thanks). . . . .	2	v0.0.8a General: Fixed a bug in the code which disables <code>\pageDuration</code> if the <code>pdfpageduration</code> key doesn't exist (spotted by Friedrich Eisenbrand). . . . .
			v0.0.8b General: Added additional 'dimmed' and 'enhanced' color sets for all standard colors, with corresponding commands <code>\dimcolors</code> and <code>\enhancecolors</code> . Added a 'color stack correction' option <code>fixcolorstack</code> , which should avoid that the dupli- cation of "color push" and "color pop" specials con- fuses the driver's color stack (spotted by Ross Moore). Added new display commands <code>\hidedimmed</code> , <code>\hidevanish</code> , <code>\highlightenhanced</code> . Added <code>\step</code> variants <code>\dstep</code> and <code>\vstep</code> . Added patches for <code>\[</code> , <code>equation</code> , <code>eqnarray</code> , and <code>eqnar- ray*</code> when the <code>colormath</code> option is used. Now also saving and restoring footnotes at <code>\pause</code> and <code>\stepwise</code> . Added a com- mand <code>\releasecounter</code> to keep a counter from 'freezing' during the execution of <code>\stepwise</code> . . . . .
v0.0.7b	General: Changed hyperref ver- sion check from 2000/03/22 to 2000/03/23. Spotted by Ross Moore. . . . .	2	
v0.0.7c	General: <code>\eject</code> changed to <code>\newpage</code> in <code>\stepwise</code> to cure some problems with the foils package. Spotted by Ross Moore. . . . .	2	
v0.0.7d	General: <code>\everydisplay</code> finally re- moved from <code>colormath</code> option because it only causes trou- ble. Maybe I should look for a less fragile solution for the whole thing. Fixed a bug in <code>texpower</code> 's definition of <code>\set@page@color</code> (used only if <code>pdftex.def</code> doesn't define it) which would cause a fatal er- ror if two <code>\pagecolor</code> com- mands occur. Added command <code>\replacecolor</code> . . . . .	2	v0.0.8c General: The default duplication method will now (only) attack <code>\protected@write</code> . There's a new option <code>hackwrite</code> which restores the former default method (of attacking <code>\write</code> ). Corrected a bug newly intro- duced into <code>\switch</code> with version 0.0.8b. Corrected a bug in the color correction code (spotted by Ross Moore). . . . .
			v0.0.8d General: Corrected a minor quirk in <code>\hidetext</code> . Added a com- mand <code>\addTPcolor</code> for defining new 'standard' colors. In print-

	out versions, the last step will no longer think it's 'first activated'. . . . .	2
v0.0.8e	General: Yet another rewrite of the page duplication code. I hope it's perfect this time :) The options <code>robustduplicates</code> and <code>hackwrite</code> are obsolete now. Thanks to Martin Schroeder for permission to use his <code>everyshi</code> code. Fixed a small quirk in <code>\dstep</code> and <code>\vstep</code> . <code>\darkbackground</code> and relatives now set both page and text color. . . . .	2
v0.0.8f	General: A small change in the definition of <code>\liststepwise</code> to enhance vertical spacing. . . . .	2
v0.0.8g	General: Color management extended and largely rewritten. A small change to make page transition and page duration settings local to groups. <code>\dstep</code> and <code>\vstep</code> now understand the usual optional arguments. . . . .	2
v0.0.9	General: Added support for structured backgrounds (command <code>\backgroundstyle</code> ). New commands for gradient rules and boxes. Added a hack to keep <code>hyperref</code> from producing duplicate page anchors (suggested by Thomas Emmel). Some slight changes in the mode of accounting in <code>\step</code> to hopefully give better results for 'complicated' orders of activating steps using <code>\step</code> 's optional arguments. Added (experimental) commands <code>\multistep</code> and <code>\movie</code> for aiding in (simple) animations. <code>\pagecolor</code> hack removed, as <code>pdftex.def</code> on CTAN now supports <code>\pagecolor</code> . <code>\set@color</code> hack for seminar removed (made unnecessary	
	by enhancements to <code>powersem</code> ). Added rudimentary support for panels. Added rudimentary support for navigation elements. Now put a hyper anchor "firstpage.n" on the first element of the sequence for page n. . . . .	2
v0.0.9a	General: Tidying up the inline documentation. When the color package is loaded before <code>texpower</code> , <code>texpower</code> 's color management is no longer activated automatically. Definitions of standard colors moved to file <code>tpcolors.cfg</code> . Option 'slifonts' is obsolete now. The code is now part of the much more sophisticated package "tpsli-fonts". Now the 'colormath' option cooperates with <code>array.sty</code> (and thus <code>colortbl.sty</code> ). New option "nineminutes" to circumvent a strange behaviour of <code>acrobat/acroread v4.05</code> and later <code>wrt.</code> page duration. Option 'fixcolorstack' now checks also for <code>VTeX</code> . <code>\hidedimmed</code> , <code>\highlightenhanced</code> and <code>\dstep</code> now check for math mode. Now using <code>ifpdf</code> package if available. . . . .	2
v0.0.9b	General: <code>colormath</code> adapted to different handling of 'equation' by <code>amsmath 2.x</code> . Adapted for new version of <code>soul</code> package. Added another patch to <code>colormath</code> for handling <code>array</code> package's "m" columns without color change. 'Turn on' seminar parameters in panel boxes even before <code>\begin{document}</code> . Added a command <code>\overlays</code> , sibling of <code>\multistep</code> , which prints all steps over each other. Added a command <code>\steponce</code> , sibling of <code>\step</code> , which is active only for one step. Removed a	

bug in color correction code introduced in v0.0.8g. . . . .	2	with <code>\shipout</code> created display errors with some packages like pdfscreen. Thanks to Maarten Fokkinga for spotting it. Fixed.	2
v0.0.9c		v0.1a	
General: A small fix to give <code>\overlays</code> a width. <code>\mklength</code> is now a user command. The <code>\@nobreak</code> switch and <code>\everypar</code> are now saved and restored by <code>\stepwise</code> , hopefully enhancing cooperation with section headings and list environments. Changed <code>\newcommand</code> to <code>\providecommand</code> to allow background.sty to be loaded in parallel (thanks to Hans Fr. Nordhaug for the original patch). . . . .	2	General: Color management extended a little to integrate better with LaTeX. Made <code>\step</code> -like commands give better error messages when outside <code>\stepwise</code> . . . . .	2
v0.0.9d		v0.1b	
General: Release. . . . .	2	General: Moved to dtx format. No other code changes. . . . .	2
v0.1		v0.2	
General: Removed font stuff (now resides in tpslifonts). A small fix to avoid warnings about extremely overfull hboxes when measuring steps. Our dabbling		General: Fixed bugs #1029803 and #1073319 reported at SourceForge. Made the handling of whatsits smarter (making write to file and hyperref commands) stepwise-aware. Added option/command to turn on/off the old aggressive/robust filtering. Added fragilesteps environment. . . . .	2

## Index

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

<b>Symbols</b> . . . . .	856, 859	. . .	1269, 1272,
<code>\,</code> . . . . .	707, 711	<code>\@@@movie@TP</code>	3126, 3128
<code>\@@@dblgradrule@TP</code>	993, 996, 1034, 1362, 1793, 3248	<code>\@@button@TP</code> . . . . .	<code>\@hgradrule@TP</code> . . .
<code>\@@@dblvgradrule@TP</code>	. 864, 867, 909, 1331, 1785, 3246	<code>\@dblgradbox@TP</code> .	. 932, 938, 961, 1300, 1777, 3247
<code>\@@@dblgradbox@TP</code> .	. . . . . 1356, 1359	<code>\@@dblgradrule@TP</code> .	<code>\@mk@panel@TP</code> . . . .
<code>\@@@dblgradrule@TP</code>	. . . . . 985, 988	<code>\@@dblgradbox@TP</code> .	. . 1915, 2048, 2062
<code>\@@@dblgradbox@TP</code> .	. . . . . 1325, 1328	<code>\@@dblgradrule@TP</code> .	<code>\@movie@TP</code> . . . . .
<code>\@@@dblvgradrule@TP</code>		<code>\@@dblgradbox@TP</code> .	. . 3122, 3123, 3125
		<code>\@@dblvgradrule@TP</code> .	<code>\@multistep@TP</code> 3074,
		<code>\@@dgradrule@TP</code> . . . . .	3077, 3087, 3129
		<code>\@@dgradbox@TP</code> . . . . .	<code>\@overlays@TP</code> . . . .
		<code>\@@gradrule@TP</code> . . . . .	. . 3145, 3148, 3149
		<code>\@@gradbox@TP</code> . . . . .	<code>\@par</code> . . . . . 649
			<code>\@steponce@TP</code> . . . .

.. 3037, 3040, 3041	\@elt ... 2475, 2478,	1889, 1896,
\@switch@TP .. 2894,	2483, 3226,	2401, 2462, 2906
2896, 2996,	3234, 3238, 3240	\@ignoretrue .. 552, 554
2998, 3012,	\@empty . 650, 3318, 3376	\@iiiparbox ..... 645
3014, 3284, 3305	\@end@tempboxa .... 662	\@linelen ..... 1236
\@vgradrule@TP ...	\@endpbox .....	\@liststepwise@TP .
. 793, 797, 824,	. 599–601, 628–630	.. 3188, 3189, 3192
1269, 1770, 3245	\@firstofone ..... 1929	\@makeoother . 3361, 3367
\@M ... 1947, 1983, 3288	\@gobble .. 317, 382,	\@mk@panel@TP .....
\@acol ..... 640	515, 517, 519,	... 2017, 2025,
\@addtopreamble 589, 620	520, 2224, 2228,	2033, 2041, 2046
\@afterstep@TP ....	2282, 2286,	\@mkpream .... 634, 637
.. 3163, 3329, 3339	2572, 2947, 3243	\@movie@TP . 3117, 3119
\@begin@tempboxa .. 649	\@gtempa ... 2434, 2435	\@multistep@TP .....
\@bstep@TP . 2972, 2973	\@height . 612, 2078,	..... 3062, 3064
\@button@TP . 2098, 2101	2336, 2350,	\@namedef .... 553, 554
\@cclv ..... 2298,	2370, 2797, 3181	\@nb@TPfalse 3221, 3229
2302, 2310,	\@hgradbox@TP 1294, 1297	\@nb@TPtrue . 3222, 3229
2311, 2328,	\@hgradrule@TP 924, 927	\@nextchar .....
2331, 2345,	\@hyper@anchor ...	. 599–601, 628–630
2365, 2380,	.. 2492, 2494, 2496	\@nil ... 588, 2446, 2451
2392, 2527, 2538	\@iden . 2620, 2810, 2816	\@nbreakfalse ... 3334
\@chnum ..... 591, 620	\@ifclassloaded ...	\@nbreaktrue .... 3334
\@classiv ..... 642	... 67, 2052, 2484	\@nomath ..... 531
\@classx ..... 585, 617	\@ifnextchar .. 1036,	\@onlyinstepwise@TP
\@classz .. 585, 617, 641	1096, 2833,	... 2590, 2892,
\@dblhgradbox@TP ..	2836, 2861,	2895, 2956,
..... 1344, 1347	2864, 3025,	2963, 2970,
\@dblhgradrule@TP .	3029, 3062,	2974, 2990,
..... 973, 976	3066, 3117,	2999, 3006,
\@dblvgradbox@TP ..	3120, 3133, 3137	3015, 3022,
..... 1313, 1316	\@ifpackagelater ..	3043, 3098, 3130
\@dblvgradrule@TP .	.. 557, 2632, 2725	\@overlays@TP 3133, 3135
..... 840, 843	\@ifpackageloaded .	\@parboxrestore ... 649
\@defbp . 1047, 1050,	.... 503, 555,	\@parboxto ... 652, 659
1053, 1057, 1084	575, 583, 2153,	\@parstepwise@TP ..
\@defpt . 1107, 1110,	2205, 2216,	.. 3203, 3204, 3207
1113, 1117, 1165	2262, 2273,	\@pboxswfalse ..... 647
\@depth ..... 2797	2317, 2409,	\@pboxswtrue ..... 657
\@dgradrule@TP ....	2459, 2486,	\@replacecolor@TP .
..... 1210, 1213	2508, 2631,	.... 324, 336, 516
\@dintobp 1066–1069, 1090	2723, 2812, 2817	\@replacecolors@TP .
\@dintopt 1130, 1131,	\@ifstar ..... 380,	.... 381, 382, 385
1133, 1134, 1170	1824, 3050,	\@resultcnt@TP 1978,
\@do@s@TP 2873, 2879,	3105, 3169,	1979, 2003, 2004
2893, 2896,	3176, 3187, 3202	\@secondoftwo 1910, 1911
2972, 2992, 3008	\@ifundefined 49, 107,	\@sline ..... 1238
\@dstep@TP .....	333, 1046, 1106,	\@startpbox .....
.. 2992, 2996, 2998	1875, 1882,	. 599–601, 628–630
		\@startvbox ..... 636



1468,	1508,	1646,	1648,	312,	326,	475,	
1514,	1542,	1663,	1692,	1694	482,	487,	492,
1544,	1550,	<code>\bgndrightpanelwidth</code>		493,	512,	513,	
1594,	1597,	.. 1395, 1642, 1688		527,	538,	665,	
1601,	1603,	<code>\bgndrightpanelwidth@TP</code>		669,	681,	688,	
1608,	1611,	... 1380, 1410,		695,	705,	1477,	
1613,	1616,	1472,	1499,	1518,	1560,		
1629,	1650,	1501,	1505,	1569,	1578,		
1730,	1733,	1554,	1558,	1599,	1623,		
1737,	1739,	1639,	1642,	1644,	1660,		
1744,	1747,	1646,	1649,	1669,	1690,		
<code>\bgndbottompanelwidth@TP</code>		1653,	1656,	1714,	1735,		
... 1402, 1514,		1658,	1685,	1751,	1940,		
1550,	1572,	1688,	1692,	1969,	2070,		
1601,	1604,	1695,	1699,	2201,	2227,		
1708,	1737,	1702,	1704,	1707	2256,	2258,	
<code>\bgndbox@TP</code> .....		<code>\bgndsecondgradprogression</code>		2285,	2399,		
... 1473, 1474,		..... 1374, 1388		2403,	2461,		
1755,	1762,	<code>\bgndstripes</code> 1375, 1385		2467,	2485,		
1770,	1777,	<code>\bgndtoppanelheight</code>		2489–2491,			
1785,	1793,	.. 1389, 1576, 1712		2493–2496,			
<code>\bgndfirstgradprogression</code>		<code>\bgndtoppanelheight@TP</code>		2500,	2501,		
..... 1373, 1387		... 1377, 1396,		2503–2506,			
<code>\bgndgradmidpoint</code> .		1466,	1482,	2512,	2513,		
..... 1376, 1386		1487,	1529,	2515–2518,			
<code>\bgndleftpanelheight@TP</code>		1535,	1542,	2520,	2537,		
..... 1404,		1573,	1576,	2560,	2582,		
1496,	1500,	1580,	1582,	2658,	2664,		
1524,	1542,	1587,	1590,	2667,	2675,		
1615,	1617,	1592,	1616,	2698,	2709,		
1625,	1627,	1709,	1712,	2720,	2727,		
1662,	1671,	1716,	1718,	2783,	2805,		
<code>\bgndleftpanelwidth</code>		1723,	1726,	1728	2819,	2824,	
.. 1393, 1621, 1667		<code>\bgndtoppanelwidth@TP</code>		2913,	2941,		
<code>\bgndleftpanelwidth@TP</code>		..... 1398,		2948,	2981,		
... 1379, 1406,		1487,	1499,	2994,	3010,		
1470,	1492,	1535,	1541,	3041,	3082,		
1496,	1499,	1571,	1580,	3155, 3306, 3310			
1520,	1524,	1583,	1706,	<code>\bottompanelcontents@TP</code>			
1618,	1621,	1708,	1716,	1719	..... 1596,		
1625,	1628,	<code>\bgroup</code> ... 579, 590, 640		1602,	1732,		
1632,	1635,	<code>\blackbackground</code> ..		1738,	1821,		
1637,	1664,	..... 488, 492		1844,	1882,		
1667,	1671,	<code>\blacktriangleleft</code> 2155		1884, 2023, 2025			
1674,	1678,	<code>\blacktriangleright</code>		<code>\bottompanelheight</code> .			
1681,	1683,	..... 2156		... 1603, 1608,			
1707,	1720,	1741	<code>\boldmath</code> ... 2155–	1613,	1739,		
<code>\bgndrightpanelheight@TP</code>		2157,	2160–2162	1744,	1749,		
... 1408, 1505,		<code>\boolean</code> .....		43,	1812, 1905, 2025		
1558,	1617,	46,	69,	97,	98,	<code>\bottompanelheight@TP</code>	

..... 1846, 1886	2164, 2184,	\calcmxbadness@TP .
\bottompanelshift .	2189, 2194, 2199	..... 1954, 1976
..... 1605,	\buttonrule .....	\calcrightdimen@TP 1907
1741, 1813, 2375	... 2086, 2111,	\calctopdimen@TP . 1904
\bottompanelshift@TP	2114, 2117,	\calcvdimen@TP ....
..... 1847, 1887	2127–2129, 2137	... 1580, 1601,
\bottompanelwidth .	\buttonsep 2085, 2111,	1716, 1737,
... 1604, 1740,	2114, 2117,	1904, 1905, 1913
1811, 1905, 2025	2127–2129, 2138	\carg ..... 680
\bottompanelwidth@TP	\buttonshadowshift	\catcode 3365, 3368, 3370
..... 1845, 1885	..... 2087, 2121	\char .. 2642, 2772, 2800
\box .... 1282, 2311,	\buttonshadowvshift	\cl@ckpt ..... 3227
2392, 2527,	.. 2088, 2122, 2131	\cl@ckpt@TP .....
2774, 2792, 2801	\buttonssymbolsize .	.. 3235, 3238, 3240
\boxedsteps .....	.. 2152, 2155–	\cl@ckptpause@TP .
.. 2680, 2973, 3211	2157, 2160–2162	.. 2475, 2479, 2483
\brack@movie@TP ...		\cleaders .. 2764, 2797
.. 3117, 3120, 3122	<b>C</b>	\clearcolorcorrections@TP
\brack@multistep@TP	\c@dgradhslope@TP 1229	..... 2412, 2414
.. 3062, 3068, 3071	\c@dgradvslope@TP 1229	\clipbox ..... 1036,
\brack@overlays@TP .	\c@firststep .. 3168,	1096, 1177, 1192
.. 3133, 3139, 3142	3171, 3303, 3346	\clipbox@ .... 1036,
\brack@steponce@TP .	\c@lor@namefile ... 107	1037, 1096, 1097
.. 3025, 3031, 3034	\c@ofs@TP . 3168, 3346	\closeout ..... 3354
\bstep .. 2970, 2979,	\c@step ..... 2916,	\code . 671, 675, 678–680
3261, 3278–3280	3294, 3296–	\codeswitch ... 672, 676
\bt@depth@TP .....	3298, 3303,	\col@sep ..... 579
.. 2103, 2115–	3314, 3316–	\color .... 99, 329,
2117, 2127,	3318, 3321, 3343	333, 342, 343,
2129, 2143, 2147	\c@stepcommand ....	395, 495, 499,
\bt@height@TP . 2097,	... 2900, 2909,	502, 506, 512,
2112–2114,	2916, 2925,	531, 542, 549,
2129, 2143, 2147	2960, 2967,	551, 553, 561,
\bt@width@TP .. 2091,	2978, 3003,	569, 573, 579,
2109–2111,	3019, 3218,	633, 640, 657,
2128, 2141, 2148	3283, 3291, 3325	664, 672, 2072,
\bullet ..... 574	\c@stepsperformed .	2748, 2757,
\button ..... 2089,	... 2916, 2927,	2759, 2764,
2168, 2173,	3304, 3313, 3340	2766, 2773,
2178, 2183,	\c@totalsteps . 3291,	2775, 2777,
2188, 2193, 2198	3296, 3307,	2984, 2987, 3012
\button@TP . 2092, 2095	3313, 3316, 3340	\color@begingroup .
\buttonbackarrowsymbol	\calcbottomdimen@TP	..... 1041, 1101
.. 2157, 2162, 2179	..... 1905	\color@cmyk ..... 124
\buttonleftarrowsymbol	\calchdimen@TP ....	\color@endgroup ...
... 2155, 2160,	... 1625, 1646,	..... 1043, 1103
2169, 2174, 2199	1671, 1692,	\color@gray ..... 109
\buttonrightarrowsymbol	1906, 1907, 1918	\color@rgb ... 110, 119
... 2156, 2161,	\calcleftdimen@TP 1906	\colorbetween . 188,
		258, 437, 454,

811, 813, 882,	1828, 1829,	<code>\DeclareOption</code> . . . .
884, 895, 897,	1835–1838,	. . . 5, 7, 8, 10,
949, 951, 1008,	1844–1847,	12, 16, 19, 22,
1010, 1021,	1853–1856,	25, 29, 32, 35, 38
1023, 1235,	1862–1865,	<code>\DeclarePanel</code> . . . . <u>1824</u>
1245, 1798, 1804	1870, 1877–	<code>\declarepanel@TP</code> . .
<code>\colorbox</code> . . . . . 2706	1880, 1884–	. . 1824, 1826, 1870
<code>\colorcorrections@TP</code>	1887, 1891–	<code>\DeclareRobustCommand</code>
. . . 2395, 2418,	1894, 1898–	. . . . . 529,
2440, 2441,	1901, 2424,	633, 671, 672,
2446, 2447,	2478, 2900,	675, 676, 683,
2451, 2454,	2925, 3226,	686, 690, 693,
2456, 2549, 2550	3234, 3298, 3318	697, 700, 2633,
<code>\colors@TP</code> . . . . .	<code>\CT@column@color</code> . . 606	2689, 2734, 2785
. 350, 359, 362,	<code>\CT@do@color</code> . . . . . 608	<code>\def</code> . . 101, 139, 148,
363, 368, 388,	<code>\CT@extract</code> . . . . . 588	159, 165, 174,
427, 444, 461, 466	<code>\CT@row@color</code> . . . . . 607	179, 264, 273,
<code>\columncolor</code> . . . . . 588	<code>\CT@setup</code> . . . . . 605	275, 276, 353,
<code>\commandapp</code> . . . . . <u>679</u>	<code>\current@color</code> . . . .	359, 368, 387,
<code>\commitcolor@TP</code> 2659,	. . 499, 2424, 2429	406, 419, 443,
2662, 2669, 2826	<code>\current@step@TP</code> . 2928	460, 465, 498,
<code>\commitcolors@TP</code> . .	<code>\currentpagevalue</code> .	506, 549–552,
. . 2983, 2987, 2996	. . . 2165, 2168,	561, 562, 569,
<code>\complement@three@TP</code>	2188, 2193, 2320	570, 577, 585,
. 264, 283, 294, 304	<code>\currentvariant@TP</code> .	617, 634, 640,
<code>\complement@TP</code> <u>259</u> ,	. . . . . 411, 415, 424	645, 652, 664,
266, 268, 270, 288		831, 966, 1036,
<code>\complementcolor</code> . . <u>278</u>		1037, 1084,
<code>\concept</code> . . . . . <u>688</u>		1090, 1096,
<code>\convert@cmyk@rgb@TP</code>		1097, 1165,
. . . . . <u>165</u> , 200, 233	<code>\dollarbegin</code> . . . . .	1170, 1200,
<code>\convert@cmykvalue@rgbvalue@TP</code>	. 579, 593, 596,	1306, 1337,
. 159, 167, 169, 171	597, 622, 625, 626	1436, 1438,
<code>\convert@RGB@rgb@TP</code>	<code>\dollarend</code> . . . 580,	1441, 1584,
. 179, 205, 239, 299	595–597, 624–626	1605, 1675,
<code>\convert@RGBvalue@rgbvalue@TP</code>	<code>\darkbackground</code> 483, 487	1696, 1927,
. 174, 181, 183, 185	<code>\dblgradbox</code> . . . . . <u>1333</u>	1936, 2395,
<code>\copy</code> . . . 2339, 2524,	<code>\dblgradbox@TP</code> . . .	2411, 2421,
2529, 2538,	. . . . . 1338, 1341	2428, 2429,
2542, 2556, 2564	<code>\dblgradrule</code> . . . . . <u>962</u>	<code>\dblgradrule@TP</code> . .
<code>\count</code> . . . . . 2531, 2558	. . . . . 967, 970	2432, 2437,
<code>\count@</code> . . . . .	<code>\dblvgradbox</code> . . . . . <u>1302</u>	2444, 2451,
586, 618, 3362–3365	<code>\dblvgradbox@TP</code> . . .	2475, 2476,
<code>\count@em@TP</code> 3216, 3284	. . . . . 1307, 1310	2478, 2483,
<code>\csname</code> 51, 104, 105,	<code>\dblvgradrule</code> . . . . . <u>825</u>	2488, 2499,
252, 254, 307,	<code>\dblvgradrule@TP</code> . .	2511, 2521,
329, 342–345,	. . . . . 832, 835	2636, 2637,
406, 412, 547,	<code>\deactivate@inner@TP</code>	2738, 2742,
548, 660, 1083,	. . . . . 2881,	2745, 2752,
1164, 1383,	2896, 2992, 3008	2762, 2768,
		2788, 2792,



2796,	2798,	1453, 1458, 1463	88, 91, 93, 104,
2842,	2844,	<code>\defineTPcolor</code> 371, 518	132, 137, 142,
2846,	2848,	<code>\depth</code> . . . . . 2705, 2716	144, 146, 151,
2853,	2855,	<code>\depthof</code> . . . . . 2116	153, 155, 157,
2870,	2872,	<code>\dgradrule</code> . . . . . 1201	163, 168, 170,
2876,	2878,	<code>\dgradrule@TP</code> 1204, 1207	172, 177, 182,
3034,	3036,	<code>\dgradslope</code> . 1194, 1203	184, 186, 201,
3040,	3071,	<code>\dimcolor</code> . 430, 443, 521	206, 216, 222,
3073,	3077,	<code>\dimcolors</code> . . . . .	231, 246, 255,
3103,	3104,	441, 522, 2668, 2996	262, 267, 269,
3122,	3123,	<code>\dimen</code> . . . . . 2532, 2559	271, 284, 289,
3142,	3144,	<code>\dimen@ii</code> .. 2756, 2757	295, 305, 308,
3148,	3188,	<code>\dimexpr</code> . . . 1091, 1171	323, 338, 1087,
3189,	3203,	<code>\dimlevel</code> . 429, 430, 441	1167, 1433,
3204,	3221,	<code>\disable@counting@TP</code>	1957, 1963,
3222,	3224,	. . . . . 3241, 3289	1964, 2133,
3226,	3232,	<code>\discretionary</code> . . . .	2134, 2275, 2288
3234,	3238,	. . 2639, 2770, 2798	<code>\egroup</code> . . . . . 603
3240,	3274,	<code>\displayboxed</code> 2610, 2681	<code>\eject</code> . . . . . 2528
3349,	3359,	<code>\displayidentical</code> .	<code>\else</code> . . . . 53, 55, 57,
3372,	3386,	. . . 2620, 2673,	111, 115, 119,
<code>\default@color</code> . . . 499		2684, 2686,	124, 195, 198,
<code>\define@key</code> . . . 1415,		2830, 2883, 2884	203, 208, 223,
1418,	1420,	<code>\displaystepcontents</code>	232, 247, 650,
1423,	1425,	. . 2681, 2684, 2889	655–657, 1089,
1428,	1430,	<code>\do</code> . . . . . 3361, 3363	1169, 1191,
1433,	1436,	<code>\do@colorcorrections@TP</code>	1566, 1577,
1438,	1441,	. . . . . 2546, 2555	1598, 1622,
1443,	1446,	<code>\do@insert@firstanchor@TP</code>	1643, 1668,
1448,	1451,	. . . . . 2319,	1689, 1713,
1453,	1456,	2322, 2324, 2387	1734, 1972,
1458,	1461,	<code>\dont@insert@firstanchor@TP</code>	2007, 2016,
1463,	1466,	. . . . . 2323, 2385	2024, 2032,
1468,	1470,	1472	2040, 2304,
<code>\definecolor</code> 100, 101,		<code>\dospecials</code> . . . . . 3361	2386, 2417,
112, 113, 116,		<code>\dp</code> 1048, 1054, 1067,	2455, 2614,
117, 120–122,		1069, 1114,	2660, 2693,
125–127, 256,		1134, 1279,	2847, 2902,
309, 375, 376,		1281, 1916, 1950	2924, 2984,
784, 792, 847,		<code>\dstep</code> . . 2990, 3004,	3229, 3334,
855, 863, 923,		3263, 3278–3280	3375, 3377, 3389
931, 978, 984,		<code>\dstripewd@TP</code> . . . . .	<code>\em</code> . . . . . 529, 531
992, 1209, 1215,		. . . 1219, 1227,	<code>\emph</code> . . . . . 527
1262,	1268,	1228, 1230,	<code>\empty</code> . . . 362, 1377–
1293,	1299,	1236, 1237, 1246	1381, 1564,
1318,	1324,	<code>\dumpcolorset</code> . . . . .	1575, 1596,
1330,	1349,	. . . 398, 520, 3256	1620, 1641,
1355,	1361,	<code>\dur@ms@TP</code> . 3103, 3126	1666, 1687,
1420,	1425,		1711, 1732,
1430,	1448,	<b>E</b>	1875, 1882,
		<code>\edef</code> . . . . . 87,	



602, 631, 653,	1847, 1853–	1785, 1793,
658, 661, 1093,	1856, 1862–	1915, 1943,
1173, 1193,	1865, 1875,	2068, 2329,
1568, 1589,	1877–1880,	2339, 2355,
1610, 1634,	1882, 1884–	2357, 2373,
1655, 1680,	1887, 1889,	2375, 2538,
1701, 1725,	1891–1894,	2542, 2755,
1746, 1974,	1896, 1898–	2764, 2772,
2005, 2010,	1901, 1948,	2797, 2800, 3288
2011, 2018,	1979, 2004,	<code>\height</code> . . . . 2705, 2716
2026, 2034,	2298, 2316,	<code>\heightof</code> . . 2113, 2743
2042, 2306,	2385, 2387,	<code>\hfil</code> . . . . . 621, 624–626
2388, 2419,	2391, 2478,	<code>\hfill</code> . . . 596, 597, 2356
2457, 2616,	2524, 2527,	<code>\hfuzz</code> . 1947, 1983, 3288
2660, 2695,	2529–2532,	<code>\hgradbox</code> . . . . . <a href="#">1284</a>
2849, 2904,	2538, 2542,	<code>\hgradbox@TP</code> 1288, 1291
2926, 2984,	2556–2559,	<code>\hgradrule</code> . . . . . <a href="#">910</a>
3229, 3334,	2570–2572,	<code>\hgradrule@TP</code> . 916, 919
3380, 3382, 3391	2574, 2575,	<code>\hidedimmed</code> . 2665, 2673
<code>\filterpage@TP</code> 2327,	2909, 2925,	<code>\hideignore</code> . 2627, 2684
2535, 2570, 2575	2927, 2960,	<code>\hidephantom</code> . . . . .
<code>\filterwhatsits@TP</code> .	2967, 2978,	. . . 2621, 2650,
. . . . . 2536, 2570	3003, 3019,	2655, 2678, 2681
<code>\first@TP@</code> 2899, 2923,	3168, 3218,	<code>\hidesmartignore</code> . 2628
2925, 3293,	3221, 3222,	<code>\hidestepcontents</code> .
3298, 3314, 3318	3226, 3230,	. . 2681, 2684, 2890
<code>\firstcol@TP</code> . . 252, 253	3234, 3335, 3346	<code>\hidetext</code> 2633, 2649,
<code>\firstgradprogression@TP</code>	<code>\gobackbutton</code> . . . . 2176	2650, 2654, 2655
. . . . 739, 746,	<code>\grabfourth@TP</code> <a href="#">273</a> , 293	<code>\hidevanish</code> . 2676, 2678
773, 827, 912,	<code>\gradmidpoint@TP</code> 745,	<code>\highlightboxed</code> . . 2689
964, 1253, 1304,	830, 831, 879,	<code>\highlightboxedsep</code> . .
1335, 1373, 1436	966, 1005, 1306,	. . . 2687, 2688,
<code>\font</code> . . 2642, 2772, 2800	1337, 1376, 1433	2702, 2704,
<code>\footins</code> . . . . 2529–		2713, 2715,
2532, 2556–2559		2741, 2743,
<code>\footnotesize</code> . . . . .	<b>H</b>	2748, 2759,
. . . 707, 711, 2152	<code>\h</code> 1057, 1060, 1117, 1123	2766, 2775, 2777
<code>\fullscreenbutton</code> 2196	<code>\hb@xt@</code> . . . . 2002, 2353	<code>\highlightenhanced</code> .
	<code>\hbadness</code> . . . . .	. . . . . 2821, 2830
<b>G</b>	<code>\hbox</code> . 579, 590, 640,	<code>\highlighttext</code> . . . .
<code>\gdef</code> 1828, 2314, 2418,	796, 1040, 1045,	. . . 2734, 2785,
2454, 2456,	1100, 1105,	2809, 2810,
2550, 2928,	1181, 1184,	2813, 2815, 2816
3126, 3129,	1187, 1192,	<code>\hoffset</code> 2339, 2355,
3163, 3371, 3385	1238, 1276,	2358, 2373, 2375
<code>\global</code> . . . . . 1755,	1485, 1490,	<code>\hpagecolor</code> . . . . . 1795
1762, 1770,	1507, 1511,	<code>\hpanelsvalue@TP</code> . .
1777, 1785,	1532, 1538,	. . . . . 1381,
1793, 1835–	1547, 1762,	1441, 1564, 1567
1838, 1844–	1770, 1777,	

<code>\hrule</code> .. 2078, 2336,	<code>\ifcase</code> ... 56, 591, 620	1508,	1518,
2350, 2370, 3181	<code>\ifcolorexists@TP</code> .	1520,	1529,
<code>\hsize</code> ..... 649	. 332, 339, 435, 452	1544,	1554,
<code>\hskip</code> 592, 595, 625,	<code>\iffalse</code> ..... 3221	1560,	1569,
626, 2538, 2542,	<code>\ifFileExists</code> ..... 50	1573,	1578,
2757, 2765,	<code>\ifhbox</code> ..... 2001	1590,	1594,
2773, 2797, 2801	<code>\ifmeasuring@</code> 2906, 2924	1599,	1611,
<code>\hspace</code> ..... 2121	<code>\ifmmode</code> . 656, 2612,	1618,	1623,
<code>\hss</code> 659, 1078, 1142, 1238	2660, 2691, 2984	1635,	1639,
<code>\hstripe@TP</code> .....	<code>\ifnormalvariant@TP</code>	1644,	1656,
. 795, 816, 818,	. 414, 421, 432, 449	1660,	1664,
885, 891, 900, 902	<code>\ifnum</code> ..... 2003,	1669,	1681,
<code>\ht</code> . 610, 1055, 1069,	3296, 3316, 3340	1685,	1690,
1108,	<code>\ifpdf</code> .... 63, 66, 1176	1702,	1709,
1131, 1134,	<code>\ifshippingduplicate</code>	1714,	1726,
1281, 1916,	.. 2384, 2416, 2470	1730,	1735,
1950, 2331,	<code>\ifthenelse</code> .. 43, 46,	1747,	1751,
2345, 2365, 2380	69, 97, 98, 134,	1797,	1803,
<code>\Hy@colorlink</code> .... 1929	136, 162, 312,	1829,	1951,
<code>\Hy@endcolorlink</code> . 1930	323, 326, 338,	1955,	1961,
<code>\hyper@@anchor</code> 2316,	355, 374, 415,	1967,	1968,
2382, 2488,	475, 482, 487,	2070,	2109,
2571, 2572, 2574	492, 493, 512,	2112,	2115,
<code>\hyper@anchor</code> .... 2487	513, 527, 538,	2201,	2227,
<code>\hyper@anchor@TP</code> . 2487	665, 669, 679,	2256,	2258,
<code>\hyper@anchorstart</code> .	681, 688, 695,	2285,	2399,
..... 2498, 2499	705, 754, 756,	2403,	2461,
<code>\hyper@anchorstart@TP</code>	764, 774, 782,	2467,	2485,
..... 2498,	790, 799, 815,	2489–2496,	
2502, 2504, 2506	823, 829, 837,	2500–2506,	
<code>\hyperlink</code> ... 1910,	845, 853, 861,	2512–2518,	
2168, 2188, 2193	869, 890, 899,	2520,	2537,
<code>\hypersetup</code> ... 2205,	905, 908, 913,	2560,	2582,
2207, 2216,	921, 929, 940,	2658,	2664,
2218, 2219,	953, 960, 966,	2667,	2675,
2262, 2264,	972, 978, 984,	2698,	2709,
2273, 2275, 2277	990, 998, 1016,	2720,	2727,
<code>\hypertarget</code> ..... 2320	1025, 1031,	2783,	2805,
<code>\hyphenchar</code> .....	1033, 1203,	2819,	2824,
.. 2642, 2772, 2800	1209, 1215,	2846,	2848,
	1254, 1262,	2872,	2913,
	1268, 1287,	2941,	2948,
<b>I</b>	1293, 1299,	2981,	2994,
<code>\if</code> ..... 654, 655	1306, 1312,	3010,	3093,
<code>\if@first@TP@true</code> .	1318, 1324,	3155, 3306, 3310	
..... 2866,	1330, 1337,	<code>\iftrue</code> ..... 3222	
2867, 2897, 2922	1343, 1349,	<code>\ifvbox</code> ..... 2008	
<code>\if@nb@TP</code> .....	1355, 1361,	<code>\ifvoid</code> ..... 2302	
.. 3221, 3222, 3334	1477, 1482,	<code>\ifx</code> . 52, 54, 109, 110,	
<code>\if@nobreak</code> ..... 3229	1492, 1501,	119, 124, 193,	
<code>\if@pboxsw</code> ..... 661			

196, 199, 204,	197, 201, 206, 245	<code>\leftpanelheight</code> ..
209, 220, 229,	<code>\interpolate@TP</code> ...	... 1627, 1673,
244, 650, 1083,	.... <u>129</u> , 141,	1815, 1906, 2033
1164, 1564,	143, 145, 150,	<code>\leftpanelheight@TP</code>
1575, 1596,	152, 154, 156, 221	..... 1855, 1893
1620, 1641,	<b>J</b>	<code>\leftpanelraise</code> 1629,
1666, 1687,	<code>\jobname</code> ..... 3349	1675, 1816, 2355
1711, 1732,	<b>K</b>	<code>\leftpanelraise@TP</code> .
1970, 2015,	<code>\kern</code> ..... 592,	..... 1856, 1894
2023, 2031,	597, 818, 886,	<code>\leftpanelwidth</code> ...
2039, 2453,	902, 956, 1012,	... 1628, 1632,
2845, 2900,	1028, 1181,	1637, 1674,
3373, 3376, 3387	1186–1188,	1678, 1683,
<code>\immediate</code> ... 3350,	1276, 2067,	1814, 1906, 2033
3354, 3379, 3390	2073, 2080,	<code>\leftpanelwidth@TP</code> .
<code>\inactive</code> ..... <u>695</u>	2082, 2337–	..... 1854, 1892
<code>\infinitepageduration</code>	2339, 2341,	<code>\lengthtest</code> 134, 136,
..... 2254,	2342, 2351,	162, 1482, 1492,
2257, 2275, 2288	2352, 2355,	1501, 1508,
<code>\initpanels@TP</code> ....	2358, 2361,	1520, 1529,
... 1562, 1752,	2362, 2371–	1544, 1554,
1761, 1769,	2373, 2375–2377	1951, 1968,
1776, 1784, 1792	<b>L</b>	2109, 2112, 2115
<code>\inner@display@TP</code> .	<code>\labelitemi</code> ..... 574	<code>\let</code> ..... 70, 96, 99,
..... 2883,	<code>\large</code> ..... 2054	100, 251, 317,
2889, 2942, 2943	<code>\lastbox</code> ..... 1948,	329, 342, 344,
<code>\inner@hide@TP</code> 2884,	1981, 1985, 2000	362, 381, 382,
2890, 2949, 2950	<code>\lastpenalty</code> ..... 1991	497, 499, 505,
<code>\input</code> ... 107, 470, 3355	<code>\laststep@ms@TP</code> ...	515, 517, 519,
<code>\InputIfFileExists</code> .	.. 3052, 3085, 3107	520, 543–548,
..... 40, 3403	<code>\leaders</code> ... 2538, 2542	559, 560, 567,
<code>\insert@column</code> ....	<code>\leavevmode</code> .....	568, 579, 580,
. 594, 596, 597,	. 578, 640, 646,	636, 640–642,
599–601, 623,	707, 711, 1045,	659, 663, 667,
625, 626, 628–630	1105, 1183,	773, 827, 828,
<code>\insert@firstanchor@TP</code>	1192, 1274,	830, 912, 964–
..... 2324,	2118, 2624,	966, 1253, 1286,
2383, 2385, 2387	2634, 2736, 3152	1304–1306,
<code>\insertdup@TP</code> ....	<code>\left</code> ..... 680	1335–1337,
.. 3323, 3331, 3342	<code>\leftarrow</code> ..... 2162	1373, 1374,
<code>\insertfilterwhatsits@TP</code>	<code>\leftpanelcontents@TP</code>	1376–1381,
.. 2539, 2541, 2563	..... 1620,	1571, 1572,
<code>\insertfirstduplicate@TP</code>	1626, 1666,	1582, 1583,
.. 2544, 2581, 3323	1672, 1822,	1592, 1603,
<code>\insertsecondduplicate@TP</code>	1853, 1889,	1604, 1613,
.. 2553, 2586, 3342	1891, 2031, 2033	1617, 1627–
<code>\interpolate@four@TP</code>		1629, 1637,
..... <u>148</u> , 230		1648–1650,
<code>\interpolate@three@TP</code>		1658, 1662,
.... <u>139</u> , 194,		1663, 1673,



2111, 2113,	\newsavebox . . . . . 714	\o@endeqastar@TP . . . . . 548, 554
2114, 2116, 2117	\newtoks . . . 2474, 3223	. . . . . 559, 561, 567, 569
\mkpanels@TP . . . . .	\newwrite . . . . . 3357	\o@eq@TP . . . . .
. . . 1475, 1561,	\next 2275, 2276, 2288,	. 559, 561, 567, 569
1762, 1770,	3374, 3381,	\o@eqa@TP . . . . 545, 551
1777, 1785, 1793	3383, 3388, 3392	\o@eqastar@TP . 547, 553
\mkpbox@TP 708, 712, 726	\nextfullpagebutton	\o@fboxrule@TP . . . . .
\monochromeinactive	. . . . . 2191	. . . . . 2133, 2145
. . . . . 700, 703	\nextpagebutton . . 2186	\o@fboxsep@TP 2134, 2145
\moveleft . . . . 2339,	\nextstepbutton . . 2181	\o@hyper@@anchor . . . . .
2353, 2373, 2375	\noexpand . 201, 206,	. . . . . 2316, 2382
\movie . . . . . 3048,	256, 309, 2275,	\o@hyper@@anchor@TP
3094, 3098, 3269	2288, 2441, 3399	. . . . . 2571, 2574
\movie@TP . . . . .	\noindent . . . . 719, 726	\o@reset@color@TP . . . . .
. . 3108, 3112, 3116	\nonboxedsteps . . . . .	. . . 2427, 2440,
\multistep . . . 2598,	. . . . . 2683, 2685	2441, 2448, 2453
3043, 3128, 3268	\nonnormalwarnings@TP	\o@rlap@TP . 2791, 2792
\multistep@TP . . . . .	. . . . . 400, 417	\o@set@color@TP . . . . .
. . 3053, 3057, 3061	\normalcolor . . . . .	. . . . . 2421, 2439
	. . . . 495, 498, 506	\o@shipout@TP 2292, 2311
<b>N</b>	\normalfont . . . . . 2076	\o@text@TP . . . 505, 506
\newboolean . 4, 9, 11,	\normalsize . . . . . 2054	\o@textnormal@TP . . . . .
15, 17, 18, 21,	\normalstep@ms@TP . . . . .	. . . . . 497, 498
24, 27, 31, 34,	. . . 3047, 3056,	\o@textsuperscript@TP
37, 47, 82, 1412,	3079, 3102, 3111	. . . . . 663, 664
1414, 2608,	\nosteps@ms@TP . . . . .	\offinterlineskip . . . . .
2609, 3250, 3251	. . . . . 3126, 3129	. . . . 805, 807,
\newbox . . 80, 1473, 2472	\NOT . . . . 2490, 2491,	874, 876, 1181,
\newcolordef@TP . . . . .	2495, 2500,	1187, 1479,
. . . . 142, 144,	2501, 2505,	1481, 1526,
146, 151, 153,	2512, 2513, 2517	1528, 2066,
155, 157, 168,	\not . . . . . 1955, 1967	2335, 2349, 2369
170, 172, 182,	\ns@ms@TP . . . . .	\oldfilteringoff . . 14
184, 186, 201,	. . 3047, 3095, 3102	\oldfilteringon . . . 13
202, 206, 207,	\nshook@ms@TP . . . . .	\openout . . . . . 3350
216, 222, 231,	. . 3048, 3095, 3103	\optarg@ms@TP . 3073,
234, 240, 246,	\null . . . . . 1474, 1755	3077, 3081,
256, 267, 269,	\number . . . . 1963, 2320	3086, 3122, 3123
271, 284, 289,		\optarg@ov@TP . 3144,
295, 300, 305, 309	<b>O</b>	3148, 3155, 3158
\newcounter . . . . .	\o@afterstep@TP . . . . .	\optarg@so@TP . . . . .
. . . 74, 75, 735,	. . . . . 2936, 2953	. . 3036, 3040, 3041
737, 743, 1196,	\o@definecolor@TP . . . . .	\optwidth@TP . . . . .
1198, 2603–	. . . . . 100, 103	. . 1920, 1932, 1975
2607, 3042, 3165	\o@dm@TP . . . . . 543, 549	\optwidthdisablecommands@TP
\newif . . . 51, 2470, 2906	\o@enddm@TP . . . 544, 550	. . . . . 1927, 1945
\newinsert . . . . . 2473	\o@endeq@TP . . . . .	\optwidthlinetolerance
\newlength . . . . 77,	. 560, 562, 568, 570	. . 1925, 1955, 1967
78, 2687, 2731–2733	\o@endeqa@TP . . 546, 552	
\newpage . . . . . 2471		

<code>\optwidthsteps</code> . . . . .	<code>\pageTransitionGlitter</code>	<code>\pdfxform</code> . . . . .
. . . 1924, 1940,	. . . . . 2251	<code>\phantom</code> 2624, 2636, 2642
1946, 1969, 1973	<code>\pageTransitionReplace</code>	<code>\phantomrule@TP</code> . . .
<code>\or</code> 595–597, 599, 600,	. . . . . 2253	. . . . . 1753, 1762
624–626, 628, 629	<code>\pageTransitionSplitHI</code>	<code>\pickup@s@optargs@TP</code>
<code>\orig@mathchoice@TP</code>	. . . . . 2231	. . . . . 2831,
. . . . . 3273, 3276	<code>\pageTransitionSplitHO</code>	2893, 2896,
<code>\origmath</code> . 573, 574,	. . . . . 2229	2972, 2992, 3008
667, 680, 2155–	<code>\pageTransitionSplitVI</code>	<code>\prepnex@tok</code> . . . . .
2157, 2160–2162	. . . . . 2235	. 587, 614, 619, 631
<code>\output</code> 2526, 2527, 2533	<code>\pageTransitionSplitVO</code>	<code>\present</code> . . . . . 705, 721
<code>\outputduplicate@TP</code>	. . . . . 2233	<code>\presentbox</code> . . . . . 708
. . 2567, 2584, 3341	<code>\pageTransitionWipe</code>	<code>presentbox</code> (environ-
<code>\overlays</code> . . . . .	. . . . . 2246	ment) . . . . . 714
. . 2598, 3130, 3270	<code>\panel@sanitize@TP</code> .	<code>\processcolor</code> . . . . .
	. . . . . 1908, 1915	. 190, 253, 280, 307
	<code>\panelalignment</code> 1921,	<code>\processcolor@TP@cmyk</code>
	2054, 2057, 2077	. 199, 227, 229, 291
	<code>\panelmargin</code> . . . . .	<code>\processcolor@TP@gray</code>
	. . 1807, 1920–	. 196, 218, 220, 286
	1922, 2048,	<code>\processcolor@TP@hsb</code>
	2067, 2073,	. 209, 242, 244, 302
	2074, 2080, 2082	<code>\processcolor@TP@RGB</code>
	<code>\paperheight</code> . . . . . 84, 88	. . . . . 204, 237, 297
	<code>\paperwidth</code> . . . . . 83, 87	<code>\processcolor@TP@rgb</code>
	<code>\par</code> . . . . . 717, 728, 2523	. 191, 193, 224,
	<code>\par@stepcapsule</code> . .	234, 240, 281, 300
	. . . . . 3184, 3212	<code>\processme@TP</code> . . . . .
	<code>\parbox</code> 1944, 1946, 2074	. . . . . 353, 359,
	<code>\paren@movie@TP</code> . . .	363, 368, 387,
	. . . . . 3120, 3123	419, 443, 460, 465
	<code>\paren@multistep@TP</code>	<code>\ProcessOptions</code> . . . 41
	. . . . . 3067, 3077	<code>\proper@bstep@TP</code> . .
	<code>\paren@overlays@TP</code> .	. . 2972, 2979, 3261
	. . . . . 3138, 3148	<code>\proper@dstep@TP</code> . .
	<code>\paren@steponce@TP</code> .	. . 2992, 3004, 3263
	. . . . . 3030, 3040	<code>\proper@movie@TP</code> . .
	<code>\parskip</code> . . . . . 3181	. . . . . 3100, 3269
	<code>\parstepwise</code> . . . . .	<code>\proper@multistep@TP</code>
	. . 3184, 3200, 3213	. . . . . 3045, 3268
	<code>\pause</code> . . . . . 2578	<code>\proper@overlays@TP</code>
	<code>\pausesafecounter</code> .	. . . . . 3132, 3270
	. . . . . 2482, 2484	<code>\proper@rstep@TP</code> .
	<code>\pbox@TP</code> . . 714, 718, 726	. . . . . 2976, 3262
	<code>\pdfastxform</code> . . . . . 1187	<code>\proper@redstep@TP</code> .
	<code>\pdfliteral</code> . . . . .	. . . . . 3001, 3264
	. . 1058, 1064, 1075	<code>\proper@rstep@TP</code> .
	<code>\pdfoutput</code> . . 52, 54, 56	. . . . . 2958, 3260
	<code>\pdfrefxform</code> . . . . . 1187	<code>\proper@reswitch@TP</code>
	<code>\pdftrue</code> . . . . . 58	. . . . . 2965, 3272

## P

<code>\PackageError</code> . . . . .		
. . . 210, 248, 2594		
<code>\PackageInfo</code> . . . . .		
. . 44, 45, 2648,		
2653, 2808,		
2814, 2915, 3307		
<code>\PackageWarning</code> . . .		
. . . . 423, 2210,		
2222, 2267,		
2280, 2405, 2464		
<code>\pagecolor</code> 396, 502, 512		
<code>\pageDuration</code> . . . . .		
. . . 2260, 2270,		
2277, 2282,		
2286, 2288, 3103		
<code>\pagetransition</code> . . .		
. . . 2201, 2229,		
2231, 2233,		
2235, 2237,		
2239, 2241,		
2243, 2246,		
2248, 2251, 2253		
<code>\pageTransitionBlindsH</code>		
. . . . . 2237		
<code>\pageTransitionBlindsV</code>		
. . . . . 2239		
<code>\pageTransitionBoxI</code>		
. . . . . 2243		
<code>\pageTransitionBoxO</code>		
. . . . . 2241		
<code>\pageTransitionDissolve</code>		
. . . . . 2248		



<code>\proper@revstep@TP</code> .	1164, 1171,	. 317, 341, 381, 382
..... 3017, 3266	1930, 1947,	<code>\replacecolor@TP</code> ..
<code>\proper@step@TP</code> ...	1979, 2302,	.... 318, 321, 387
.. 2893, 2961, 3259	2336–2338,	<code>\replacecolors@TP</code> .
<code>\proper@steponce@TP</code>	2341, 2342,	..... 378,
..... 3024, 3267	2350–2352,	392–394, 401–403
<code>\proper@switch@TP</code> .	2361, 2362,	<code>\replacecolorsbyone@TP</code>
.. 2896, 2968, 3271	2370–2372,	..... 463, 469
<code>\proper@vstep@TP</code> ..	2376, 2377,	<code>\RequirePackage</code> ...
.. 3008, 3020, 3265	2395, 2418,	..... 1–3, 50, 97
<code>\protected@write</code> ..	2492, 2494,	<code>\reset@color</code> .. 2427,
..... 2509–2511	2496, 2535,	2428, 2444, 2448
<code>\protected@write@TP</code>	2550, 2575,	<code>\restep</code> ..... 2956,
..... 2510,	2909, 2927,	3260, 3278–3280
2514, 2516, 2518	3048, 3049,	<code>\restorecounters@TP</code>
<code>\provideboolean</code> .... 6	3168, 3171,	..... 3227, 3327
<code>\providecommand</code> ...	3181, 3218,	<code>\restorepanels</code> ... 1873
.... 703, 1177,	3283, 3288,	<code>\restorepausecounters@TP</code>
1192, 1795,	3294, 3297,	..... 2479, 2585
1801, 2471,	3304, 3314,	<code>\restoreTPcounters@TP</code>
2578, 2592, 3132	3317, 3321,	..... 3235, 3328
<code>\pushcolor@TP</code> 2429, 2435	3325, 3329,	<code>\result@TP</code> 137, 142,
<code>\pushcolorname@TP</code> .	3343, 3346, 3397	144, 146, 151,
..... 2432, 2456	<code>\releasecounter</code> .. 3239	153, 155, 157,
	<code>\remove@resetcolor@TP</code>	163, 168, 170,
	..... 2428, 2431	172, 177, 182,
	<code>\removecolor@TP</code> 351, 367	184, 186, 222,
	<code>\renewcommand</code> . 495,	262, 267, 269,
	574, 2219, 2257,	271, 276, 289, 294
	2277, 3244–3248	<code>\reswitch</code> 2922, 2963,
	<code>\renewenvironment</code> 3180	3272, 3278–3280
	<code>\repeat</code> 3300, 3320, 3344	<code>\revstep</code> ..... 3015,
	<code>\replacecolor</code> .....	3266, 3278–3280
	.... 315, 336,	<code>\rhd</code> ..... 2161
	436, 453, 465,	<code>\right</code> ..... 680
	532–534, 721,	<code>\rightpanelcontents@TP</code>
	783, 791, 846,	..... 1641,
	854, 862, 922,	1647, 1687,
	930, 978, 984,	1693, 1823,
	991, 1209, 1215,	1862, 1896,
	1262, 1268,	1898, 2039, 2041
	1293, 1299,	<code>\rightpanelheight</code> .
	1318, 1324,	... 1648, 1694,
	1330, 1349,	1818, 1907, 2041
	1355, 1361,	<code>\rightpanelheight@TP</code>
	1366–1372,	..... 1864, 1900
	1418, 1423,	<code>\rightpanelraise</code> ..
	1428, 1446,	..... 1650,
	1451, 1456, 1461	<code>\replacecolor@hook@TP</code>
	<code>\replacecolor@hook@TP</code>	1696, 1819, 2357

<code>\rightpanelraise@TP</code>	..... 1865, 1901	<code>\s@step@TPcheck</code> 2872, 2878, 2938, 2951	<code>\setcolor@TP</code> .....
<code>\rightpanelwidth</code> ..	... 1649, 1653, 1658, 1695, 1699, 1704, 1817, 1907, 2041	<code>\save@TP</code> 2521, 2580, 3254	.. 99, 515, 796, 801, 870, 936, 942, 999, 1238, 1246, 1486, 1495, 1504, 1513, 1523, 1534, 1549, 1557, 2125, 2146
<code>\rightpanelwidth@TP</code>	..... 1863, 1899	<code>\savecounters@TP</code> ..	
<code>\rightskip</code> .....	2058	..... 3224, 3255	
<code>\rlap</code> ...	1074, 1140, 1277, 2118, 2119, 2132, 2331, 2345, 2365, 2380, 2757, 2759, 2773, 2775, 2791, 2792, 2801, 3152, 3155	<code>\savepausecounters@TP</code>	
<code>\rule</code> .....	796, 801, 870, 936, 942, 999, 1487, 1496, 1505, 1514, 1524, 1535, 1550, 1558, 1753, 2126, 2748, 2759, 2764, 2766, 2775, 2777	..... 2476, 2525	
<code>\rulefirstgradprogression</code>	.... 732, 773, 827, 912, 964, 1253, 1304, 1335	<code>\saveTPcounters@TP</code> .	<code>\setcounter</code> . 46, 753, 775, 776, 808, 838, 839, 877, 878, 914, 915, 946, 972, 1003, 1004, 1200, 1226–1228, 1255, 1256, 1287, 1312, 1343, 1375, 1415, 1934, 2168, 2188, 2193, 3089, 3151
<code>\rulegradmidpoint</code> .	..... 734, 830, 966, 1306, 1337	..... 3232, 3326	<code>\setdgradslope</code> ....
<code>\rulesecondgradprogression</code>	. 733, 828, 965, 1286, 1305, 1336	<code>\scolname@TP</code> .....	..... 1200, 1203
<code>\rulestripes</code> .....	. 730, 775, 838, 914, 972, 1255, 1287, 1312, 1343	. 338, 339, 343, 345	<code>\setdoublehgradientbgnd@TP</code>
<b>S</b>		<code>\secondfactor@TP</code> ..	..... 1789
<code>\s@brackstep@TP</code> ...	..... 2862, 2870	..... 132, 133	<code>\setdoublelevgradientbgnd@TP</code>
<code>\s@parenstep@TP</code> ...	.. 2865, 2866, 2876	<code>\secondgradprogression@TP</code>	..... 1781
		. 741, 748, 828, 965, 1286, 1305, 1336, 1374, 1438	<code>\sethgradientbgnd@TP</code>
		<code>\sem@ptsize</code> .....	..... 1773
		..... 2054	<code>\setkeys</code> 1760, 1768, 1775, 1783, 1791
		<code>\set@color</code> 1276, 2423, 2431, 2437, 2439, 2996, 3012	<code>\setlength</code> .... 91, 93, 131, 133, 135, 136, 161, 162, 176, 261, 648, 651, 1039, 1085, 1086, 1099, 1166, 1179, 1236, 1938, 1950, 1980, 2058, 2137, 2138, 2145, 2688, 2704, 2715, 2740, 2743, 2744
		<code>\set@typeset@protect</code>	<code>\setnonebgnd@TP</code> .. 1754
		..... 2330	<code>\setplainbgnd@TP</code> . 1758
		<code>\setboolean</code> 5, 7, 8, 10, 12–14, 16, 20, 23, 26, 28, 30, 33, 36, 39, 48, 66, 1382, 1443, 1565, 1567, 1937, 1958, 2910, 2930, 2933, 2940, 2946, 2954, 3258	<code>\setvgradientbgnd@TP</code>
		<code>\setbox</code> .. 590, 1040, 1100, 1180, 1275, 1474, 1755, 1762, 1770, 1777, 1785, 1793, 1915, 1943, 1948, 1981, 2000, 2002, 2298, 2328, 2524, 2527, 2529, 2538, 2542, 2556, 2562, 2628, 2755, 2772, 2792, 2800, 3285	..... 1766

<code>\shipout</code> . . . . .	2292, 2293, 2395, 2411	<code>\SOUL@preamble</code> . . .	2738	<code>\storebottompanel@TP</code>	
<code>\shipout@hook@TP</code> . .	2310, 2325, 2411, 2412	<code>\SOUL@setkern</code> . . . . .	2636, 2642, 2755, 2772, 2800	.....	1842
<code>\shipout@output@TP</code> .	2303, 2305, 2308	<code>\SOUL@setup</code> . . . . .	2635	<code>\storeleftpanel@TP</code>	1851
<code>\shipout@test@TP</code> . .	2297, 2300	<code>\SOUL@token</code> .	2636, 2755	<code>\storerightpanel@TP</code>	1860
<code>\shipout@TP</code> .	2293, 2295	<code>\SOUL@uldepth</code> .	2740–2742	<code>\storetoppanel@TP</code>	1833
<code>\shipout@init@TP</code> . . .	2289, 2290	<code>\SOUL@uldp</code> . . . . .	2797	<code>\stretch</code> . . . . .	592, 595
<code>\shippingduplicatefalse</code>	2576	<code>\SOUL@uleverysyllable</code>	2793	<code>\string</code> . . .	329, 333, 342, 343, 2434, 2595, 2597–2599, 2649, 2654, 2809, 2815, 3398, 3399
<code>\shippingduplicatetrue</code>	2569	<code>\SOUL@ulht</code> . . . . .	2797	<code>\strip@pt</code> . . . . .	83, 84, 87, 88, 91, 132, 137, 163, 177, 262, 1087, 1091, 1167, 1171
<code>\skip</code> . . . . .	2530, 2557	<code>\SOUL@ulleaders</code> . . .	2757, 2773, 2801	<code>\stripeoverlap</code> . . . .	731, 818, 886, 902, 956, 1012, 1028
<code>\slide@ptsize</code> . . . .	2054	<code>\SOUL@ulpreamble</code> .	2750	<code>\switch</code> .	2597, 2881, 2895, 2899, 2909, 2920, 2938, 2968, 3081, 3086, 3155, 3158, 3271, 3278–3280
<code>\smallskip</code> . . .	717, 727	<code>\SOUL@ulsetup</code> .	2737, 2787	<b>T</b>	
<code>\smash</code> . . . . .	2736, 2746, 2757, 2759, 2764, 2766, 2773, 2775, 2777, 2792, 2797, 2801	<code>\SOUL@ulthickness</code> .	2745	<code>\tabcolsep</code> . . . . .	579
<code>\SOUL@</code> .	2645, 2781, 2803	<code>\space</code> . .	1060, 1121, 1123, 2598, 2599, 2649, 2654, 2809, 2815	<code>\tcolname@TP</code> . . . . .	323, 341, 342, 344
<code>\SOUL@boxdepth@TP</code> .	2733, 2740, 2744, 2748, 2759, 2764, 2766, 2775, 2777	<code>\spaceskip</code> .	2765, 2797	<code>\tempbox@TP</code> .	80, 1275, 1279–1282, 1915, 1916, 1948, 1950, 1954, 2542, 2562, 2564, 2628, 2755, 2756, 2758, 3285
<code>\SOUL@boxheight@TP</code> .	2731, 2742–2744	<code>\special</code> . . . . .	1118, 1127, 1141, 1146	<code>\tempdima@TP</code> . . .	77, 91, 93, 131–137, 161–163, 176, 177, 261, 262, 1950, 1951, 1968
<code>\SOUL@boxtotalheight@TP</code>	2732, 2744, 2745, 2748, 2759, 2764, 2766, 2775, 2777	<code>\star@TP</code> . . . . .	3188, 3189, 3194, 3203, 3204, 3209	<code>\tempdimb@TP</code> . . . . .	78, 1938, 1951, 1968
<code>\SOUL@charkern</code> . . . . .	2636, 2755	<code>\stdbuttonwidth</code> .	2164, 2166, 2171, 2176, 2181, 2186, 2191, 2196	<code>\texpower@endfragilesteps</code>	3374, 3388, 3402
<code>\SOUL@everyhyphen</code> .	2637, 2768, 2798	<code>\step</code> .	2597, 2887, 2892, 2961, 3041, 3218, 3219, 3259, 3277–3280, 3283, 3284, 3291, 3304, 3305, 3325		
<code>\SOUL@everyspace</code> . .	2762, 2796	<code>\stepcounter</code> . . . . .	814, 887, 889, 898, 952, 1013, 1015, 1024, 1239, 1247, 1942, 3092, 3156		
<code>\SOUL@everysyllable</code>	2780, 2788	<code>\steponce</code> . .	3022, 3267		
<code>\SOUL@everytoken</code> . .	2636, 2752	<code>\stepwise</code> .	2070, 2595, 2599, 3163, 3166, 3194, 3209, 3333, 3355		
<code>\SOUL@hyphkern</code> . . . .	2642, 2772, 2800	<code>\stopAdvancing</code> . . . .	2287, 3128		

<code>\texpower@processframefirstline</code>	956, 1008, 1010, ..... 3368, 3371	1011, 1021,	<code>\unkern</code> . 1990–1999, 2641, 2771, 2799
<code>\texpower@processframeline</code>	1023, 1026, ..... 3381, 3385	1028, 1234,	<code>\unpenalty</code> .. 1990–1999
<code>\texpower@stopframe</code>	1235, 1244, ..... 3387, 3398	1245, 2844, 2845	<code>\unskip</code> ..... 1990–1999
<code>\texpower@stopframefirst</code>	<code>\toks</code> ..... 588		<code>\unvbox</code> ..... 659, 1984
<code>\texpower@test</code> ....	..... 3372, 3373, 3376, 3386, 3387	1575, 1581, 1711, 1717, 1820,	<code>\usebox</code> ..... 726
<code>\texpower@verbatimfilename</code>	.. 3349, 3350, 3355	1835, 1875, 1877, 2015, 2017	<code>\usecolorset</code> .. 390, 473, 480, 485, 490, 519, 2071, 2667, 2824, 3333
<code>\texpower@verbatimfileout</code>	... 3350, 3354, 3357, 3379, 3390	<code>\toppanelheight</code> ... ... 1582, 1587, 1592, 1718,	<b>V</b>
<code>\texpower@verbatimreadframe</code>	..... 3351, 3359	1723, 1728, 1809, 1904, 2017	<code>\value</code> .... 754, 756, 758, 761, 764, 799, 809, 812, 815, 816, 818, 820, 823, 869, 879, 880, 883, 886, 890, 891, 893, 896, 899, 900, 902, 904, 905, 908, 940, 947, 950, 953, 954, 956, 958, 960, 998, 1005, 1006, 1009, 1012, 1016, 1017, 1019, 1022, 1025, 1026, 1028, 1030, 1031, 1033, 1232, 1234, 1236, 1237, 1242, 1244, 1940, 1946, 1969, 2165, 2492, 2502, 2514, 2846, 2923, 2925, 2960, 2967, 2978, 3003, 3019, 3090, 3093, 3153, 3293, 3314
<code>\text</code> ..... 505, 506	<code>\toppanelheight@TP</code> . ..... 1837, 1879		
<code>\text@db@TP</code> . 2615, 2618	<code>\toppanelshift</code> 1584, 1720, 1810, 2373		
<code>\text@hb@TP</code> ... 2694, 2697, 2700, 2711	<code>\toppanelshift@TP</code> . ..... 1838, 1880		
<code>\textbackslash</code> .... 678	<code>\toppanelwidth</code> .... ... 1583, 1719, 1808, 1904, 2017		
<code>\textbf</code> ... 671, 675, 686, 693, 707, 711	<code>\toppanelwidth@TP</code> . ..... 1836, 1878		
<code>\textcolor</code> ... 671, 683, 690, 697, 2660, 2662, 2676	<code>\TPeject</code> ... 2471, 2573		
<code>\textnormal</code> ... 497, 498	<code>\TPpageheight</code> ..... .. 84, 88, 1616, 1662, 1663, 1809, 1812, 1815, 1818, 2333, 2347, 2367		
<code>\textsuperscript</code> .. ..... 663, 664	<code>\TPpagewidth</code> ..... .. 83, 87, 1571, 1572, 1707, 1808, 1811, 1814, 1817, 2353		
<code>\texttt</code> ..... 671, 675	<code>\ttfamily</code> .... 672, 676		
<code>\textwidth</code> ..... 1921			
<code>\the</code> 93, 506, 542, 588, 762, 2133, 2134, 2441, 2478, 2636, 2755, 2900, 2916, 2925, 3226, 3234, 3298, 3307, 3318, 3379			
<code>\thetmpcnta@TP</code> .... ... 1957, 1964, 2168, 2188, 2193			
<code>\thicklines</code> ..... 1225			
<code>\tmp@TP</code> ... 811, 813, 816, 818, 882, 884, 885, 895, 897, 900, 902, 949, 951, 954,	<b>U</b>		
	<code>\undefinecolor@TP</code> . 328		<code>\vanishcolor</code> ..... ... 468, 469, 2676
	<code>\undefined</code> ..... 52		<code>\vanishcolors</code> ..... 469, 525, 2676, 3012
	<code>\underl</code> ..... 681		<code>\vartriangleleft</code> . 2157
	<code>\unhbox</code> 613, 659, 1074, 1140, 2002, 2758		

