The texpower package*

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

April 9, 2005

Contents

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

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.

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

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

```
\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.

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

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

```
\newboolean{oldfiltering@TP}
\DeclareOption{oldfiltering}{\setboolean{oldfiltering@TP}{true}}
\newcommand{\oldfilteringon}{\setboolean{oldfiltering@TP}{true}}
\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.

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

3.1.2 Color options

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

```
\newboolean{TPcolor}
```

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

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

The option colormath makes math formulae be color highlighted.

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

The option colorhighlight makes highlighting commands use colors.

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

The option whitebackground selects standard colors for white backgrounds.

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

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

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

The option blackbackground selects standard colors for black backgrounds.

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

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

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

Load the config file with default options if file exists.

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

Process options.

```
\ProcessOptions

\ifthenelse{\boolean{display}}
{\PackageInfo{texpower}{Producing display version. Dynamic features activated.}}
{\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.

```
\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.

```
\newboolean{psspecialsallowed}
\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:

```
\@ifundefined{pdftrue}{
\IfFileExists{ifpdf.sty}{\RequirePackage{ifpdf}}{%
 \expandafter\newif\csname ifpdf\endcsname
 \ifx\pdfoutput\undefined
  \else
   \ifx\pdfoutput\relax
```

```
    \else
     \ifcase\pdfoutput
      \else
       \pdftrue
      \fi
    \fi
  \fi
 }
}{% \ifpdf is defined - nothing to do
}

\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.

```
\@ifclassloaded{slides}%
{%
  \ifthenelse{\boolean{display}}%
  {\let\@topfil\relax}%
  {}%
  }
{}
```

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

```
\newcounter{tmpcnta@TP}
\newcounter{tmpcntb@TP}

\newlength{\tempdima@TP}
\newlength{\tempdimb@TP}

\newbox\tempbox@TP

\newboolean{carryon@TP}
```

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

```
\newcommand{\TPpagewidth}{\strip@pt\paperwidth truept}
\newcommand{\TPpageheight}{\strip@pt\paperheight truept}
% \AtBeginDocument
% {%
%   \edef\TPpagewidth{\strip@pt\paperwidth truept}%
%   \edef\TPpageheight{\strip@pt\paperheight truept}%
% }
```

`\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>`.

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

Make a string representation of a length expression.

```
\newcommand{\mklength@TP}[2]
{\setlength{\tempdima@TP}{#2}\edef#1{\the\tempdima@TP}}

\newcommand{\mklength}{}
\let\mklength\mklength@TP
```

3.2 Color management, color emphasis and highlighting

Initialization.

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

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

3.2.1 Color management kernel

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

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

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

```
  \let\setcolor@TP=\color%
```

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

```
  \let\o@definecolor@TP=\definecolor%
  \def\definecolor#1#2#3%
  {%
    \o@definecolor@TP{#1}{#2}{#3}%
    \expandafter\edef\csname colordef@TP@#1\endcsname%
    {\csname processcolor@TP@#2\endcsname{#3}}%
    }%
```

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

```
  \@ifundefined{c@lor@namefile}{}{\input{\c@lor@namefile}}

  \ifx\color@gray\@undefined
    \ifx\color@rgb\@undefined
    \else
      \definecolor{black}{rgb}{0,0,0}
      \definecolor{white}{rgb}{1,1,1}
    \fi
  \else
    \definecolor{black}{gray}{0}
    \definecolor{white}{gray}{1}
  \fi
  \ifx\color@rgb\@undefined\else
    \definecolor{red}{rgb}{1,0,0}
```

```
    \definecolor{green}{rgb}{0,1,0}
    \definecolor{blue}{rgb}{0,0,1}
  \fi
  \ifx\color@cmyk\@undefined\else
    \definecolor{cyan}{cmyk}{1,0,0,0}
    \definecolor{magenta}{cmyk}{0,1,0,0}
    \definecolor{yellow}{cmyk}{0,0,1,0}
  \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.

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

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

```
\def\interpolate@three@TP#1,#2,#3;#4,#5,#6;#7%
{%
  \interpolate@TP{#7}{#1}{#4}%                      First intermediary value.
  \edef\newcolordef@TP{\result@TP,}%                Store first value.
  \interpolate@TP{#7}{#2}{#5}%                      Second intermediary value.
  \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
  \interpolate@TP{#7}{#3}{#6}%                      Third intermediary value.
  \edef\newcolordef@TP{\newcolordef@TP\result@TP}%  Store third value.
  }
```

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

```
\def\interpolate@four@TP#1,#2,#3,#4;#5,#6,#7,#8;#9%
{%
  \interpolate@TP{#9}{#1}{#5}%                      First intermediary value.
  \edef\newcolordef@TP{\result@TP,}%                Store first value.
  \interpolate@TP{#9}{#2}{#6}%                      Second intermediary value.
  \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
  \interpolate@TP{#9}{#3}{#7}%                      Third intermediary value.
  \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store third value.
  \interpolate@TP{#9}{#4}{#8}%                      Fourth intermediary value.
  \edef\newcolordef@TP{\newcolordef@TP\result@TP}%  Store fourth value.
  }
```

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

```
\def\convert@cmykvalue@rgbvalue@TP#1#2%
{%
  \setlength{\tempdima@TP}{1pt-#1pt-#2pt}%
  \ifthenelse{\lengthtest{\tempdima@TP<0pt}}{\setlength{\tempdima@TP}{0pt}}{}%
  \edef\result@TP{\strip@pt\tempdima@TP}%
  }%
```

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

```
\def\convert@cmyk@rgb@TP#1,#2,#3,#4;%
{%
  \convert@cmykvalue@rgbvalue@TP{#1}{#4}%
  \edef\newcolordef@TP{\result@TP,}%                Store first value.
  \convert@cmykvalue@rgbvalue@TP{#2}{#4}%
  \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
  \convert@cmykvalue@rgbvalue@TP{#3}{#4}%
  \edef\newcolordef@TP{\newcolordef@TP\result@TP}%  Store third value.
  }
```

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

```
\def\convert@RGBvalue@rgbvalue@TP#1%
{%
  \setlength{\tempdima@TP}{#1pt/255}%
  \edef\result@TP{\strip@pt\tempdima@TP}%
  }%
```

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

```
\def\convert@RGB@rgb@TP#1,#2,#3;%
{%
  \convert@RGBvalue@rgbvalue@TP{#1}%
  \edef\newcolordef@TP{\result@TP,}%                Store first value.
  \convert@RGBvalue@rgbvalue@TP{#2}%
  \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
  \convert@RGBvalue@rgbvalue@TP{#3}%
  \edef\newcolordef@TP{\newcolordef@TP\result@TP}%  Store third value.
  }
```

`\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:

As I don't know how to convert hsb to rgb, mixing hsb with another color model will always raise an error.

```
\newcommand{\colorbetween}[4][.5]%
{%
  \begingroup%                                  Make the definition of \processcolor... local.
    \newcommand{\processcolor@TP@rgb}[3]%       What if the first color is an rgb color?
    {%
      \ifx##2\processcolor@TP@rgb%              Are both colors rgb colors?
        \interpolate@three@TP##1;##3;{#1}%      Calculate interpolated values.
       \else
        \ifx##2\processcolor@TP@gray%           Is the second color from the gray model?
          \interpolate@three@TP##1;##3,##3,##3;{#1}%    Calculate interpolated values.
         \else
          \ifx##2\processcolor@TP@cmyk%         Is the second color from the cmyk model?
            \convert@cmyk@rgb@TP##3;%           Convert to rgb.
            \edef\newcolordef@TP{\noexpand\interpolate@three@TP##1;\newcolordef@TP;{#1}}%
            \newcolordef@TP
           \else
            \ifx##2\processcolor@TP@RGB%        Is the second color from the RGB model?
              \convert@RGB@rgb@TP##3;%          Convert to rgb.
              \edef\newcolordef@TP{\noexpand\interpolate@three@TP##1;\newcolordef@TP;{#1}}%
              \newcolordef@TP
             \else
              \ifx##2\processcolor@TP@hsb%      Is the second color from the hsb model?
                \PackageError{texpower}{Don't know how to convert an hsb color!}
              \fi
            \fi
          \fi
        \fi
      \fi
      \edef\newcolordef@TP{{rgb}{\newcolordef@TP}}% Store the result
      }%
    \newcommand{\processcolor@TP@gray}[3]%          What if the first color is a gray color?
    {%
      \ifx##2\processcolor@TP@gray%                 Are both colors gray colors?
        \interpolate@TP{#1}{##1}{##3}%              Calculate interpolated value.
        \edef\newcolordef@TP{{gray}{\result@TP}}%   Store the result
       \else
        \processcolor@TP@rgb{##1,##1,##1}{##2}{##3}% Otherwise, convert first color to rgb an
      \fi
      }%
    \newcommand{\processcolor@TP@cmyk}[3]%          What if the first color is a cmyk color?
    {%
      \ifx##2\processcolor@TP@cmyk%                 Are both colors cmyk colors?
        \interpolate@four@TP##1;##3;{#1}%           Calculate interpolated values.
        \edef\newcolordef@TP{{cmyk}{\newcolordef@TP}}% Store the result
       \else
        \convert@cmyk@rgb@TP##1;%                   Otherwise, convert first color to rgb ...
        \expandafter\processcolor@TP@rgb\expandafter{\newcolordef@TP}{##2}{##3}% ... and cont
```

```
      \fi
      }%
    \newcommand{\processcolor@TP@RGB}[3]%          What if the first color is an RGB color?
    {%
      \convert@RGB@rgb@TP##1;%                     Convert to rgb ...
      \expandafter\processcolor@TP@rgb\expandafter{\newcolordef@TP}{##2}{##3}% ... and contin
      }%
    \newcommand{\processcolor@TP@hsb}[3]%          What if the first color is an hsb color?
    {%
      \ifx##2\processcolor@TP@hsb%                 Are both colors hsb colors?
        \interpolate@three@TP##1;##3;{#1}%         Calculate interpolated values.
        \edef\newcolordef@TP{{hsb}{\newcolordef@TP}}% Store the result
       \else
        \PackageError{texpower}{Don't know how to convert an hsb color!}
      \fi
      }%
    \expandafter\let\expandafter
    \firstcol@TP\csname colordef@TP@#3\endcsname    % Retrieve definition of color <source1>.
    \expandafter\expandafter\expandafter\firstcol@TP% and apply (remember \processcolor... is
    \csname colordef@TP@#4\endcsname%                to definition of color <source2>.
    \edef\end@TP%                                    Define color <target> (outside the encl
    {\endgroup\noexpand\definecolor{#2}\newcolordef@TP}%
  \end@TP
  }% matches \newcommand{\colorbetween}
```

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

```
\newcommand{\complement@TP}[1]%
{%
  \setlength{\tempdima@TP}{1pt-#1pt}%
  \edef\result@TP{\strip@pt\tempdima@TP}%
  }
```

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

```
\def\complement@three@TP#1,#2,#3;%
{%
  \complement@TP{#1}%
  \edef\newcolordef@TP{\result@TP,}%                Store first value.
  \complement@TP{#2}%
  \edef\newcolordef@TP{\newcolordef@TP\result@TP,}% Store second value.
  \complement@TP{#3}%
  \edef\newcolordef@TP{\newcolordef@TP\result@TP}%  Store third value.
  }
```

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

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

\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.

```
\newcommand{\complementcolor}[2]%
{%
  \begingroup%                                  Make the definition of \processcolor... local.
    \newcommand{\processcolor@TP@rgb}[1]%       What if the color is an rgb color?
    {%
      \complement@three@TP##1;%                 Calculate complemented values.
      \edef\newcolordef@TP{{rgb}{\newcolordef@TP}}% Store the result
      }%
    \newcommand{\processcolor@TP@gray}[1]%      What if the color is a gray color?
    {%
      \complement@TP{##1}%                      Calculate complemented value.
      \edef\newcolordef@TP{{gray}{\result@TP}}% Store the result
      }%
    \newcommand{\processcolor@TP@cmyk}[1]%      What if the color is a cmyk color?
    {%
      \grabfourth@TP##1;%                       Remember fourth element.
      \expandafter\complement@three@TP\result@TP%  Calculate complemented values of first thr
      \edef\newcolordef@TP{{cmyk}{\newcolordef@TP,\mem@TP}}% Store the result, putting back t
      }%
    \newcommand{\processcolor@TP@RGB}[1]%       What if the color is an RGB color?
    {%
      \convert@RGB@rgb@TP##1;%                  Convert to rgb ...
      \expandafter\processcolor@TP@rgb\expandafter{\newcolordef@TP}% ... and continue.
      }%
    \newcommand{\processcolor@TP@hsb}[1]%       What if the color is an hsb color?
    {%
      \complement@three@TP##1;%                 Calculate complemented values.
      \edef\newcolordef@TP{{hsb}{\newcolordef@TP}}% Store the result
      }%
    \csname colordef@TP@#2\endcsname%           Execute definition of color <source> (which cont
    \edef\end@TP%                               Define color <target> (outside the enclosing gro
    {\endgroup\noexpand\definecolor{#1}\newcolordef@TP}%
  \end@TP
  }% matches \newcommand{\complementcolor}
}% matches \ifthenelse{\boolean{TPcolor}}{% Yes.
{% No. Do nothing.
}
```

3.3 Color name and color set management

\replacecolor `\replacecolor[<tset>]{<tcol>}[<sset>]{<scol>}` will make `<tcol>` have the same definition as `<scol>` (if `<scol>` is defined at all), where ¡tcol¿ and ¡scol¿ 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.

```
\newcommand{\replacecolor}
{%
  \let\replacecolor@hook@TP=\@gobble% This hook can be used for variant checking (see below).
  \replacecolor@TP%                  Pick up arguments.
  }

\newcommand{\replacecolor@TP}[2][]% Pick up the first two arguments of \replacecolor.
{%
  \ifthenelse{\equal{#1}{}}{\edef\tcolname@TP{#2}}{\edef\tcolname@TP{#2@#1}}% Construct 'real'
  \@replacecolor@TP%                                                        Read second argumen
  }%

\ifthenelse{\boolean{TPcolor}}% Only if TeXPower's color management is active.
{% Yes.
  \newcommand{\undefinecolor@TP}[1]%                                  Make a color undefined.
  {\expandafter\let\csname\string\color @#1\endcsname=\@undefined}%


  \newcommand{\ifcolorexists@TP}[3]%                                  Conditional for testing whe
  {\@ifundefined{\string\color @#1}{#3}{#2}}%                         Test whether a given color


  \newcommand{\@replacecolor@TP}[2][]%                                Second part of \replacecolo
  {%
    \ifthenelse{\equal{#1}{}}{\edef\scolname@TP{#2}}{\edef\scolname@TP{#2@#1}}% Construct 'real
    \ifcolorexists@TP{\scolname@TP}%                                  Does the source
    {% Yes.
      \replacecolor@hook@TP{\tcolname@TP}%                            Execute hook.
      \expandafter\let\csname\string\color @\tcolname@TP\expandafter\endcsname% Make value of t
      \csname\string\color @\scolname@TP\endcsname%                   identical with
      \expandafter\let\csname colordef@TP@\tcolname@TP\expandafter\endcsname%   Make definition
      \csname colordef@TP@\scolname@TP\endcsname%                     identical with
      }%
    {% No. Do nothing.
      }%
    }%
```

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.

```
  \newcommand{\colors@TP}{}
```

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

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

```
      {% Yes. Do nothing, so it vanishes.
        }
      {% No. Re-insert.
        \expandafter\def\expandafter\colors@TP\expandafter{\colors@TP\processme@TP{##1}}%
        }%
      }%
    \expandafter\let\expandafter\colors@TP\expandafter\empty% Initialize \colors@TP.
    \colors@TP%                                              Execute \processme@TP for every c
    }
```

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

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

`\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.

```
\newcommand{\defineTPcolor}[4][]
{%
  \addTPcolor{#2}%                   Add color to the list.
  \ifthenelse{\equal{#1}{}}%         Color from the current color set?
  {\definecolor{#2}{#3}{#4}}%        Yep. Just define the color.
  {\definecolor{#2@#1}{#3}{#4}}%     No. Add color set identifyer.
  }
```

Some commands for manipluating whole color sets.

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

```
\newcommand{\replacecolors@TP}%
{%
  \@ifstar%                                    The starred version will put the color set int
  {\let\replacecolor@hook@TP=\register@normalvariant@TP\@replacecolors@TP}
  {\let\replacecolor@hook@TP=\@gobble\@replacecolors@TP}%
  }

\newcommand{\@replacecolors@TP}[4]%            This part does the real work.
{%
  \def\processme@TP##1{\replacecolor@TP[#1]{#2##1}[#3]{#4##1}}%
  \colors@TP
  }
```

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

```
\newcommand{\usecolorset}[1]%
{%
  \replacecolors@TP*{}{}{#1}{}%   Replace normal variant (registering variants).
  \replacecolors@TP{}{d}{#1}{d}%  Replace dimmed variant.
  \replacecolors@TP{}{e}{#1}{e}%  Replace enhanced variant.
  \color{textcolor}%              Activate textcolor.
  \pagecolor{pagecolor}%          Activate pagecolor.
  }%
```

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

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

Commands for color variants.

```
\newcommand{\registervariant@TP}[2]%          Remember which variant a color is currently in
{\expandafter\def\csname cvar@#1@TP\endcsname{#2}}

\newcommand{\register@normalvariant@TP}[1]%   Register that a color is now in the normal var
{\registervariant@TP{#1}{}}

\newcommand{\currentvariant@TP}[1]%           Return the current variant of a color.
{\csname cvar@#1@TP\endcsname}

\newcommand{\ifnormalvariant@TP}[3]%          Conditional for checking whether a color is in
{\ifthenelse{\equal{\currentvariant@TP{#1}}{}}{#2}{#3}}

\newcommand{\nonnormalwarnings@TP}[1]%        Checks the current variant for every standard
{%                                            if it's not the normal one.
  \def\processme@TP##1%
  {%
    \ifnormalvariant@TP{##1}{}
    {%
      \PackageWarning{texpower}
      {#1\MessageBreak when color ##1 is in \currentvariant@TP{##1} variant}%
      }%
    }%
  \colors@TP
  }
```

Default dim level for automatic color dimming.

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

`\dimcolor[<level>]{<color>}` dims the color named `<color>`. It checks whether an explicit 'dimmed' variant d¡color¿ 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).

```
\newcommand{\dimcolor}[2][\dimlevel]
{%
  \ifnormalvariant@TP{#2}%            Color in the normal variant?
  {% Yes.
    \registervariant@TP{#2}{d}%       Register dimmed variant.
    \ifcolorexists@TP{d#2}%           Dedicated dimmed color found?
    {\replacecolor{#2}{d#2}}%         Yes. use that one.
    {\colorbetween[#1]{#2}{pagecolor}{#2}}% No. Dim numerically using \colorbetween.
    }%
  {}% No. Do nothing.
  }
```

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

```
\newcommand{\dimcolors}[1][\dimlevel]
{%
  \def\processme@TP##1{\dimcolor[#1]{##1}}%
  \colors@TP
  }%
```

Default enhance level for automatic color enhancing.

```
\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).

```
\newcommand{\enhancecolor}[2][\enhancelevel]
{%
  \ifnormalvariant@TP{#2}%            Color in the normal variant?
  {%
    \registervariant@TP{#2}{e}%       Register enhanced variant.
    \ifcolorexists@TP{e#2}%           Dedicated enhanced color found?
    {\replacecolor{#2}{e#2}}%         Yes. use that one.
    {\colorbetween[-#1]{#2}{pagecolor}{#2}}% No. Enhance numerically using \colorbetween.
    }%
  {}%
  }%
```

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

```
\newcommand{\enhancecolors}[1][\enhancelevel]
{%
  \def\processme@TP##1{\enhancecolor[#1]{##1}}%
  \colors@TP
  }%
```

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

```
\newcommand{\replacecolorsbyone@TP}[2]%
{%
  \def\processme@TP##1{\replacecolor{#1##1}{#2}}%
  \colors@TP
  }
```

The color used to make things 'vanish'.

```
\newcommand{\vanishcolor}{pagecolor}
```

`\vanishcolors` replaces all standard colors by `\vanishcolor`.

```
\newcommand{\vanishcolors}[1][\vanishcolor]{\replacecolorsbyone@TP{}{#1}}
```

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

```
\input{tpcolors.cfg}
```

`\whitebackground` sets the standard colors up for white background.

```
\newcommand{\whitebackground}%
{%
  \usecolorset{whitebg}%
  }
```

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).

```
\ifthenelse{\boolean{whitebackground@TP}}
{\AtEndOfPackage{\whitebackground}}
{}
```

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

```
\newcommand{\lightbackground}%
{%
  \usecolorset{lightbg}%
  }
```

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

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

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

```
\newcommand{\darkbackground}%
{%
  \usecolorset{darkbg}%
  }
```

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

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

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

```
\newcommand{\blackbackground}%
{%
  \usecolorset{blackbg}%
  }
```

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

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

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

```
\ifthenelse{\boolean{TPcolor}}
{%
  \renewcommand{\normalcolor}{\color{textcolor}}%      \normalcolor should produce textcolor.

  \let\o@textnormal@TP=\textnormal%                       \textnormal should also set text co
  \def\textnormal#1{\o@textnormal@TP{\normalcolor#1}}
```

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

```
  \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.

```
  \AtBeginDocument%
  {%
    \pagecolor{pagecolor}\color{textcolor}%
    \@ifpackageloaded{amstext}%
    {%
      \let\o@text@TP=\text%
      \def\text#1{\o@text@TP{\normalcolor\expandafter\everymath\expandafter{\the\everymath\co
    }%
    {}%
  }%
}
{}
```

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

```
\ifthenelse{\boolean{TPcolor}}{\AtBeginDocument{\pagecolor{pagecolor}\color{textcolor}}}{}
}% matches \ifthenelse{\boolean{TPcolor}}{% Yes.
{% No; provide dummies.
\let\setcolor@TP=\@gobble%
\newcommand{\@replacecolor@TP}[2][]{}%
\let\addTPcolor=\@gobble
\newcommand{\defineTPcolor}[4][]{}%
\let\usecolorset=\@gobble
\let\dumpcolorset=\@gobble
\newcommand{\dimcolor}[2][]{}
\newcommand{\dimcolors}[1][]{}
\newcommand{\enhancecolor}[2][]{}
\newcommand{\enhancecolors}[1][]{}
\newcommand{\vanishcolors}[1][]{}
}
```

3.3.1 Implementation of the coloremph option

```
\ifthenelse{\boolean{coloremph@TP}}%                    Should \emph use color?
{%  Yes;
  \DeclareRobustCommand{\em}%                           Redefine \em.
  {%
    \@nomath\em \color{emcolor}%                        Change color.
    \replacecolor{tmp@TP}{emcolor}%                     Exchange emcolor and altemcolor.
    \replacecolor{emcolor}{altemcolor}%
    \replacecolor{altemcolor}{tmp@TP}%
    }%
  }%
{}% 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.

```
\ifthenelse{\boolean{colormath@TP}}%                    Should we color math?
{%  Yes.
  \AtBeginDocument
  {%
```

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

```
    \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.

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

```
    \def\endeqnarray{\o@endeqa@TP\endgroup\@ignoretrue}%
    \@namedef{eqnarray*}{\begingroup\color{mathcolor}\o@eqastar@TP}
    \@namedef{endeqnarray*}{\o@endeqastar@TP\endgroup\@ignoretrue}
    \@ifpackageloaded{amsmath}%                              Amsmath's displayed math environment
    {%                                                       \everymath hack because they are 'fa
      \@ifpackagelater{amsmath}{2000/01/15}%                   As amsmath 1.x redefines the equat
      {%                                                       variant of gather, treating it as
        \let\o@eq@TP=\equation%                                environment would lead to problems
        \let\o@endeq@TP=\endequation%                          made only if amsmath 1.x is not lo
        \def\equation{\o@eq@TP\begingroup\color{mathcolor}}%
        \def\endequation{\endgroup\everymath{}\o@endeq@TP}%
      }%
      {}%
    }%
    {%
      \let\o@eq@TP=\equation%
      \let\o@endeq@TP=\endequation%
      \def\equation{\o@eq@TP\begingroup\color{mathcolor}}%
      \def\endequation{\endgroup\everymath{}\o@endeq@TP}%
    }%
  }% 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.

```
  \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.

```
  \renewcommand*\labelitemi{\origmath{\m@th\bullet}}%
  \@ifpackageloaded{array}%                                The array package redefines \@tabular
  {%
    \def\@tabular{%
      \leavevmode
      \hbox \bgroup \everymath{}$\everymath{\color{mathcolor}}\col@sep\tabcolsep \let\d@llarbeg
      \let\d@llarend\endgroup
      \@tabarray
    }%
    \@ifpackageloaded{colortbl}
    {%
      \def\@classz{\@classx
        \@tempcnta \count@
        \prepnext@tok
        \expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil
        \@addtopreamble{%
          \setbox\z@\hbox\bgroup\bgroup
          \ifcase \@chnum
          \hskip\stretch{.5}\kern\z@
```

```
          \d@llarbegin
          \insert@column
          \d@llarend\hskip\stretch{.5}\or
          \d@llarbegin \insert@column \d@llarend \hfill \or
          \hfill\kern\z@ \d@llarbegin \insert@column \d@llarend \or
          \@startvcenter
          \@startpbox{\@nextchar}\insert@column \@endpbox $\or % $
          \vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
          \vbox \@startpbox{\@nextchar}\insert@column \@endpbox
          \fi
          \egroup\egroup
          \begingroup
          \CT@setup
          \CT@column@color
          \CT@row@color
          \CT@do@color
          \endgroup
          \@tempdima\ht\z@
          \advance\@tempdima\minrowclearance
          \vrule\@height\@tempdima\@width\z@
          \unhbox\z@}%
        \prepnext@tok}%
    }
    {%
      \def\@classz{\@classx
        \@tempcnta \count@
        \prepnext@tok
        \@addtopreamble{\ifcase \@chnum
          \hfil
          \d@llarbegin
          \insert@column
          \d@llarend \hfil \or
          \hskip1sp\d@llarbegin \insert@column \d@llarend \hfil \or
          \hfil\hskip1sp\d@llarbegin \insert@column \d@llarend \or
          \@startvcenter
          \@startpbox{\@nextchar}\insert@column \@endpbox $\or        % $
          \vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
          \vbox \@startpbox{\@nextchar}\insert@column \@endpbox
          \fi}\prepnext@tok}%
    }
    \DeclareRobustCommand\@startvcenter{\everymath{}$\everymath{\color{mathcolor}}\vcenter}% $
    \expandafter\def\expandafter\@mkpream\expandafter#\expandafter1%
    \expandafter{%
      \expandafter\let\expandafter\@startvbox\expandafter\relax
      \@mkpream{#1}}
  }
  {%
    \def\@tabular{\leavevmode \hbox \bgroup \everymath{}$\everymath{\color{mathcolor}}\let\@aco
      \let\@classz\@tabclassz
      \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray% $
```

```
      }%
    }
  \long\def\@iiiparbox#1#2[#3]#4#5{%
    \leavevmode
    \@pboxswfalse
    \setlength\@tempdima{#4}%
    \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%
    \ifx\@empty#2\else\ifx\relax#2\else
      \setlength\@tempdimb{#2}%
      \def\@parboxto{to\@tempdimb}%
    \fi\fi
    \if#1b\vbox
      \else\if #1t\vtop
      \else\ifmmode\vcenter
      \else\@pboxswtrue \everymath{}$\everymath{\color{mathcolor}}\vcenter
    \fi\fi\fi
    \@parboxto{\let\hss\vss\let\unhbox\unvbox
      \csname bm@#3\endcsname}%
    \if@pboxsw \m@th$\fi
    \@end@tempboxa}
  \let\o@textsuperscript@TP=\textsuperscript
  \def\textsuperscript#1{{\everymath{}\o@textsuperscript@TP{\everymath{\color{mathcolor}}#1}}}%
  }% matches \ifthenelse{\boolean{colormath@TP}}{%  Yes.
{% No; keep original definition.
  \let\origmath=\ensuremath%      \origmath needs to have a sensible definition.
}
```

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.

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

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

```
\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, ¡arg¿ 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.

```
\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.

```
\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.

```
\ifthenelse{\boolean{colorhighlight@TP}}%                  Color highlighting enabled?
{%  Yes;
  \DeclareRobustCommand{\underl}{\textcolor{underlcolor}}% Use color to highlight.
  }
{%  No;
  \DeclareRobustCommand{\underl}{\textbf}%                 Use bold face.
  }
```

\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.

```
\ifthenelse{\boolean{colorhighlight@TP}}%                  Color highlighting enabled?
{%  Yes;
  \DeclareRobustCommand{\concept}{\textcolor{conceptcolor}}% Use color to highlight.
  }
{%  No;
  \DeclareRobustCommand{\concept}{\textbf}%                  Use bold face.
  }
```

\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.

```
\ifthenelse{\boolean{colorhighlight@TP}}%                  Color highlighting enabled?
{%  Yes;
  \DeclareRobustCommand{\inactive}{\textcolor{inactivecolor}}% Use color to highlight.
  }
{%  No;
  \DeclareRobustCommand{\inactive}{\monochromeinactive}%       Use monochrome default.
  }

\providecommand{\monochromeinactive}{}% What should \inactive do if colors can't be used? We pr
                                      % user definitions.
```

\present `\present[<opt>]{<text>}` puts its argument into an `\fbox` with coloured background. 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.

```
\ifthenelse{\boolean{colorhighlight@TP}}%                  Color highlighting enabled?
{%  Yes; use a colored box.
  \newcommand{\present}[2][]{\leavevmode\llap{\textbf{\footnotesize#1}\,}\fcolorbox{textcolor}{
  \newcommand{\mkpbox@TP}[1]{\fcolorbox{presentcolor}{presentcolor}{#1}}% Internal macro for us
  }
```

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

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.

```
\newsavebox{\pbox@TP}%                                  Container for the minipage to be boxed.
\newenvironment{presentbox}%
{%
  \par\smallskip%                                       First a small space to separate the are
  \begin{lrbox}{\pbox@TP}%                              Save the contents in a minipage inside
    \noindent
    \begin{minipage}{\linewidth-2\fboxsep-2\fboxrule}%  Reduce the width of the minipage to lea
      \replacecolor{presentcolor}{pagecolor}%           If \present is used inside the colored
      }%
    {%
    \end{minipage}
  \end{lrbox}%
  \noindent\mkpbox@TP{\usebox{\pbox@TP}}%               This typesets the saved minipage inside
  \smallskip%                                           A small space to separate the area from
  \par
  }
```

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.

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

Default stripe overlap for avoiding 'gaps' in color gradients.

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

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

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

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

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

Default position of the 'middle' color of a double gradient.

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

The following are used internally when making color gradients.

```
\newcounter{stripe@TP}

\newcounter{stripes@TP}
```

```
\newcommand{\firstgradprogression@TP}{1}

\newcommand{\secondgradprogression@TP}{1}

\newcounter{gradprogression@TP}

\newcommand{\gradmidpoint@TP}{.5}
```

Special versions of `\mkfactor` which apply gradient progressions.

```
\newcommand{\mkgradfirstfactor@TP}{\mkgradfactor@TP\firstgradprogression@TP}

\newcommand{\mkgradsecondfactor@TP}{\mkgradfactor@TP\secondgradprogression@TP}

\newcommand{\mkgradfactor@TP}[3]% Calculate a factor modified by a 'progression' parameter.
{%
  \mkfactor{#2}{#3}%                                Calculate the unmodified factor.
  \setcounter{gradprogression@TP}{#1}%              Factor definition may contain a calc-express
  \ifthenelse{\value{gradprogression@TP}=1}{}%      Progression value 1 is neutral.
  {%
    \ifthenelse{\value{gradprogression@TP}<0}%      'Negative' progression?
    {% Yes.
      \@tempcnta-\value{gradprogression@TP}\relax% Complement progression wrt 0.
      \mkfactor{#2}{1-1pt*\real{#2}}%               Complement factor definition wrt 1pt.
      }
    {\@tempcnta\value{gradprogression@TP}\relax}%  No; Use progression as given.
    \whiledo{\the\@tempcnta>1}%                     Calculate (factor definition)^(progression).
    {\advance\@tempcnta by -1\relax\mkfactor{#2}{\real{#2}*\real{#2}}}%
    \ifthenelse{\value{gradprogression@TP}<0}%      'Negative' progression?
    {% Yes.
      \mkfactor{#2}{1-1pt*\real{#2}}%               Complement result wrt 1pt.
      }
    {}%
  }%
}
```

`\vgradrule` `\vgradrule[<stripes>][<startmodel>]{<startcolor>}[<endmodel>]{<endcolor>}[<raise>]{<widt` 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>]`...

```
\newcommand{\vgradrule}[1][]% Pick up first optional argument: [<stripes>]
{%
  \let\firstgradprogression@TP=\rulefirstgradprogression%  Use progression parameter for rules.
  \ifthenelse{\equal{#1}{}}%                               First optional argument given?
  {\setcounter{stripes@TP}{\rulestripes}}%                 No; use default value.
  {\setcounter{stripes@TP}{#1}}%                           Yes.
  \vgradrule@TP%                                           Pick up [<startmodel>]{<startcolor>}
}

\newcommand{\vgradrule@TP}[2][]% Pick up next pair of arguments: [<startmodel>]{<startcolor>}.
{%
  \ifthenelse{\equal{#1}{}}%               <startmodel> given?
  {\replacecolor{startcolor@TP}{#2}}%      No; <startcolor> is a color name.
  {\definecolor{startcolor@TP}{#1}{#2}}%   Yes; {<startmodel>}{<startcolor>} is a color definit
  \@vgradrule@TP%                          Pick up [<endmodel>]{<endcolor>}.
}

\newcommand{\@vgradrule@TP}[2][]% Pick up next pair of arguments: [<endmodel>]{<endcolor>}.
{%
  \ifthenelse{\equal{#1}{}}
  {\replacecolor{endcolor@TP}{#2}}
  {\definecolor{endcolor@TP}{#1}{#2}}%
  \@@vgradrule@TP%                         Pick up rule arguments and proceed.
}
```

Helper command for making **one** stripe. Can be overladed for making histograms.

```
\newcommand{\hstripe@TP}[4]%
{\hbox{{\setcolor@TP{stripecolor@TP}\rule{#2}{#3}}}#4}
```

Main part of `\vgradrule`.

```
\newcommand{\@@vgradrule@TP}[3][0pt]%
{%
  \ifthenelse{\value{stripes@TP}<2}%       A 'pathological case' which can happen in animations
                                    %      requested, a division by zero error would be produce
  {\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][0pt]{%
```

\dblvgradrule `\dblvgradrule[<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{\dblvgradrule}[1][]% Pick up first optional argument: [<midpoint>]
826 {%
```

```
  \let\firstgradprogression@TP=\rulefirstgradprogression%  Use progression parameters for rules
  \let\secondgradprogression@TP=\rulesecondgradprogression
  \ifthenelse{\equal{#1}{}}%                               First optional argument given?
  {\let\gradmidpoint@TP=\rulegradmidpoint}%                No; use default value.
  {\def\gradmidpoint@TP{#1}}%                              Yes.
  \dblvgradrule@TP%                                        Pick up second optional argument.
}

\newcommand{\dblvgradrule@TP}[1][]% Pick up second optional argument: [<stripes>]
{%
  \ifthenelse{\equal{#1}{}}
  {\setcounter{stripes@TP}{\rulestripes}}
  {\setcounter{stripes@TP}{#1}}%
  \@dblvgradrule@TP
}

\newcommand{\@dblvgradrule@TP}[2][]% Pick up next pair of arguments: [<startmodel>]{<startcolor
{%
  \ifthenelse{\equal{#1}{}}
  {\replacecolor{startcolor@TP}{#2}}
  {\definecolor{startcolor@TP}{#1}{#2}}%
  \@@dblvgradrule@TP
}

\newcommand{\@@dblvgradrule@TP}[2][]% Pick up next pair of arguments: [<midmodel>]{<midcolor>}.
{%
  \ifthenelse{\equal{#1}{}}
  {\replacecolor{midcolor@TP}{#2}}
  {\definecolor{midcolor@TP}{#1}{#2}}%
  \@@@dblvgradrule@TP
}

\newcommand{\@@@dblvgradrule@TP}[2][]% Pick up next pair of arguments: [<endmodel>]{<endcolor>}
{%
  \ifthenelse{\equal{#1}{}}
  {\replacecolor{endcolor@TP}{#2}}
  {\definecolor{endcolor@TP}{#1}{#2}}%
  \@@@@dblvgradrule@TP%                          Pick up rule arguments and proceed.
}

\newcommand{\@@@@dblvgradrule@TP}[3][0pt]%             Main part of \dblvgradrule.
{%
  \ifthenelse{\value{stripes@TP}<2}%                   The gradient code is not equipped for m
  {\mbox{{\setcolor@TP{midcolor@TP}\rule[#1]{#2}{#3}}}}% In this case, just produce a rule colo
  {%
    \raisebox{#1}%                                     Evaluate the <raise> argument of the ru
    {%
      \vbox%                                           A vbox with \offinterlineskip allows to
      {%                                               vertically.
        \offinterlineskip
```

```
        \setcounter{stripe@TP}{0}%                      Initialize the number of the current st
        \setcounter{tmpcnta@TP}%                        Number of the 'middle' stripe.
        {\value{stripes@TP}*\real{\gradmidpoint@TP}}%
        \whiledo{\value{stripe@TP}<\value{tmpcnta@TP}}% Produce the upper part of the gradient.
        {%
          \mkgradfirstfactor@TP{\tmp@TP}%               Make the weight for \colorbetween, base
          {\value{stripe@TP}/\value{tmpcnta@TP}}%       current stripe and the first gradient p
          \colorbetween[\tmp@TP]{stripecolor@TP}{midcolor@TP}{startcolor@TP}% Calculate stripe
          \hstripe@TP{\tmp@TP}%                                               Make stripe with
          {#2}{(#3)/\value{stripes@TP}+\stripeoverlap}{\kern-\stripeoverlap}% This is the upper
          \stepcounter{stripe@TP}%
          }%
        \stepcounter{tmpcnta@TP}%
        \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}% Only one stripe left to produce?
        {\hstripe@TP{1}{#2}{(#3)/\value{stripes@TP}}{}}%    Just produce one stripe colored wit
        {%
          \whiledo{\value{stripe@TP}<\value{stripes@TP}}%   Produce the lower part of the gradi
          {%
            \mkgradsecondfactor@TP{\tmp@TP}%                Make the weight for \colorbetween.
            {(\value{stripe@TP}-\value{tmpcnta@TP}+1)/(\value{stripes@TP}-\value{tmpcnta@TP})}%
            \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{midcolor@TP}% Calculate stripe
            \stepcounter{stripe@TP}%
            \ifthenelse{\value{stripe@TP}=\value{stripes@TP}}%     Last stripe?
            {\hstripe@TP{\tmp@TP}{#2}{(#3)/\value{stripes@TP}}{}}% Yes; make stripe w/o overlap
            {%                                                     No; add kern to make stripes
              \hstripe@TP{\tmp@TP}{#2}{(#3)/\value{stripes@TP}+\stripeoverlap}{\kern-\stripeove
              }%
            }% matches \whiledo{\value{stripe@TP}<\value{stripes@TP}}%
          }% matches second argument of \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}%
        }% matches \vbox{%
      }% matches \raisebox{#1}{%
    }% matches second argument of \ifthenelse{\value{stripes@TP}<2}%
  }% matches \newcommand{\@@@@dblvgradrule@TP}[3][0pt]%
```

\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.

```
\newcommand{\hgradrule}[1][]% Pick up first optional argument: [<stripes>].
{%
  \let\firstgradprogression@TP=\rulefirstgradprogression%  Use progression parameter for rules.
  \ifthenelse{\equal{#1}{}}%                               First optional argument given?
  {\setcounter{stripes@TP}{\rulestripes}}%                 No; use default value.
  {\setcounter{stripes@TP}{#1}}%                           Yes.
  \hgradrule@TP%                                           Pick up [<startmodel>]{<startcolor>}
}

\newcommand{\hgradrule@TP}[2][]% Pick up next pair of arguments: [<startmodel>]{<startcolor>}.
```

```
{%
  \ifthenelse{\equal{#1}{}}
  {\replacecolor{startcolor@TP}{#2}}
  {\definecolor{startcolor@TP}{#1}{#2}}%
  \@hgradrule@TP
}

\newcommand{\@hgradrule@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}
  {\replacecolor{endcolor@TP}{#2}}
  {\definecolor{endcolor@TP}{#1}{#2}}%
  \@@hgradrule@TP%                          Pick up rule arguments and proceed.
}

\newcommand{\vstripe@TP}[4]%              Helper command for making _one_ stripe. Can be overla
{{\setcolor@TP{stripecolor@TP}\rule{#2}{#3}#4}}

\newcommand{\@@hgradrule@TP}[3][0pt]%     Main part of \hgradrule.
{%
  \ifthenelse{\value{stripes@TP}<2}%      A 'pathological case' which can happen in animations
                                    %     requested, a division by zero error would be produce
  {\mbox{{\setcolor@TP{startcolor@TP}\rule[#1]{#2}{#3}}}}% In this case, just produce a colored
  {%
    \raisebox{#1}%                        Evaluate the <raise> argument of the rule.
    {%
      \setcounter{stripe@TP}{0}%          Initialize the number of the current stripe.
      \whiledo{\value{stripe@TP}<\value{stripes@TP}}
      {%
        \mkgradfirstfactor@TP{\tmp@TP}%   Make the weight for \colorbetween, based on the numb
        {\value{stripe@TP}/(\value{stripes@TP}-1)}% and the first gradient progression.
        \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{startcolor@TP}% Calculate stripe co
        \stepcounter{stripe@TP}%
        \ifthenelse{\value{stripe@TP}=\value{stripes@TP}}%     Last stripe?
        {\vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}}{#3}{}}% Yes; make stripe w/o overlap.
        {%                                                     No; add a kern to make stripes o
          \vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}+\stripeoverlap}{#3}{\kern-\stripeoverlap
          }%
        }% matches \whiledo{\value{stripe@TP}<\value{stripes@TP}}{%
      }% matches \raisebox{#1}{%
    }% matches second argument of \ifthenelse{\value{stripes@TP}<2}%
  }% matches \newcommand{\@@hgradrule@TP}[3][0pt]{%
```

`\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.

```
\newcommand{\dblhgradrule}[1][]% Pick up first optional argument: [<midpoint>].
{%
  \let\firstgradprogression@TP=\rulefirstgradprogression%  Use progression parameters for rules
  \let\secondgradprogression@TP=\rulesecondgradprogression
  \ifthenelse{\equal{#1}{}}{\let\gradmidpoint@TP=\rulegradmidpoint}{\def\gradmidpoint@TP{#1}}%
  \dblhgradrule@TP
}

\newcommand{\dblhgradrule@TP}[1][]% Pick up second optional argument: [<stripes>].
{%
  \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}
  \@dblhgradrule@TP
}

\newcommand{\@dblhgradrule@TP}[2][]% Pick up next pair of arguments: [<startmodel>]{<startcolor
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
  \@@dblhgradrule@TP
}

\newcommand{\@@dblhgradrule@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{midcolor@TP}{#2}}{\definecolor{midcolor@TP}{#1}{#2}}%
  \@@@dblhgradrule@TP
}

\newcommand{\@@@dblhgradrule@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}
  {\replacecolor{endcolor@TP}{#2}}
  {\definecolor{endcolor@TP}{#1}{#2}}%
  \@@@@dblhgradrule@TP%                          Pick up rule arguments and proceed.
}

\newcommand{\@@@@dblhgradrule@TP}[3][0pt]%             Main part of \dblhgradrule.
{%
  \ifthenelse{\value{stripes@TP}<2}%                   The gradient code is not equipped for ma
  {\mbox{{\setcolor@TP{midcolor@TP}\rule[#1]{#2}{#3}}}}% In this case, just produce a rule colo
  {%
    \raisebox{#1}%                                     Evaluate the <raise> argument of the rul
    {%
      \setcounter{stripe@TP}{0}%                       Initialize the number of the current str
      \setcounter{tmpcnta@TP}%                          Number of the 'middle' stripe.
      {\value{stripes@TP}*\real{\gradmidpoint@TP}}%
      \whiledo{\value{stripe@TP}<\value{tmpcnta@TP}}%   Produce the left part of the gradient.
      {%
        \mkgradfirstfactor@TP{\tmp@TP}%                 Make the weight for \colorbetween, base
        {\value{stripe@TP}/\value{tmpcnta@TP}}%         current stripe and the first gradient p
        \colorbetween[\tmp@TP]{stripecolor@TP}{midcolor@TP}{startcolor@TP}% Calculate stripe co
        \vstripe@TP{\tmp@TP}%                                               Make stripe with ov
```

```
      {(#2)/\value{stripes@TP}+\stripeoverlap}{#3}{\kern-\stripeoverlap}% This is the left pa
      \stepcounter{stripe@TP}%
      }%
    \stepcounter{tmpcnta@TP}%
    \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}% Only one stripe left to produce?
    {\vstripe@TP{1}{(#2)/\value{stripes@TP}}{#3}{}}%    Just produce one stripe colored with
    {%
      \whiledo{\value{stripe@TP}<\value{stripes@TP}}%   Produce the right part of the gradien
      {%
        \mkgradsecondfactor@TP{\tmp@TP}%                Make the weight for \colorbetween.
        {(\value{stripe@TP}-\value{tmpcnta@TP})/(\value{stripes@TP}-\value{tmpcnta@TP}-1)}%
        \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{midcolor@TP}% Calculate stripe co
        \stepcounter{stripe@TP}%
        \ifthenelse{\value{stripe@TP}=\value{stripes@TP}}%     Last stripe?
        {\vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}}{#3}{}}% Yes; make stripe w/o overlap.
        {%                                                     Add kern to make stripes overl
          \vstripe@TP{\tmp@TP}{(#2)/\value{stripes@TP}+\stripeoverlap}{#3}{\kern-\stripeoverl
          }%
        }% matches \whiledo{\value{stripe@TP}<\value{stripes@TP}}{%
      }% matches second argument of \ifthenelse{\value{stripes@TP}=\value{tmpcnta@TP}}%
    }% matches \raisebox{#1}{%
  }% matches second argument of \ifthenelse{\value{stripes@TP}<2}%
}% matches \newcommand{\@@@@dblhgradrule@TP}[3][0pt]{%
```

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

```
% PDF:
% \def\clipbox{\@ifnextchar[{\clipbox@}{\clipbox@[0pt]}}
% \def\clipbox@[#1]#2{%
%   \begingroup
%     \setlength{\@tempdima}{#1}%
%     \setbox\@tempboxa=\hbox{%
%       \color@begingroup
%       #2%
%       \color@endgroup
%     }%
%     \leavevmode\hbox to \wd\@tempboxa{%
%       \@ifundefined{dimexpr}{%
%         \@defbp\x{-\@tempdima}%
%         \@tempdimc=\dp\@tempboxa
%         \advance\@tempdimc by \@tempdima
%         \@defbp\y{-\@tempdimc}%
%         \@tempdimc=\wd\@tempboxa
%         \advance\@tempdimc by 2\@tempdima
%         \@defbp\w{\@tempdimc}%
%         \@tempdimc=\dp\@tempboxa
%         \advance\@tempdimc by \ht\@tempboxa
%         \advance\@tempdimc by 2\@tempdima
%         \@defbp\h{\@tempdimc}%
%         \pdfliteral{%
%           q % gsave
```

```
%           \x\space\y\space\w\space\h\space re % rectangle
%           W n% make clip-path
%         }%
%       }{% e-TeX
%          \pdfliteral{%
%           q % gsave
%           \@dimtobp{-\@tempdima} % x
%           \@dimtobp{-\dp\@tempboxa-\@tempdima} % y
%           \@dimtobp{\wd\@tempboxa+2\@tempdima} % width
%           \@dimtobp{\dp\@tempboxa+\ht\@tempboxa+2\@tempdima} % height
%           re % rectangle
%           W n% make clip-path
%         }%
%       }%
%       \rlap{\unhbox\@tempboxa}%
%       \pdfliteral{%
%         Q% grestore
%       }%
%       \hss
%     }%
%   \endgroup
% }
% \begingroup\expandafter\expandafter\expandafter\endgroup
% \expandafter\ifx\csname dimexpr\endcsname\relax
%   \def\@defbp#1#2{%
%     \setlength{\@tempdimb}{#2}%
%     \setlength{\@tempdimb}{.99626401\@tempdimb}%
%     \edef#1{\strip@pt\@tempdimb}%
%   }
% \else
%   \def\@dimtobp#1{%
%     \strip@pt\dimexpr.99626401\dimexpr#1\relax\relax
%   }
% \fi
%
% PS:
% \def\clipbox{\@ifnextchar[{\clipbox@}{\clipbox@[0pt]}}
% \def\clipbox@[#1]#2{%
%   \begingroup
%     \setlength{\@tempdima}{#1}%
%     \setbox\@tempboxa=\hbox{%
%       \color@begingroup
%       #2%
%       \color@endgroup
%     }%
%     \leavevmode\hbox to \wd\@tempboxa{%
%       \@ifundefined{dimexpr}{%
%         \@defpt\x{-\@tempdima}%
%         \@tempdimc=\ht\@tempboxa
%         \advance\@tempdimc by \@tempdima
```

```
%         \@defpt\y{-\@tempdimc}%
%         \@tempdimc=\wd\@tempboxa
%         \advance\@tempdimc by 2\@tempdima
%         \@defpt\w{\@tempdimc}%
%         \@tempdimc=\dp\@tempboxa
%         \advance\@tempdimc by \ht\@tempboxa
%         \advance\@tempdimc by 2\@tempdima
%         \@defpt\h{\@tempdimc}%
%         \special{ps:%
%           gsave %
%           SDict begin %
%           \x\space PTtoDVIPS \y\space PTtoDVIPS rmoveto %
%           currentpoint %
%           \w\space PTtoDVIPS \h\space PTtoDVIPS rectclip %
%           end%
%         }%
%       }{% e-TeX
%          \special{ps:%
%           gsave %
%           SDict begin %
%           \@dimtopt{-\@tempdima} PTtoDVIPS % x
%           \@dimtopt{-\ht\@tempboxa-\@tempdima} PTtoDVIPS % y
%           rmoveto currentpoint %
%           \@dimtopt{\wd\@tempboxa+2\@tempdima} PTtoDVIPS % width
%           \@dimtopt{\dp\@tempboxa+\ht\@tempboxa+2\@tempdima} %
%           PTtoDVIPS % height
%           rectclip %
%           end%
%         }%
%       }%
%       \rlap{\unhbox\@tempboxa}%
%       \special{ps:grestore}%
%       \hss
%     }%
%   \endgroup
% }
% \special{!%
%   /PTtoDVIPS{72.27 div Resolution mul}def%
%   % rectclip is a level 2 feature
%   /rectclip where{pop}{%
%     /rectclip{%
%       newpath %
%       4 2 roll moveto %
%       exch dup 0 rlineto %
%       exch 0 exch rlineto %
%       neg 0 rlineto %
%       closepath %
%       clip %
%       newpath %
%     }bind def%
```

```
% }%
% ifelse%
% }
% \begingroup\expandafter\expandafter\expandafter\endgroup
% \expandafter\ifx\csname dimexpr\endcsname\relax
%   \def\@defpt#1#2{%
%     \setlength{\@tempdimb}{#2}%
%     \edef#1{\strip@pt\@tempdimb}%
%   }
% \else
%   \def\@dimtopt#1{%
%     \strip@pt\dimexpr#1\relax
%   }
% \fi


\ifpdf
  \providecommand{\clipbox}[2][\z@]
  {%
    \setlength{\@tempdima}{#1}%
    \setbox\@tempboxa=
    \hbox{\kern\@tempdima\vbox{\offinterlineskip\kern\@tempdima\hbox{#2}\kern\@tempdima}\kern\@
    \pdfxform\@tempboxa
    \leavevmode
    \hbox
    {%
      \kern-\@tempdima
      \vbox{\offinterlineskip\kern-\@tempdima\hbox{\pdfrefxform\pdflastxform}\kern-\@tempdima}%
      \kern-\@tempdima
      }%
    }
 \else
  \providecommand{\clipbox}[2][\z@]{\leavevmode\hbox{#2}}
\fi
```

`\dgradslope` stuff.

```
\newcommand{\dgradslope}{1,1}

\newcounter{dgradhslope@TP}

\newcounter{dgradvslope@TP}

\def\setdgradslope(#1,#2){\setcounter{dgradhslope@TP}{#1}\setcounter{dgradvslope@TP}{#2}}
```

`\dgradrule` stuff.

```
\newcommand{\dgradrule}[1][]
{%
  \ifthenelse{\equal{#1}{}}{\expandafter\setdgradslope\expandafter(\dgradslope)}{\setdgradslope
  \dgradrule@TP
}

```

```
\newcommand{\dgradrule@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
  \@dgradrule@TP
}

\newcommand{\@dgradrule@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
  \@@dgradrule@TP
}

\newcommand{\dstripewd@TP}{.7pt}

\newcommand{\@@dgradrule@TP}[3][0pt]
{%
  \raisebox{#1}
  {%
    \thicklines
    \setcounter{stripe@TP}{0}%
    \setcounter{tmpcnta@TP}{1*\ratio{#3}{\dstripewd@TP}}%
    \setcounter{stripes@TP}{1*\ratio{#3}{\dstripewd@TP}+1*\ratio{#2}{\dstripewd@TP}}%
    \@xarg\c@dgradhslope@TP\@yarg\c@dgradvslope@TP
    \makebox[\dstripewd@TP]
    {%
      \whiledo{\value{stripe@TP}<\value{tmpcnta@TP}}
      {%
        \mkfactor{\tmp@TP}{\value{stripe@TP}/(\value{stripes@TP}-1)}%
        \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{startcolor@TP}%
        \setlength{\@linelen}{\dstripewd@TP*\value{stripe@TP}}%
        \raisebox{\dstripewd@TP*(\value{tmpcnta@TP}-\value{stripe@TP})}
        {\makebox[0pt]{{\setcolor@TP{stripecolor@TP}\hbox to 0pt{\@sline\hss}}}}%
        \stepcounter{stripe@TP}%
        }%
      }%
    \whiledo{\value{stripe@TP}<\value{stripes@TP}}
    {%
      \mkfactor{\tmp@TP}{\value{stripe@TP}/(\value{stripes@TP}-1)}%
      \colorbetween[\tmp@TP]{stripecolor@TP}{endcolor@TP}{startcolor@TP}%
      \makebox[\dstripewd@TP]{{\setcolor@TP{stripecolor@TP}.}}%
      \stepcounter{stripe@TP}%
      }%
    }%
  }%
```

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.

```
\newcommand{\vgradbox}[1][]% Pick up first optional argument: [<stripes>].
{%
  \let\firstgradprogression@TP=\rulefirstgradprogression%  Use progression parameter for rules.
  \ifthenelse{\equal{#1}{}}%                               First optional argument given?
  {\setcounter{stripes@TP}{\rulestripes}}%                 No; use default value.
  {\setcounter{stripes@TP}{#1}}%                           Yes.
  \vgradbox@TP%                                            Pick up remaining optional arguments
}

\newcommand{\vgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
  \@vgradbox@TP
}

\newcommand{\@vgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
  \@@gradbox@TP\@@vgradrule@TP%                    Make a generic background box with vertical
}

\newcommand{\@@gradbox@TP}[2]%                     Generic background box.
{%
  \leavevmode%                                     Make box behave like \mbox.
  \setbox\tempbox@TP
  \hbox{\kern\fboxsep{\set@color#2}\kern\fboxsep}% An \hbox containing <contents> plus addition
  \rlap%                                           Underlay box background with rule command pa
  {%
    #1[-\fboxsep-\dp\tempbox@TP]%                  Box depth augmented by 'white' space.
    {\wd\tempbox@TP}%                              Space on the sides has already been added.
    {\ht\tempbox@TP+\dp\tempbox@TP+2\fboxsep}}%    Total height.
  \box\tempbox@TP%                                 Overlay box contents.
  }%
```

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

```
\newcommand{\hgradbox}[1][]
{%
  \let\secondgradprogression@TP=\rulesecondgradprogression
```

```
  \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}
  \hgradbox@TP
}

\newcommand{\hgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
  \@hgradbox@TP
}

\newcommand{\@hgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
  \@@gradbox@TP\@@hgradrule@TP
}
```

\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).

```
\newcommand{\dblvgradbox}[1][]
{%
  \let\firstgradprogression@TP=\rulefirstgradprogression% Use progression parameters for rules
  \let\secondgradprogression@TP=\rulesecondgradprogression
  \ifthenelse{\equal{#1}{}}{\let\gradmidpoint@TP=\rulegradmidpoint}{\def\gradmidpoint@TP{#1}}%
  \dblvgradbox@TP
}

\newcommand{\dblvgradbox@TP}[1][]
{%
  \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}
  \@dblvgradbox@TP
}

\newcommand{\@dblvgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
  \@@dblvgradbox@TP
}

\newcommand{\@@dblvgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{midcolor@TP}{#2}}{\definecolor{midcolor@TP}{#1}{#2}}%
  \@@@dblvgradbox@TP
}

\newcommand{\@@@dblvgradbox@TP}[2][]
{%
```

```
  \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
  \@@gradbox@TP\@@@@dblvgradrule@TP
}
```

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

```
\newcommand{\dblhgradbox}[1][]
{%
  \let\firstgradprogression@TP=\rulefirstgradprogression
  \let\secondgradprogression@TP=\rulesecondgradprogression
  \ifthenelse{\equal{#1}{}}{\let\gradmidpoint@TP=\rulegradmidpoint}{\def\gradmidpoint@TP{#1}}%
  \dblhgradbox@TP
}

\newcommand{\dblhgradbox@TP}[1][]
{%
  \ifthenelse{\equal{#1}{}}{\setcounter{stripes@TP}{\rulestripes}}{\setcounter{stripes@TP}{#1}}
  \@dblhgradbox@TP
}

\newcommand{\@dblhgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{startcolor@TP}{#2}}{\definecolor{startcolor@TP}{#1}{#
  \@@dblhgradbox@TP
}

\newcommand{\@@dblhgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{midcolor@TP}{#2}}{\definecolor{midcolor@TP}{#1}{#2}}%
  \@@@dblhgradbox@TP
}

\newcommand{\@@@dblhgradbox@TP}[2][]
{%
  \ifthenelse{\equal{#1}{}}{\replacecolor{endcolor@TP}{#2}}{\definecolor{endcolor@TP}{#1}{#2}}%
  \@@gradbox@TP\@@@@dblhgradrule@TP
}
```

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 `\backgroundstyle` wasn't used at all (for instance, a plain area of color pagecolor 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 `\backgroundstyle` wasn't used at all (for instance, a plain area of color pagecolor 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: hpanels, autopanels, toppanelcolor, bottompanelcolor, leftpanelcolor, rightpanelcolor, toppanelcolordef, bottompanelcolordef, leftpanelcolordef, rightpanelcolordef, toppanelheight, bottompanelheight, leftpanelwidth, rightpanelwidth.

vgradient
: Vertical gradient. The page background is constructed using the `\vgradrule` 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.
  Parameters used: stripes, firstgradprogression, startcolor, startcolordef, endcolor, endcolordef, hpanels, autopanels, toppanelcolor, bottompanelcolor, leftpanelcolor, rightpanelcolor, toppanelcolordef, bottompanelcolordef, leftpanelcolordef, rightpanelcolordef, toppanelheight, bottompanelheight, leftpanelwidth, rightpanelwidth.

hgradient
: Horizontal gradient. The page background is constructed using the `\hgradrule` command. See the description of vgradient concerning panels.
  Parameters used: stripes, firstgradprogression, startcolor, startcolordef, endcolor, endcolordef, hpanels, autopanels, toppanelcolor, bottompanelcolor, leftpanelcolor, rightpanelcolor, toppanelcolordef, bottompanelcolordef, leftpanelcolordef, rightpanelcolordef, toppanelheight, bottompanelheight, leftpanelwidth, rightpanelwidth.

doublevgradient — 'Double' vertical gradient. The page background is constructed using the `\dblvgradrule` command. See the description of vgradient concerning panels. Parameters used: stripes, gradmidpoint, firstgradprogression, secondgradprogression, startcolor, startcolordef, midcolor, midcolordef, endcolor, endcolordef, hpanels, autopanels, toppanelcolor, 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, autopanels, toppanelcolor, 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
`stripes=<n>`	Set the `<stripes>` parameter of gradient rules to `<n>`. Default: `\bgndstripes` Used by: vgradient, hgradient, doublevgradient, doublehgradient
`gradmidpoint=<r>`	Set the `<midpoint>` parameter of double gradient rules to `<r>`. Default: `\bgndgradmidpoint` Used by: doublevgradient, doublehgradient
`firstgradprogression=<i>`	Set the first gradient progression of gradient rules to `<i>`. Default: `\bgndfirstgradprogression` Used by: vgradient, hgradient, doublevgradient, doublehgradient
`secondgradprogression=<i>`	Set the second gradient progression of double gradient rules to `<i>`. Default: `\bgndsecondgradprogression` Used by: doublevgradient, doublehgradient
`startcolor=<c>`	Set the `<startcolor>` parameter of gradient rules to `<c>`. Default: If neither startcolor nor startcolordef is given, the color bgndstartcolor is used as `<startcolor>`. Used by: vgradient, hgradient, doublevgradient, doublehgradient Overwrites: startcolordef
`startcolordef={<cm>}{<cd>}`	Set the `<startcolor>` 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 startcolor nor startcolordef is given, the color bgndstartcolor is used as `<startcolor>`. Used by: vgradient, hgradient, doublevgradient, doublehgradient Overwrites: startcolor
`endcolor=<c>`	Set the `<endcolor>` parameter of gradient rules to `<c>`. Default: If neither endcolor nor endcolordef is given, the color bgndendcolor is used as `<endcolor>`. Used by: vgradient, hgradient, doublevgradient, doublehgradient Overwrites: endcolordef

`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

`autopanels=<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 autopanels=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¿. 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 autopanels parameter. If autopanels=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 autopanels=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¿. 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

```
\newcommand{\backgroundstyle}[2][]
{%
  \replacecolor{startcolor@TP}{bgndstartcolor}%                          Initialize the intern
  \replacecolor{midcolor@TP}{bgndmidcolor}%                              of parameters to thei
  \replacecolor{endcolor@TP}{bgndendcolor}%
  \replacecolor{bgndtoppanelcolor@TP}{toppanelcolor}%
  \replacecolor{bgndbottompanelcolor@TP}{bottompanelcolor}%
  \replacecolor{bgndleftpanelcolor@TP}{leftpanelcolor}%
  \replacecolor{bgndrightpanelcolor@TP}{rightpanelcolor}%
  \let\firstgradprogression@TP=\bgndfirstgradprogression
  \let\secondgradprogression@TP=\bgndsecondgradprogression%
  \setcounter{stripes@TP}{\bgndstripes}%
  \let\gradmidpoint@TP=\bgndgradmidpoint
  \let\bgndtoppanelheight@TP=\empty%                 The panel dimensions depend on other param
  \let\bgndbottompanelheight@TP=\empty%              set directly, defaults are calculated afte
  \let\bgndleftpanelwidth@TP=\empty%                 parameters.
  \let\bgndrightpanelwidth@TP=\empty
  \let\hpanelsvalue@TP=\empty
  \setboolean{autopanels@TP}{true}%
  \csname set#2bgnd@TP\endcsname{#1}%                Execute the style-specific command which d
  }
```

Background-specific default values.

Default number of stripes for gradient page backgrounds.

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

Default position of the 'middle' color of a double gradient.

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

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

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

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

```
\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 autopanels 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.

```
\newcommand{\bgndtoppanelheight}{0pt}

\newcommand{\bgndbottompanelheight}{0pt}

\newcommand{\bgndleftpanelwidth}{0pt}

\newcommand{\bgndrightpanelwidth}{0pt}
```

Internal names for parameter values.

```
\newcommand{\bgndtoppanelheight@TP}{0pt}

\newcommand{\bgndtoppanelwidth@TP}{0pt}

\newcommand{\bgndbottompanelheight@TP}{0pt}

\newcommand{\bgndbottompanelwidth@TP}{0pt}

\newcommand{\bgndleftpanelheight@TP}{0pt}

\newcommand{\bgndleftpanelwidth@TP}{0pt}

\newcommand{\bgndrightpanelheight@TP}{0pt}

\newcommand{\bgndrightpanelwidth@TP}{0pt}

\newboolean{hpanels@TP}

\newboolean{autopanels@TP}
```

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

```
\define@key{bgnd@TP}{stripes}{\setcounter{stripes@TP}{#1}}


\define@key{bgnd@TP}{startcolor}{\replacecolor{startcolor@TP}{#1}}

\define@key{bgnd@TP}{startcolordef}{\definecolor{startcolor@TP}#1}


\define@key{bgnd@TP}{midcolor}{\replacecolor{midcolor@TP}{#1}}

\define@key{bgnd@TP}{midcolordef}{\definecolor{midcolor@TP}#1}

```

```
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}{autopanels}[true]{\setboolean{autopanels@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).

```
\newbox\bgndbox@TP
\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.

```
\newcommand{\mkpanels@TP}[1]%
{%
  \ifthenelse{\boolean{hpanels@TP}}%                                    'horizontal' panels?
  {% Yes. Vertically align top panel, center area with left and right panels, and bottom panel.
    \vbox%                                                              A vbox with \offinterli
    {%                                                                  horizontal panels with
      \offinterlineskip
      \ifthenelse{\lengthtest{\bgndtoppanelheight@TP=0pt}}%             Should top panel be cre
      {}% No.
      {%
        \hbox{{%
            \setcolor@TP{bgndtoppanelcolor@TP}%
            \rule{\bgndtoppanelwidth@TP}{\bgndtoppanelheight@TP}%       Make horizontal colored
            }}%
        }%
      \hbox%                                                            Make 'background center
      {%
        \ifthenelse{\lengthtest{\bgndleftpanelwidth@TP=0pt}}%           Should left panel be cr
        {}% No.
        {{%
            \setcolor@TP{bgndleftpanelcolor@TP}%
            \rule{\bgndleftpanelwidth@TP}{\bgndleftpanelheight@TP}%     Make vertical colored a
            }}%
        #1%                                                             Make main background ob
        {\bgndtoppanelwidth@TP-\bgndleftpanelwidth@TP-\bgndrightpanelwidth@TP}% Calculate remai
        {\bgndleftpanelheight@TP}%
        \ifthenelse{\lengthtest{\bgndrightpanelwidth@TP=0pt}}%          Should right panel be c
        {}% No.
        {{%
            \setcolor@TP{bgndrightpanelcolor@TP}%
            \rule{\bgndrightpanelwidth@TP}{\bgndrightpanelheight@TP}%   Make vertical colored a
            }}%
        }% matches \hbox{%
      \ifthenelse{\lengthtest{\bgndbottompanelheight@TP=0pt}}%          Should bottom panel be
      {}% No.
      {%
        \hbox
        {{%
            \setcolor@TP{bgndbottompanelcolor@TP}%
            \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. ¡hpanels¿

gives the background style dependent default of the hpanels option.

```
\newcommand{\initpanels@TP}[1]%
{%
  \ifx\hpanelsvalue@TP\empty%                          Has the hpanels parameter been given?
    \setboolean{hpanels@TP}{#1}%                       No; use default.
   \else
    \setboolean{hpanels@TP}{\hpanelsvalue@TP}%         Yes; use parameter setting.
  \fi
  \ifthenelse{\boolean{hpanels@TP}}%                   'horizontal' panel direction?
  {% Yes. Horizontal panels are 'outer', vertical panels are 'inner'.
    \let\bgndtoppanelwidth@TP=\TPpagewidth%            Full width for horizontal panels.
    \let\bgndbottompanelwidth@TP=\TPpagewidth%
    \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}%     Has the top panel height been set?
    {% No. Guess default.
      \ifx\toppanelcontents@TP\empty%                  Is the panel defined?
        \mklength@TP{\bgndtoppanelheight@TP}{\bgndtoppanelheight}% Use background-specific defa
       \else
        \ifthenelse{\boolean{autopanels@TP}}%          Calculate panel dimensions?
        {% Yes.
          \calcvdimen@TP{\bgndtoppanelheight@TP}{\bgndtoppanelwidth@TP}% Measure the height of
          {\toppanelcontents@TP}%
          \let\toppanelheight=\bgndtoppanelheight@TP%  Overwrite panel settings.
          \let\toppanelwidth=\bgndtoppanelwidth@TP%
          \def\toppanelshift{0pt}%                     Top panel spans the whole upper screen.
          }
        {% No
          \mklength@TP{\bgndtoppanelheight@TP}{\toppanelheight}%   Use panel-specific default.
          }%
      \fi
      }% matches \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}
    {% Yes.
      \let\toppanelheight=\bgndtoppanelheight@TP%       Overwrite panel settings  - use user-sup
    }
    \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}%         Has the bottom panel height been
    {% No. Guess default.
      \ifx\bottompanelcontents@TP\empty%                      Is the bottom panel defined?
        \mklength@TP{\bgndbottompanelheight@TP}{\bgndbottompanelheight}% Use background-specifi
       \else
        \ifthenelse{\boolean{autopanels@TP}}%                 Calculate panel dimensions?
        {% Yes.
          \calcvdimen@TP{\bgndbottompanelheight@TP}{\bgndbottompanelwidth@TP}% Measure the heig
          {\bottompanelcontents@TP}%
          \let\bottompanelheight=\bgndbottompanelheight@TP%   Overwrite panel settings.
          \let\bottompanelwidth=\bgndbottompanelwidth@TP%
          \def\bottompanelshift{0pt}%                         Bottom panel spans the whole uppe
          }
        {% No
          \mklength@TP{\bgndbottompanelheight@TP}{\bottompanelheight}%   Use panel-specific def
          }%
```

```
  \fi
  }% matches \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}
{% Yes.
  \let\bottompanelheight=\bgndbottompanelheight@TP%   Overwrite panel settings - use user-s
}
\mklength@TP{\bgndleftpanelheight@TP}%                Calculate remaining space in the cent
{\TPpageheight-\bgndtoppanelheight@TP-\bgndbottompanelheight@TP}%
\let\bgndrightpanelheight@TP=\bgndleftpanelheight@TP% Height of left and right panels is eq
\ifthenelse{\equal{\bgndleftpanelwidth@TP}{}}%        Has the left panel width been set?
{% No. Guess default.
  \ifx\leftpanelcontents@TP\empty%                    Is the panel defined?
    \mklength@TP{\bgndleftpanelwidth@TP}{\bgndleftpanelwidth}% Use background-specific defa
   \else
    \ifthenelse{\boolean{autopanels@TP}}%             Calculate panel dimensions?
    {% Yes.
      \calchdimen@TP{\bgndleftpanelwidth@TP}{\bgndleftpanelheight@TP}% Measure the ‘optimal
      {\leftpanelcontents@TP}%
      \let\leftpanelheight=\bgndleftpanelheight@TP%   Overwrite panel settings.
      \let\leftpanelwidth=\bgndleftpanelwidth@TP%
      \let\leftpanelraise=\bgndbottompanelheight@TP%  Left panel is raised above bottom pan
      }
    {% No
      \mklength@TP{\bgndleftpanelwidth@TP}{\leftpanelwidth}%   Use panel-specific default.
      }%
  \fi
  }% matches \ifthenelse{\equal{\bgndleftpanelwidth@TP}{}}
{% Yes.
  \let\leftpanelwidth=\bgndleftpanelwidth@TP%         Overwrite panel settings - use user-s
}
\ifthenelse{\equal{\bgndrightpanelwidth@TP}{}}%       Has the right panel width been set?
{% No. Guess default.
  \ifx\rightpanelcontents@TP\empty%                   Is the panel defined?
    \mklength@TP{\bgndrightpanelwidth@TP}{\bgndrightpanelwidth}% Use background-specific de
   \else
    \ifthenelse{\boolean{autopanels@TP}}%             Calculate panel dimensions?
    {% Yes.
      \calchdimen@TP{\bgndrightpanelwidth@TP}{\bgndrightpanelheight@TP}% Measure ‘optimal’
      {\rightpanelcontents@TP}%
      \let\rightpanelheight=\bgndrightpanelheight@TP% Overwrite panel settings.
      \let\rightpanelwidth=\bgndrightpanelwidth@TP%
      \let\rightpanelraise=\bgndbottompanelheight@TP% Right panel is raised above bottom pa
      }
    {% No
      \mklength@TP{\bgndrightpanelwidth@TP}{\rightpanelwidth}%   Use panel-specific default
      }%
  \fi
  }% matches \ifthenelse{\equal{\bgndrightpanelwidth@TP}{}}
{% Yes.
  \let\rightpanelwidth=\bgndrightpanelwidth@TP%       Overwrite panel settings - use user-s
}
```

```
  }% matches \ifthenelse{\boolean{hpanels@TP}}
{% No. Vertical panels are ‘outer’, horizontal panels are ‘inner’.
  \let\bgndleftpanelheight@TP=\TPpageheight%          Full height for vertical panels.
  \let\bgndrightpanelheight@TP=\TPpageheight%
  \ifthenelse{\equal{\bgndleftpanelwidth@TP}{}}%      Has the left panel width been set?
  {% No. Guess default.
    \ifx\leftpanelcontents@TP\empty%                  Is the panel defined?
      \mklength@TP{\bgndleftpanelwidth@TP}{\bgndleftpanelwidth}% Use background-specific defa
     \else
      \ifthenelse{\boolean{autopanels@TP}}%           Calculate panel dimensions?
      {% Yes.
        \calchdimen@TP{\bgndleftpanelwidth@TP}{\bgndleftpanelheight@TP}% Measure the ‘optimal
        {\leftpanelcontents@TP}%
        \let\leftpanelheight=\bgndleftpanelheight@TP% Overwrite panel settings.
        \let\leftpanelwidth=\bgndleftpanelwidth@TP%
        \def\leftpanelraise{0pt}%                     Left panel spans the whole left part of
        }
      {% No
        \mklength@TP{\bgndleftpanelwidth@TP}{\leftpanelwidth}%   Use panel-specific default.
        }%
    \fi
    }% matches \ifthenelse{\equal{\bgndleftpanelwidth@TP}{}}
  {% Yes.
    \let\leftpanelwidth=\bgndleftpanelwidth@TP%       Overwrite panel settings - use user-sup
  }
  \ifthenelse{\equal{\bgndrightpanelwidth@TP}{}}%       Has the right panel width been set?
  {% No. Guess default.
    \ifx\rightpanelcontents@TP\empty%                   Is the panel defined?
      \mklength@TP{\bgndrightpanelwidth@TP}{\bgndrightpanelwidth}% Use background-specific de
     \else
      \ifthenelse{\boolean{autopanels@TP}}%             Calculate panel dimensions?
      {% Yes.
        \calchdimen@TP{\bgndrightpanelwidth@TP}{\bgndrightpanelheight@TP}% Measure ‘optimal’
        {\rightpanelcontents@TP}%
        \let\rightpanelheight=\bgndrightpanelheight@TP% Overwrite panel settings.
        \let\rightpanelwidth=\bgndrightpanelwidth@TP%
        \def\rightpanelraise{0pt}%                      Right panel spans the whole left part
        }
      {% No
        \mklength@TP{\bgndrightpanelwidth@TP}{\rightpanelwidth}%   Use panel-specific default
        }%
    \fi
    }% matches \ifthenelse{\equal{\bgndrightpanelwidth@TP}{}}
  {% Yes.
    \let\rightpanelwidth=\bgndrightpanelwidth@TP%       Overwrite panel settings - use user-s
  }
  \mklength@TP{\bgndtoppanelwidth@TP}%                Calculate remaining space in the center
  {\TPpagewidth-\bgndleftpanelwidth@TP-\bgndrightpanelwidth@TP}%
  \let\bgndbottompanelwidth@TP=\bgndtoppanelwidth@TP% Width of top and bottom panels is equal
  \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}%      Has the top panel height been set?
```

```
    {% No. Guess default.
      \ifx\toppanelcontents@TP\empty%                   Is the panel defined?
        \mklength@TP{\bgndtoppanelheight@TP}{\bgndtoppanelheight}% Use background-specific defa
       \else
        \ifthenelse{\boolean{autopanels@TP}}%           Calculate panel dimensions?
        {% Yes.
          \calcvdimen@TP{\bgndtoppanelheight@TP}{\bgndtoppanelwidth@TP}% Measure the height of
          {\toppanelcontents@TP}%
          \let\toppanelheight=\bgndtoppanelheight@TP%   Overwrite panel settings.
          \let\toppanelwidth=\bgndtoppanelwidth@TP%
          \let\toppanelshift=\bgndleftpanelwidth@TP%    Shift top panel to the right of left pa
          }
        {% No
          \mklength@TP{\bgndtoppanelheight@TP}{\toppanelheight}%   Use panel-specific default.
          }%
      \fi
      }% matches \ifthenelse{\equal{\bgndtoppanelheight@TP}{}}
    {% Yes.
      \let\toppanelheight=\bgndtoppanelheight@TP%       Overwrite panel settings - use user-sup
    }
    \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}%       Has the bottom panel height been se
    {% No. Guess default.
      \ifx\bottompanelcontents@TP\empty%                    Is the panel defined?
        \mklength@TP{\bgndbottompanelheight@TP}{\bgndbottompanelheight}% Use background-specifi
       \else
        \ifthenelse{\boolean{autopanels@TP}}%               Calculate panel dimensions?
        {% Yes.
          \calcvdimen@TP{\bgndbottompanelheight@TP}{\bgndbottompanelwidth@TP}% Measure the heig
          {\bottompanelcontents@TP}%
          \let\bottompanelheight=\bgndbottompanelheight@TP% Overwrite panel settings.
          \let\bottompanelwidth=\bgndbottompanelwidth@TP%
          \let\bottompanelshift=\bgndleftpanelwidth@TP%     Shift bottom panel to the right of
          }
        {% No
          \mklength@TP{\bgndbottompanelheight@TP}{\bottompanelheight}%   Use panel-specific def
          }%
      \fi
      }% matches \ifthenelse{\equal{\bgndbottompanelheight@TP}{}}
    {% Yes.
      \let\bottompanelheight=\bgndbottompanelheight@TP%     Overwrite panel settings - use user
    }
    }% matches second argument of \ifthenelse{\boolean{hpanels@TP}}
  }% matches \newcommand{\initpanels@TP}[1]{
```

Make an 'invisible' rule.

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

Implementations of individual background styles.

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

```

```
\newcommand{\setplainbgnd@TP}[1]%             Implementation of the background style ‘‘plain’’.
{%
  \setkeys{bgnd@TP}{#1}%                      Evaluate parameters.
  \initpanels@TP{true}%                       Initialize panel parameters.
  \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\phantomrule@TP}}% Make panels only.
  }%


\newcommand{\setvgradientbgnd@TP}[1]%         Implementation of the background style ‘‘vgradien
{%
  \setkeys{bgnd@TP}{#1}%                      Evaluate parameters.
  \initpanels@TP{false}%                      Initialize panel parameters.
  \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@@vgradrule@TP}}% Make background box.
  }

\newcommand{\sethgradientbgnd@TP}[1]%         Implementation of the background style ‘‘hgradien
{%
  \setkeys{bgnd@TP}{#1}%                      Evaluate parameters.
  \initpanels@TP{true}%                       Initialize panel parameters.
  \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@@hgradrule@TP}}% Make background box.
  }


\newcommand{\setdoublevgradientbgnd@TP}[1]%   Implementation of the background style ‘‘doublevg
{%
  \setkeys{bgnd@TP}{#1}%                      Evaluate parameters.
  \initpanels@TP{false}%                      Initialize panel parameters.
  \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@@@@dblvgradrule@TP}}% Make background box.
  }


\newcommand{\setdoublehgradientbgnd@TP}[1]%   Implementation of the background style ‘‘doublehg
{%
  \setkeys{bgnd@TP}{#1}%                      Evaluate parameters.
  \initpanels@TP{true}%                       Initialize panel parameters.
  \global\setbox\bgndbox@TP=\hbox{\mkpanels@TP{\@@@@dblhgradrule@TP}}% Make background box.
  }
```

`\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).

```
\providecommand{\hpagecolor}[2][]
{%
  \ifthenelse{\equal{#1}{}}
  {\colorbetween{ppendcolor}{#2}{white}\backgroundstyle[startcolor=#2,endcolor=ppendcolor]{hgra
  {\backgroundstyle[startcolor=#1,endcolor=#2]{hgradient}}%
  }
```

`\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).

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

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).

```
\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.

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

Dimensions of bottom panel.

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

Dimensions of left panel.

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

Dimensions of right panel.

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

Some internal panel parameters.

Storage for panel contents.

```
\newcommand*{\toppanelcontents@TP}{}
\newcommand*{\bottompanelcontents@TP}{}
\newcommand*{\leftpanelcontents@TP}{}
\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\rightpanelconte
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 {%
```

```
  \@ifundefined{toppanelcontents@TP@#1}{\global\let\toppanelcontents@TP\empty}
  {%
    \expandafter\global\expandafter\let\expandafter\toppanelcontents@TP\csname toppanelcontents
    \expandafter\global\expandafter\let\expandafter\toppanelwidth@TP\csname toppanelwidth@TP@#1
    \expandafter\global\expandafter\let\expandafter\toppanelheight@TP\csname toppanelheight@TP@
    \expandafter\global\expandafter\let\expandafter\toppanelshift@TP\csname toppanelshift@TP@#1
    }%
  \@ifundefined{bottompanelcontents@TP@#1}{\global\let\bottompanelcontents@TP\empty}
  {%
    \expandafter\global\expandafter\let\expandafter\bottompanelcontents@TP\csname bottompanelco
    \expandafter\global\expandafter\let\expandafter\bottompanelwidth@TP\csname bottompanelwidth
    \expandafter\global\expandafter\let\expandafter\bottompanelheight@TP\csname bottompanelheig
    \expandafter\global\expandafter\let\expandafter\bottompanelshift@TP\csname bottompanelshift
    }%
  \@ifundefined{leftpanelcontents@TP@#1}{\global\let\leftpanelcontents@TP\empty}
  {%
    \expandafter\global\expandafter\let\expandafter\leftpanelcontents@TP\csname leftpanelconten
    \expandafter\global\expandafter\let\expandafter\leftpanelwidth@TP\csname leftpanelwidth@TP@
    \expandafter\global\expandafter\let\expandafter\leftpanelheight@TP\csname leftpanelheight@T
    \expandafter\global\expandafter\let\expandafter\leftpanelraise@TP\csname leftpanelraise@TP@
    }%
  \@ifundefined{rightpanelcontents@TP@#1}{\global\let\rightpanelcontents@TP\empty}
  {%
    \expandafter\global\expandafter\let\expandafter\rightpanelcontents@TP\csname rightpanelcont
    \expandafter\global\expandafter\let\expandafter\rightpanelwidth@TP\csname rightpanelwidth@T
    \expandafter\global\expandafter\let\expandafter\rightpanelheight@TP\csname rightpanelheight
    \expandafter\global\expandafter\let\expandafter\rightpanelraise@TP\csname rightpanelraise@T
    }%
}
```

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.

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

Remove any contents which could mess up the box analysis.

```
\newcommand{\panel@sanitize@TP}
{%
  \let\hyperlink=\@secondoftwo
  \let\Acrobatmenu=\@secondoftwo
}
```

Calculate height of 'horizontal' panel.

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

```
  \mklength@TP{#1}{\ht\tempbox@TP+\dp\tempbox@TP}%                                % Meas
}
```

Calculate 'optimal' width of 'vertical panel.

```
\newcommand{\calchdimen@TP}[3]
{%
  \optwidth@TP{#1}{#2-\panelmargin*2}%              Calculate 'optimal' width of a parbox. Pane
  {.5\textwidth-\panelmargin*2}{\panelalignment#3}% .5\textwidth is the hardcoded absolute maxi
  \mklength@TP{#1}{#1+\panelmargin*2}%
  }
```

User-configurable: Which 'resolution' should be used when searching for 'best' width?

```
\newcommand{\optwidthsteps}{100}
```

User-configurable: Which badness should be tolerated as 'perfect' (stopping the search for a better one).

```
\newcommand{\optwidthlinetolerance}{200}
```

Internal parameter: Badness of the parbox currently under consideration.

```
\let\maxbadness@TP=\@tempcnta
```

A hook to disable some commands which would be in the way while measuring things.

```
\def\optwidthdisablecommands@TP
{%
  \let\Hy@colorlink\@firstofone
  \let\Hy@endcolorlink\relax
  }
```

Calculate 'best' width of a parbox. The current algorithm will set the textual contents into parboxes of increasing width, starting from 0pt 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.

```
\newcommand{\optwidth@TP}[4]
{%
  \setcounter{tmpcnta@TP}{0}%                                  Initialize 'probe counter' for b
  \let\best@cnt@TP=\empty%                                     Initialize number of best 'probe
  \def\bestbadness@TP{1000000}%                                Initialize badness of best 'prob
  \setboolean{carryon@TP}{true}%                               Flag for breaking out of loop.
  \setlength{\tempdimb@TP}{#2}%                                Store maximal box height.
  \whiledo
  {\value{tmpcnta@TP}<\optwidthsteps\and\boolean{carryon@TP}}% Probes done or break of loop?
  {%
    \stepcounter{tmpcnta@TP}%                                  Start next probe.
    \setbox\@tempboxa=\hbox%                                   The trick with vbox/lastbox is t
    {%                                                         produced by \parbox 'immediately
      \optwidthdisablecommands@TP%                             Turn off some nasties not needed
      \parbox[b]{(#3)/\optwidthsteps*\value{tmpcnta@TP}}%      Make the next parbox.
```

```
      {\hfuzz\maxdimen\hbadness\@M\relax#4}%
      \global\setbox\tempbox@TP=\lastbox%                        ... and assign \tempbox@TP to it
      }%
    \setlength{\tempdima@TP}{\ht\tempbox@TP+\dp\tempbox@TP}%   Measure total height.
    \ifthenelse{\lengthtest{\tempdima@TP>\tempdimb@TP}}%       If it exceeds the maximum height
    {}%                                                        acceptable anyway.
    {%
      \calcmaxbadness@TP{\maxbadness@TP}{\tempbox@TP}%         Calculate 'worst badness' of any
      \ifthenelse{\not\maxbadness@TP>\optwidthlinetolerance}%  Below Threshold?
      {% Yes. Accept this width.
        \edef\best@cnt@TP{\thetmpcnta@TP}%                     Store this probe number.
        \setboolean{carryon@TP}{false}%                        Break loop.
        }
      {% No. Carry on.
        \ifthenelse{\maxbadness@TP<\bestbadness@TP}%           Below lowest badness found so fa
        {% Yes. Store probe number.
          \edef\bestbadness@TP{\number\maxbadness@TP}%         Store badness value.
          \edef\best@cnt@TP{\thetmpcnta@TP}%                   Store probe number.
          }
        {}% No. Try next probe.
        }% matches second argument of \ifthenelse{\not\maxbadness@TP>\optwidthlinetolerance}%
      }% matches second argument of \ifthenelse{\lengthtest{\tempdima@TP>\tempdimb@TP}}%
    }% matches \whiledo{\value{tmpcnta@TP}<\optwidthsteps\and\boolean{carryon@TP}}{%
  \ifx\best@cnt@TP\empty%                                      Was _any_ badness below the init
    \mklength@TP{#1}{#3}% No; return max width.
   \else
    \mklength@TP{#1}{(#3)/\optwidthsteps*\best@cnt@TP}% Yes; return width of best probe.
  \fi
  }% matches \newcommand{\optwidth@TP}[4]{%
```

Calculate maximal badness of any hbox occurring in a vbox.

```
\newcommand{\calcmaxbadness@TP}[2]
{%
  \let\@resultcnt@TP=#1%           Here we store the result.
  \global\@resultcnt@TP=\z@\relax% Just in case no hbox occurs...
  \setlength{\@tempdima}{\wd#2}%   This is the width to which every hbox is stretched for findi
  \setbox\@tempboxa=\vbox%         A dummy vbox for recursively analysing the vbox contents usi
  {%
    \hfuzz\maxdimen\hbadness\@M
    \unvbox#2%                     'free' the contents of the vbox.
    \measureboxes@TP%              Analyse 'tail to head' using \lastbox.
    }%
  }
```

Recursively analyse vertical list using `\lastbox`, to find maximum badness of any contained hbox.

```
\newcommand{\measureboxes@TP}%
{%
  \unskip\unpenalty\unkern%    This is a kluge for TeX, because there is no certain way of find
  \unskip\unpenalty\unkern%    penalty, glue or kern on the vertical list. \lastpenalty will gi
  \unskip\unpenalty\unkern%    value of 0 might mean there was none or there was one of value 0
```

```
  \unskip\unpenalty\unkern%    This is different in eTeX. I might make a switch to a smarter so
  \unskip\unpenalty\unkern
  \unskip\unpenalty\unkern
  \unskip\unpenalty\unkern
  \unskip\unpenalty\unkern
  \unskip\unpenalty\unkern
  \unskip\unpenalty\unkern
  \setbox\@tempboxa=\lastbox% Grab last box.
  \ifhbox\@tempboxa%          Was this an hbox?
    \setbox0=\hb@xt@\@tempdima{\unhbox\@tempboxa}% Yes. Reformat with given width.
    \ifnum\badness>\@resultcnt@TP%                 Badness larger than largest recorded badness
      \global\@resultcnt@TP=\badness%              Yes. Memorize.
    \fi
    \expandafter\measureboxes@TP%                  Recursive call.
   \else
    \ifvbox\@tempboxa%                                        Was this a vbox?
      \expandafter\expandafter\expandafter\measureboxes@TP% Ignore, but execute recursive call.
    \fi
  \fi
  }
```

3.5.3 Actually typeset panels

```
\newcommand{\mk@toppanel@TP}% top panel
{%
  \ifx\toppanelcontents@TP\empty%           top panel specified?
   \else%                                   Yes; create box with appropriate dimensions, backgr
    \@mk@panel@TP{\toppanelwidth}{\toppanelheight}{toppaneltextcolor}{\toppanelcontents@TP}%
  \fi
  }

\newcommand{\mk@bottompanel@TP}% bottom panel
{%
  \ifx\bottompanelcontents@TP\empty%        bottom panel specified?
   \else%                                   Yes; create box with appropriate dimensions, backgr
    \@mk@panel@TP{\bottompanelwidth}{\bottompanelheight}{bottompaneltextcolor}{\bottompanelcont
  \fi
  }

\newcommand{\mk@leftpanel@TP}% left panel
{%
  \ifx\leftpanelcontents@TP\empty%          left panel specified?
   \else%                                   Yes; create box with appropriate dimensions, backgr
    \@mk@panel@TP{\leftpanelwidth}{\leftpanelheight}{leftpaneltextcolor}{\leftpanelcontents@TP}
  \fi
  }

\newcommand{\mk@rightpanel@TP}% right panel
{%
  \ifx\rightpanelcontents@TP\empty%         right panel specified?
```

```
   \else%                                    Yes; create box with appropriate dimensions, text c
    \@mk@panel@TP{\rightpanelwidth}{\rightpanelheight}{rightpaneltextcolor}{\rightpanelcontents
  \fi
  }


\newcommand{\@mk@panel@TP}[2]% Generate ‘standard’ parbox parameters for panels.
{%
  \@@mk@panel@TP{#1}{[#2-\panelmargin*2][s]}%
  }


\@ifclassloaded{powersem}
{%
  \newcommand{\panelalignment}{\sem@ptsize{\slide@ptsize}\large\normalsize}%
}
{%
  \newcommand{\panelalignment}%            Justification for panels. This setting allows a cert
  {\setlength{\rightskip}{0pt plus 20pt}}% ‘right-raggedness’. Leave empty for standard parbox
}

% Make a panel box.
\newcommand{\@@mk@panel@TP}[4]
{%
  \vbox
  {%
    \offinterlineskip
    \kern\panelmargin%  Top margin.
    \hbox
    {{%
        \ifthenelse{\boolean{instepwise@TP}}%  Inside \stepwise, colors mat have been dimmed, l
        {\usecolorset{stwcolors}}{}%           Restore them, just in case.
        \color{#3}%                            Set panel text color.
        \kern\panelmargin%                     Left margin.
        \parbox[b]#2{#1-\panelmargin*2}%       The parbox with the main panel contents.
        {%
          \normalfont
          \panelalignment#4%
          \hrule\@height\z@%           The hrule makes sure the total height of this box can b
          }%
        \kern\panelmargin%             Right margin.
        }}%
    \kern\panelmargin%                 Bottom margin.
    }%
  }
```

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.

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

Width of button frame.

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

Horizontal displacement of button shadow.

```
\newcommand{\buttonshadowhshift}{.3\fboxsep}
```

Vertical displacement of button shadow.

```
\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).

```
\newcommand{\button}[1][0pt]%      Collect first optional parameter.
{%
  \mklength@TP\bt@width@TP{#1}%    Store optional argument.
  \button@TP
}

\newcommand{\button@TP}[1][0pt]%   Collect second optional parameter.
{%
  \mklength@TP\bt@height@TP{#1}%   Store optional argument.
  \@button@TP
}

\newcommand{\@button@TP}[1][0pt]%  Collect third optional parameter.
{%
  \mklength@TP\bt@depth@TP{#1}%    Store optional argument.
  \@@button@TP
}

\newcommand{\@@button@TP}[3][c]%   Collect fourth optional and two mandatory parameters and pro
{%
  \ifthenelse{\lengthtest{\bt@width@TP=0pt}}%       <width> given?
  {\mklength@TP{\bt@width@TP}{\widthof{#3}}}%       No. Calculate width of <text>.
  {\mklength@TP{\bt@width@TP}{\bt@width@TP-\buttonsep*2-\buttonrule*2}}% Yes. Calculate area le
```

```
  \ifthenelse{\lengthtest{\bt@height@TP=0pt}}%      <height> given?
  {\mklength@TP{\bt@height@TP}{\heightof{#3}}}%     No. Calculate height of <text>.
  {\mklength@TP{\bt@height@TP}{\bt@height@TP-\buttonsep-\buttonrule}}% Yes. Calculate area left
  \ifthenelse{\lengthtest{\bt@depth@TP=0pt}}%       <depth> given?
  {\mklength@TP{\bt@depth@TP}{\depthof{#3}}}%       No. Calculate depth of <text>.
  {\mklength@TP{\bt@depth@TP}{\bt@depth@TP-\buttonsep-\buttonrule}}% Yes. Calculate area left f
  \leavevmode% \rlap creates a 'raw' hbox. So we get into horizontal mode.
  \rlap%       Make shadow.
  {%
    \hspace*{\buttonshadowhshift}%  Horizontal displacement.
    \raisebox{\buttonshadowvshift}% Vertical displacement.
    {%
      {% Inner group for correct color handling.
        \setcolor@TP{buttonshadowcolor}% Button shadow color.
        \rule%                          Create colored rectangular patch of appropriate dimens
        [-\bt@depth@TP-\buttonsep-\buttonrule]
        {\bt@width@TP+\buttonsep*2+\buttonrule*2}
        {\bt@height@TP+\bt@depth@TP+\buttonsep*2+\buttonrule*2}%
        }% matches inner group
      }% matches \raisebox{\buttonshadowvshift}{%
    }% matches \rlap{%
  \edef\o@fboxrule@TP{\the\fboxrule}%    Preserve original definitions of \fbox parameters.
  \edef\o@fboxsep@TP{\the\fboxsep}%
  #2%                                    Execute <navcommand>.
  {%
    \setlength{\fboxrule}{\buttonrule}%        Set \fbox parameters for button frame.
    \setlength{\fboxsep}{\buttonsep}%
    \fcolorbox{buttonframecolor}{buttoncolor}% Create button frame with the right colors.
    {%
      \makebox[\bt@width@TP][#1]%              Create box of correct width to contain <text>.
      {%
        \raisebox{0pt}[\bt@height@TP][\bt@depth@TP]% Create box of correct height and depth.
        {%
          \setlength{\fboxrule}{\o@fboxrule@TP}\setlength{\fboxsep}{\o@fboxsep@TP}% Restore fbo
          \setcolor@TP{buttontextcolor}#3%           Produce <text>.
          }% matches \raisebox{0pt}[\bt@height@TP][\bt@depth@TP]{%
        }% matches \makebox[\bt@width@TP][#1]{%
      }% matches \fcolorbox{buttonframecolor}{buttoncolor}{%
    }% matches argument of <navcommand>.
  }% matches \newcommand{\@@button@TP}[3][c]{%
```

Some predefined buttons.
Size of predefined button symbols.

```
\newcommand{\buttonsymbolsize}{\footnotesize}
```

Define predefined button symbols.

```
\@ifpackageloaded{amssymb}%     AMS symbols available?
{% Yes. Use 'black' symbols.
  \newcommand{\buttonleftarrowsymbol}{{\buttonsymbolsize\boldmath\origmath{\blacktriangleleft}}
  \newcommand{\buttonrightarrowsymbol}{{\buttonsymbolsize\boldmath\origmath{\blacktriangleright
```

```
  \newcommand{\buttonbackarrowsymbol}{{\buttonsymbolsize\boldmath\origmath{\vartriangleleft}}}%
}
{% No. Use replacements from standard set.
  \newcommand{\buttonleftarrowsymbol}{{\buttonsymbolsize\boldmath\origmath{\lhd}}}%       `left
  \newcommand{\buttonrightarrowsymbol}{{\buttonsymbolsize\boldmath\origmath{\rhd}}}%      `righ
  \newcommand{\buttonbackarrowsymbol}{{\buttonsymbolsize\boldmath\origmath{\leftarrow}}}% `back
}
```

Width of predefined buttons.

```
\newcommand{\stdbuttonwidth}{\widthof{\buttonrightarrowsymbol\buttonrightarrowsymbol\buttonrigh
```

Customizeable default: How to find the number of the current page?

```
\newcommand{\currentpagevalue}{\value{page}}
```

Predefined button: last subpage of previous page.

```
\newcommand{\backpagebutton}[1][\stdbuttonwidth]
{%
  \button[#1]{\setcounter{tmpcnta@TP}{\currentpagevalue-1}\hyperlink{page.\thetmpcnta@TP}}
  {\buttonleftarrowsymbol\buttonleftarrowsymbol}%
  }
```

Predefined button: previous step.

```
\newcommand{\backstepbutton}[1][\stdbuttonwidth]
{%
  \button[#1]{\Acrobatmenu{PrevPage}}
  {\buttonleftarrowsymbol}%
  }
```

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

```
\newcommand{\gobackbutton}[1][\stdbuttonwidth]
{%
  \button[#1]{\Acrobatmenu{GoBack}}
  {\buttonbackarrowsymbol}%
  }
```

Predefined button: next step.

```
\newcommand{\nextstepbutton}[1][\stdbuttonwidth]
{%
  \button[#1]{\Acrobatmenu{NextPage}}
  {\buttonrightarrowsymbol}%
  }
```

Predefined button: first subpage of next page.

```
\newcommand{\nextpagebutton}[1][\stdbuttonwidth]
{%
  \button[#1]{\setcounter{tmpcnta@TP}{\currentpagevalue+1}\hyperlink{firstpage.\thetmpcnta@TP}}
  {\buttonrightarrowsymbol\buttonrightarrowsymbol}%
  }
```

Predefined button: last subpage of next page.

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

```
  {\buttonrightarrowsymbol\buttonrightarrowsymbol\buttonrightarrowsymbol}%
  }
```

Predefined button: toggle fullscreen mode.

```
\newcommand{\fullscreenbutton}[1][\stdbuttonwidth]
{%
  \button[#1]{\Acrobatmenu{FullScreen}}
  {\buttonrightarrowsymbol\buttonleftarrowsymbol}%
  }
```

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 transition with hyperref's `\hypersetup`. The command is defined to a noop in case hyperref is not loaded.

```
\ifthenelse{\boolean{display}}%            Are dynamic features enabled?
{% Yes.
  \newcommand{\pagetransition}[1]%         Definition for preamble.
  {%
    \@ifpackageloaded{hyperref}%           Can we use \hypersetup?
    {% Yes.
      \hypersetup{pdfpagetransition={#1}}% Set page transition with \hypersetup.
      }
    {% No. In this case, we can't set the page transition.
      \PackageWarning{texpower}
      {Package hyperref not loaded.\MessageBreak Page transition not set}%
      }%
    }% matches \newcommand{\pagetransition}[1]{%
  \AtBeginDocument%                        Definition for document body.
  {%
    \@ifpackageloaded{hyperref}%           Can we use \hypersetup?
    {% Yes.
      \hypersetup{pdfpagetransition={}}% Make pagetransition setting (consistently) local to gr
      \renewcommand{\pagetransition}[1]{\hypersetup{pdfpagetransition={#1}}}%
      }
    {% No. Disable page transitions.
      \PackageWarning{texpower}
      {Package hyperref not loaded.\MessageBreak Page transitions disabled}%
      \let\pagetransition=\@gobble
      }%
    }% matches \AtBeginDocument{%
  }% matches \ifthenelse{\boolean{display}}{%
{\let\pagetransition=\@gobble}% No. Disable page transitions.
```

Some standard page transitions.

```
\newcommand{\pageTransitionSplitHO}{\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 "nineminutes" 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`.

```
\newcommand{\infinitepageduration}{}%          Default: No page duration setting on every page.

\ifthenelse{\boolean{nineminutes@TP}}%         If the option nineminutes is given, every page w
{\renewcommand{\infinitepageduration}{546}}{}% gets a setting of 546 seconds (9.1 minutes).
```

Implementation of `\pageDuration`.

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

```
      {Package hyperref not loaded.\MessageBreak Page duration not set}%
      }%
    }% matches \newcommand{\pageDuration}[1]{%
  \AtBeginDocument%                        Make sure this also works if texpower is loaded befo
  {%
    \@ifpackageloaded{hyperref}%                    Can we use \hypersetup?
    {% Yes.
      \edef\next{\noexpand\hypersetup{pdfpageduration={\infinitepageduration}}}%
      \next%                                        Set default page duration.
      \renewcommand{\pageDuration}[1]{\hypersetup{pdfpageduration={#1}}}% Set page duration wit
      }%
    {% No. In this case, we can't set the page duration.
      \PackageWarning{texpower}
      {Package hyperref not loaded.\MessageBreak Page duration disabled}%
      \let\pageDuration=\@gobble
      }%
    }% matches \AtBeginDocument{%
  }% matches \ifthenelse{\boolean{display}}{%
{\let\pageDuration=\@gobble}% No. Disable page duration.
```

`\stopAdvancing` undoes any setting effected by `\pageDuration`.

```
\newcommand{\stopAdvancing}
{\edef\next{\noexpand\pageDuration{\infinitepageduration}}\next}% Reset to default page duratio
```

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`.

```
\AtBeginDocument{\shipoutinit@TP}%
```

Replace `\shipout` by own definition.

```
\newcommand*{\shipoutinit@TP}
{%
  \let\o@shipout@TP=\shipout% Memorize previous definition.
```

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

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

```
\newcommand{\shipout@TP}
{%
  \afterassignment\shipout@test@TP
  \global\setbox\@cclv= %
  }%
```

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.

```
\newcommand{\shipout@test@TP}
{%
  \ifvoid\@cclv\relax%                Is the box assigned yet?
    \aftergroup\shipout@output@TP%    No; defer execution of \shipout@output@TP.
   \else
    \shipout@output@TP%               Yes; execute \shipout@output@TP immediately.
  \fi%
  }%
```

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

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

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.

```
\newcommand{\AtShipout}[1]
{\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.

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

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

```
\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.

```
\@ifpackageloaded{hyperref}%                              Is hyperref loaded?
{% Yes. Prepare hook.
  \newcommand{\do@insert@firstanchor@TP}%                 This is executed on the first subpage o
  {\hypertarget{firstpage.\number\currentpagevalue}{}}% Create target.
  }
{\newcommand{\do@insert@firstanchor@TP}{}}% No. Leave hook empty.
```

This is executed on every subpage which is **not** the first one.

```
\newcommand{\dont@insert@firstanchor@TP}{}%
```

This is the hook itself which is executed on every page.

```
\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.

```
\newcommand{\shipout@hook@TP}
{%
  \filterpage@TP%                  Filter out whatsits on duplicate pages.
  \setbox\@cclv=%                  Create 'real' page box (which is later shipped out).
  \hbox{%
    \set@typeset@protect
    \raise\ht\@cclv\rlap%                          Place background box.
    {%
      \vtop to \TPpageheight
      {%
        \offinterlineskip
        \hrule\@height\z@\relax
        \kern -1truein\relax
        \kern -\voffset\relax
        \moveleft 1truein\hbox{\kern-\hoffset\copy\bgndbox@TP}%
        \vss
        \kern 1truein\relax
        \kern\voffset\relax
        }%
      }%
    \raise\ht\@cclv\rlap%                          Place 'vertical' panels.
    {%
      \vtop to \TPpageheight
      {%
        \offinterlineskip
```

```
      \hrule\@height\z@\relax
      \kern -1truein\relax
      \kern -\voffset\relax
      \moveleft 1truein\hb@xt@\TPpagewidth
      {%
        \kern-\hoffset\raise\leftpanelraise\hbox{\mk@leftpanel@TP}%
        \hfill
        \raise\rightpanelraise\hbox{\mk@rightpanel@TP}%
        \kern\hoffset
        }%
      \vss
      \kern 1truein\relax
      \kern\voffset\relax
      }%
    }%
  \raise\ht\@cclv\rlap%                          Place 'horizontal' panels.
  {%
    \vtop to \TPpageheight
    {%
      \offinterlineskip
      \hrule\@height\z@\relax
      \kern -1truein\relax
      \kern -\voffset\relax
      \moveleft 1truein\hbox{\kern-\hoffset\kern\toppanelshift\mk@toppanel@TP}%
      \vfill
      \moveleft 1truein\hbox{\kern-\hoffset\kern\bottompanelshift\mk@bottompanel@TP}%
      \kern 1truein\relax
      \kern\voffset\relax
      }%
    }%
  \raise\ht\@cclv\rlap
  {%
    \let\hyper@@anchor\o@hyper@@anchor%       Reactivate hyper anchors.
    \insert@firstanchor@TP%                   Execute hook for creating target firstpage.<n>
    \ifshippingduplicate%                     Will further subpages follow?
      \global\let\insert@firstanchor@TP=\dont@insert@firstanchor@TP% Deactivate hook for next
     \else
      \global\let\insert@firstanchor@TP=\do@insert@firstanchor@TP%   Reactivate hook for next
    \fi
    \at@shipout@once@TP%                      Execute code stored away by \AtShipout.
    }%
  \global\let\at@shipout@once@TP=\empty%      Clear \AtShipout container.
  \box\@cclv
  }%
}
```

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.

```
\def\colorcorrections@TP{\relax}%          The 'counter stack' of color correction codes, reset

\AtBeginDocument%                          Make sure to catch color.sty even if it's loaded aft
{%
  \ifthenelse{\boolean{fixcolorstack@TP}}% Was the fixcolorstack option given?
  {%
    \@ifundefined{VTeXversion}%            Using vtex?
    {% No. Go on.
      \ifthenelse{\boolean{pdf}}%          Producing pdf with pdftex?
      {% Yes. Color stack fixing unnecessary.
        \PackageWarning{texpower}
        {pdftex doesn't need color stack correction.\MessageBreak Option fixcolorstack disabled
        }
      {% No. We need to add corrections for the color stack...
        \@ifpackageloaded{color}%          ... but only if color.sty is loaded at all.
        {%
          \expandafter\def\expandafter\shipout@hook@TP\expandafter% Extend \shipout hook ...
          {\shipout@hook@TP\clearcolorcorrections@TP}%              ... by a command to clear c
                                  %
          \newcommand{\clearcolorcorrections@TP}%                        The counter stack is
          {%                                                             ... on all 'final' s
            \ifshippingduplicate
             \else
              \gdef\colorcorrections@TP{\relax}%
            \fi
          }
```

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.

```
          \expandafter\def\expandafter\o@set@color@TP\expandafter%
          {%
```

```
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\addpushtoccor@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\addpushtoccor@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}
```

```
        {}% No changes needed if color.sty is not loaded.
        }% matches second argument of \ifthenelse{\boolean{pdf}}
      }% matches \@ifundefined{VTeXversion}
    {% Yes. Color stack fixing unnecessary.
      \PackageWarning{texpower}
      {vtex doesn't need color stack correction.\MessageBreak Option fixcolorstack disabled.}%
      }%
    }% matches \ifthenelse{\boolean{fixcolorstack@TP}}
  {}%
  }% 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.

```
\newif\ifshippingduplicate
```

The command used to shipout a duplicate page.

```
\providecommand{\TPeject}{\newpage}
```

Some internal registers to store away things.

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

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

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

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

Original definition of output routine.

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

Save current page contents to a box.

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

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

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

```
\def\savepausecounters@TP{%
  \begingroup
    \def\@elt##1{\global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
    \xdef\restorepausecounters@TP{\cl@@ckptpause@TP}%
  \endgroup
  }
```

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

```
\newcommand{\pausesafecounter}[1]%
{\expandafter\def\expandafter\cl@@ckptpause@TP\expandafter{\cl@@ckptpause@TP\@elt{#1}}}
```

Setting `\pausesafecounter` for common classes

```
\@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.

```
\ifthenelse{\boolean{display}}{%
  \AtBeginDocument{\@ifpackageloaded{hyperref}{%
  \let\hyper@anchor@TP=\hyper@anchor
  \long\def\hyper@@anchor#1#2{%
    %%% \ifthenelse{\boolean{instepwise@TP}}%
    \ifthenelse{\boolean{instepwise@TP} \AND \NOT \boolean{oldfiltering@TP}}%
      {\ifthenelse{\NOT \boolean{instep@TP}}%
       {\ifthenelse{\value{step}=\value{firststep}}{\@hyper@@anchor#1\relax#2\relax}{}}%
       {\ifthenelse{\boolean{active} \AND \boolean{firstactivation}}%
            {\@hyper@@anchor#1\relax#2\relax}{}}% End \ifthenelse{\boolean{active} ...
      }% End \ifthenelse{\NOT \boolean{instep@TP}}
      {\@hyper@@anchor#1\relax#2\relax}% End \ifthenelse{\boolean{instepwise@TP}}
  }
  \let\hyper@anchorstart@TP=\hyper@anchorstart
  \def\hyper@anchorstart#1{%
    \ifthenelse{\boolean{instepwise@TP} \AND \NOT \boolean{oldfiltering@TP}}%
      {\ifthenelse{\NOT \boolean{instep@TP}}%
       {\ifthenelse{\value{step}=\value{firststep}}{\hyper@anchorstart@TP{#1}}{}}%
       {\ifthenelse{\boolean{active} \AND \boolean{firstactivation}}%
            {\hyper@anchorstart@TP{#1}}{}}% End \ifthenelse{\boolean{active} ...
      }% End \ifthenelse{\NOT \boolean{instep@TP}}
      {\hyper@anchorstart@TP{#1}}% End \ifthenelse{\boolean{instepwise@TP}}
  }
  }{}}% End \AtBeginDocument{\@ifpackageloaded{hyperref}{
  % Changing \protected@write to avoid duplicates in aux-file
  \let\protected@write@TP=\protected@write
  \long\def\protected@write#1#2#3{%
    \ifthenelse{\boolean{instepwise@TP} \AND \NOT \boolean{oldfiltering@TP}}%
      {\ifthenelse{\NOT \boolean{instep@TP}}%
       {\ifthenelse{\value{step}=\value{firststep}}{\protected@write@TP{#1}{#2}{#3}}{}}%
       {\ifthenelse{\boolean{active} \AND \boolean{firstactivation}}%
            {\protected@write@TP{#1}{#2}{#3}}{}}% End \ifthenelse{\boolean{active} ...
      }% End \ifthenelse{\NOT \boolean{instep@TP}}
      {\protected@write@TP{#1}{#2}{#3}}% End \ifthenelse{\boolean{instepwise@TP}}
  }%
}{}% 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.

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

```
  \global\setbox\memfootins@TP=\copy\footins%          Save footnotes.
  \global\skip\memfootins@TP=\skip\footins
  \global\count\memfootins@TP=\count\footins
  \global\dimen\memfootins@TP=\dimen\footins
  \output=\memoutput@TP%                               Restore output routine.
  }
```

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.

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

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

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

Insert saved page contents for the first time.

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

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

```
\newcommand{\do@colorcorrections@TP}
{%
  {%
    \colorcorrections@TP%               Execute color correction stack.
    \gdef\colorcorrections@TP{\relax}%  Re-initialize for next round.
    }%
  }
```

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

```
\newcommand{\insertsecondduplicate@TP}
{%
  \do@colorcorrections@TP%                          Execute color correction stack.
  \global\setbox\footins=\copy\memfootins@TP%        Restore footnotes.
  \global\skip\footins=\skip\memfootins@TP
  \global\count\footins=\count\memfootins@TP
  \global\dimen\footins=\dimen\memfootins@TP
  \ifthenelse{\boolean{oldfiltering@TP}}%
    {\unvcopy\mempageconts@TP}%
    {\setbox\tempbox@TP=\vbox{\unvcopy\mempageconts@TP}%
     \insertfilterwhatsits@TP
     \copy\tempbox@TP%
    }
```

```
}
```

Ship out a duplicated page.

```
\newcommand{\outputduplicate@TP}
{%
  \shippingduplicatetrue%                          This switch can be evaluated in panels or heade
  \global\let\filterpage@TP\filterwhatsits@TP%      Filter out file whatsits when shipping out.
  \global\let\o@hyper@@anchor@TP\hyper@@anchor%     Save definition of hyperref command for hyper a
  \global\let\hyper@@anchor\@gobble%               Disable hyper anchors on duplicate pages to avo
  \TPeject%                                        Shipout page.
  \global\let\hyper@@anchor\o@hyper@@anchor@TP%     Restore definition of hyperref command for hype
  \global\let\filterpage@TP\relax%                 Disable whatsit filtering.
  \shippingduplicatefalse%                         Unset switch.
  }%
```

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.

```
\providecommand\pause% If pause.sty is loaded, the existing definition of \pause is not overwri
{%
  \save@TP%                                        Save contents of the page...
  \insertfirstduplicate@TP%                        ... and insert again.
  \ifthenelse{\boolean{display}}%                  Are dynamic features enabled?
  {% Yes.
    \outputduplicate@TP%                           Output page.
    \restorepausecounters@TP%                      Restore counters (page number).
    \insertsecondduplicate@TP%                     Reinsert saved contents.
    }
  {}%
  }%
```

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

```
\newcommand{\@onlyinstepwise@TP}[1]
{%
  \providecommand#1%
  {%
    \PackageError{texpower}%
    {Command \string#1 can be used only inside \string\stepwise}
    {%
      Commands like \string\step, \string\switch,
      \string\multistep\space or \string\overlays\MessageBreak
      can be used only inside the argument of a \string\stepwise\space variant.
    }%
  }%
}
```

3.9.9 Registers

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

```
\newcounter{totalsteps}
```

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

```
\newcounter{firststep}
```

The number of the step currently being performed.

```
\newcounter{step}
```

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

```
\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).

```
\newcounter{stepsperformed}
```

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

```
\newboolean{firstactivation}
```

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

```
\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.

```
\newcommand{\displayboxed}
{%
  \ifmmode                                              % We need to distinguish between math and
    \expandafter\mathpalette\expandafter\math@db@TP  % In math mode, the style has to be respec
  \else
    \expandafter\text@db@TP
  \fi
}
\newcommand{\text@db@TP}[1]{\mbox{#1}}            % In text mode, we just use an \mbox.
\newcommand{\math@db@TP}[2]{\mbox{$\m@th#1{#2}$}} % In math mode, the style (#1) is inserted by
```

Display <stepcontents> 'as is'.

```
\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.

```
\newcommand{\hidephantom}[1]
{%
  {% a group makes redefinitions local
    \leavevmode\phantom{#1}%
```

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

```
    }%
  }
```

Ignore <stepcontents> completely.

```
\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.

```
\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.

```
\AtBeginDocument%
{%
  \@ifpackageloaded{soul}{%
    \@ifpackagelater{soul}{2002/05/28}{
      \DeclareRobustCommand*\hidetext{%
```

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

```
        \leavevmode
        \SOUL@setup
```

Make current token 'invisible'.

```
        \def\SOUL@everytoken{\phantom{\the\SOUL@token\SOUL@setkern\SOUL@charkern}}
        \def\SOUL@everyhyphen
        {%
          \discretionary
          {%
            \unkern
```

Output an 'invisible' hyphen if needed.

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

Too old soul package - encourage people to update.

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

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

```
      \let\hidetext=\hidephantom
    }%
```

```
  }{
    \PackageInfo{texpower}
    {Package soul not loaded.\MessageBreak Command \string\hidetext\space disabled}%
    \let\hidetext=\hidephantom
  }%
}
```

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

```
\ifthenelse{\boolean{colormath@TP}}{%
  \newcommand{\commitcolor@TP}
```

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

```
  {\textcolor{\ifmmode mathcolor\else textcolor\fi}}
}{
  \newcommand{\commitcolor@TP}{\textcolor{textcolor}}
}
```

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

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

Adding a group to make the color changes local.

```
    {
```

Just in case we currently are in dimmed mode.

```
      \ifthenelse{\boolean{instepwise@TP}}{\usecolorset{stwcolors}}{}%
      \dimcolors
      \commitcolor@TP{#1}
    }
  }
}{
  \let\hidedimmed=\displayidentical
}
```

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`.

```
\ifthenelse{\boolean{TPcolor}}{
  \newcommand{\hidevanish}[1]{\textcolor{\vanishcolor}{\vanishcolors#1}}
}{
  \let\hidevanish=\hidephantom
}
```

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.

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

```
{\let\displaystepcontents=\displayboxed\let\hidestepcontents=\hidephantom}

\newcommand{\nonboxedsteps}       % Use nonboxed versions.
{\let\displaystepcontents=\displayidentical\let\hidestepcontents=\hideignore}
```

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

```
\nonboxedsteps
```

There is another command named `\activatestep` which controls what happens when a `\step` command is activated for the first time. This is defined to do nothing by default.

```
\let\activatestep=\displayidentical
```

The length `\highlightboxsep` gives the width of the frame around the box created by `\highlightboxed`.

```
\newlength{\highlightboxsep}
\setlength{\highlightboxsep}{.5\fboxsep}
```

`\highlightboxed{<text>}` puts `<text>` into an `\mbox` with coloured background 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 resulting box has the same dimensions as the argument (the outer frame may overlap surrounding text).

```
\DeclareRobustCommand{\highlightboxed}
{%
  \ifmmode%                                          Check for math mode.
    \expandafter\mathpalette\expandafter\math@hb@TP%  Math mode version needs to respect curren
  \else
    \expandafter\text@hb@TP%                         Text mode version.
  \fi
}
```

Math mode version of `\highlightboxed`.

```
\newcommand{\math@hb@TP}[2]{\text@hb@TP{$\m@th#1{#2}$}}
```

The text mode version of `\highlightboxed` does the 'real' work.

```
\ifthenelse{\boolean{colorhighlight@TP}}%                Color highlighting enabled?
{%  Yes; use a box with colored background.
  \newcommand{\text@hb@TP}[1]
  {%
    \makebox[\width-2\highlightboxsep]%                  Make the frame stick out at the side
    {%
      \setlength{\fboxsep}{\highlightboxsep}%            Set frame size.
      \raisebox{0pt}[\height-\fboxsep][\depth-\fboxsep]% Make the frame stick out above and b
      {\colorbox{highlightcolor}{#1}}%                   Make colored box containing <text>.
      }%
    }%
  }% matches \ifthenelse{\boolean{colorhighlight@TP}}
{%  No; use an \fbox.
  \newcommand{\text@hb@TP}[1]
```

```
{%
  \makebox[\width-2\highlightboxsep-2\fboxrule]%         Make the frame stick out at the side
  {%
    \setlength{\fboxsep}{\highlightboxsep}%              Set frame size.
    \raisebox{0pt}[\height-\fboxsep-\fboxrule][\depth-\fboxsep-\fboxrule]% Make the frame sti
    {\fbox{#1}}%%                                        Make an fbox containing <text>.
    }%
  }%
}% 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.

```
\AtBeginDocument%
{%
  \@ifpackageloaded{soul}%                      Can we use the soul package?
  {%
    \@ifpackagelater{soul}{2002/05/28}%         Correct version?
    {% Yes. Let's define the necessary macros.
      \ifthenelse{\boolean{colorhighlight@TP}}% Color highlighting enabled?
      {%  Yes; use a colored background.
       %  This is implemented as an application of soul (modifying the code for underline). Se
       %  soul package for details on soul.
        \newlength{\SOUL@boxheight@TP}%         Height of colored patch.
        \newlength{\SOUL@boxtotalheight@TP}%    Total height of colored patch.
        \newlength{\SOUL@boxdepth@TP}%          Depth of colored patch.
        \DeclareRobustCommand*\highlighttext
        {%
          \leavevmode%                          To prevent \smash from inserting an hb
          \SOUL@ulsetup%                        Underline initialization.
          \def\SOUL@preamble
          {%
            \setlength{\SOUL@boxdepth@TP}%          \SOUL@uldepth is below the depth o
            {\SOUL@uldepth+\highlightboxsep}%
            \def\SOUL@uldepth{-\SOUL@boxheight@TP}% For correctly positioning the rule
            \setlength{\SOUL@boxheight@TP}{\heightof{/}+\highlightboxsep}%          Calcula
            \setlength{\SOUL@boxtotalheight@TP}{\SOUL@boxdepth@TP+\SOUL@boxheight@TP}% Calcula
            \def\SOUL@ulthickness{\SOUL@boxtotalheight@TP}% The thickness of the rule is the t
            \smash%                                 Make the left border of the colore
            {%
```

```
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}{\SOU
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
```

```
        \def\SOUL@everyhyphen{\discretionary
          {\unkern
            \setbox4\hbox{\SOUL@setkern\SOUL@hyphkern\char\hyphenchar\font}%
            \smash{\rlap{\SOUL@ulleaders\hskip\wd4}\box4}}{}%
          {}}%
        \SOUL@%
        }%
      }% matches second argument of \ifthenelse{\boolean{colorhighlight@TP}}%
    }%
    {% No. Encourage people to update.
      \PackageInfo{texpower}
      {Package soul too old.\MessageBreak Command \string\highlighttext\space disabled}%
      \let\highlighttext=\@iden%
    }%
  }% matches \@ifpackageloaded{soul}
  {% No. In this case, there is no useful definition for \highlighttext.
    \PackageInfo{texpower}
    {Package soul not loaded.\MessageBreak Command \string\highlighttext\space disabled}%
    \let\highlighttext=\@iden
    }% matches second argument of \@ifpackageloaded{soul}
  }% matches \AtBeginDocument%
```

The command `\highlightenhanced` enhances all colors so the argument 'stands out'.

```
\ifthenelse{\boolean{TPcolor}}% Can we use colors at all?
{% Yes.
  \newcommand{\highlightenhanced}[1]%             Make argument appear in 'enhanced' colors.
  {%
    {% A group makes the color changes local.
      \ifthenelse{\boolean{instepwise@TP}}{\usecolorset{stwcolors}}{}% Just in case we currentl
      \enhancecolors%          Enhance colors.
      \commitcolor@TP{#1}%     Switch on enhanced color.
      }%
    }%
  }
{\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'th 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 syntctical 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 LaTeX. That means, `<activatefirst>` (when enclosed in braces) can contain arbitrary TeXcode 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`.

```
\newcommand{\pickup@s@optargs@TP}
{%
  \@ifnextchar[%]           Check for first optional argument in [...] syntax.
  {\f@brackstep@TP}%
  {%
    \@ifnextchar(%)         Check for first optional argument in (...) syntax.
    {\f@parenstep@TP}
```

```
    {\f@brackstep@TP[]}%     No optional argument given; call with empty argument in [...] synt
    }%
}

\def\f@brackstep@TP[#1]%     First optional argument was given in [...] syntax.
{%
  \def\tmp@TP{#1}%
  \ifx\tmp@TP\empty%         Optional argument empty?
    \def\f@step@TPcheck{\ifthenelse{\value{step}=\value{stepcommand}}}% Yes; use default.
   \else
    \def\f@step@TPcheck{\ifthenelse{#1}}% No; condition is defined via \ifthenelse.
  \fi
  \f@step@TP%                Go on and check for second optional argument.
}

\def\f@parenstep@TP(#1)%     First optional argument was given in (...) syntax.
{%
  \def\f@step@TPcheck{#1}%   Save condition (given as argument).
  \f@step@TP%                Go on and check for second optional argument.
  }

\newcommand{\f@step@TP}%     Pick up the second optional argument.
{%
  \@ifnextchar[%]            Check for second optional argument in [...] syntax.
  {\s@brackstep@TP}
  {%
    \@ifnextchar(%)          Check for second optional argument in (...) syntax.
    {\s@parenstep@TP}
    {\s@parenstep@TP(\if@first@TP@true)}% No second optional argument given;
    }%                                   call with \if@first@TP@true in (...) syntax (default)
}

\def\s@brackstep@TP[#1]%     Second optional argument was given in [...] syntax.
{%
  \def\s@step@TPcheck{\ifthenelse{#1}}% Condition is defined via \ifthenelse.
  \@do@s@TP%                Go on.
  }

\def\s@parenstep@TP(#1)%     Second optional argument was given in (...) syntax.
{%
  \def\s@step@TPcheck{#1}%   Save condition (given as argument).
  \@do@s@TP%                Go on.
  }
```

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`.

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

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

```
  \let\inner@display@TP=\displayidentical%
  \let\inner@hide@TP=\displayidentical%
  }

\newcommand{\activate@inner@TP}% \step behaviour.
{%
```

Use the user interface macros \displaystepcontents and \hidestepcontents.

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

Implementation of \step.

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

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

```
\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.

```
\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.

```
\@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.

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

\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.

```
\newcommand{\if@first@TP@true}[2]%
{%
  \expandafter                    % \first@TP@<n> is set to \@undefined if \switch command numbe
  \ifx\csname first@TP@\the\c@stepcommand\endcsname\empty%                              activated
    #1%
   \else
    #2%
  \fi
  }
```

\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.

```
\@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.

```
\newcommand{\@switch@TP}[2]
{%
  \global\advance\c@stepcommand by 1\relax% This execution of \switch is counted.
  \setboolean{instep@TP}{true}%             Set indicator.
  %
  % If the verbose option is set, type out some accounting information which can be used for de
  \ifthenelse{\boolean{verbose@TP}}
  {%
    \PackageInfo{texpower}
    {Step: \the\c@step, Stepcommand: \the\c@stepcommand,\MessageBreak Stepsperformed: \the\c@st
    }
  {}%
  %
  \f@step@TPcheck% Is this step to be activated? \f@step@TPcheck is defined by the first option
  {% Yes.
    \if@first@TP@true{}% For non-unique conditions given as optional argument or when \reswitch
    {%       that \first@TP@\value{stepcommand} is already set. In this case, don't advance the
      \ifmeasuring@\else% Do nothing in AMSLaTeX's measuring pass for aligned equations.
        \global\expandafter\let\csname first@TP@\the\c@stepcommand\endcsname=\empty% Set \first
      \fi
      \global\advance\c@stepsperformed by 1\relax% Advance the counter for 'real' first activat
      \gdef\current@step@TP{#2}%
      }%
    \setboolean{firstactivation}{true}% This switch can be tested in <stepcontents>, but also i
    }
  {% No.
    \setboolean{firstactivation}{false}% This switch can be tested in <stepcontents>, but also
    }% End of \f@step@TPcheck
  %
  \let\o@afterstep@TP=\afterstep% We need to save the current definition of \afterstep.
  %
  \s@step@TPcheck% Is this step active? \s@step@TPcheck is defined by the second optional argum
  {% Yes.
    \setboolean{active}{true}%              Make this fact known to the user.
    \ifthenelse{\boolean{firstactivation}}
    {\inner@display@TP{\activatestep{#2}}}% 'First' display of <stepcontents>.
    {\inner@display@TP{#2}}%                Display <stepcontents>.
    }
  {% No.
    \setboolean{active}{false}%             Make this fact known to the user.
    \let\afterstep=\@gobble%                Don't execute \afterstep here.
    \ifthenelse{\boolean{firstactivation}}
    {\inner@hide@TP{\activatestep{#1}}}%    Hide <stepcontents>, but with 'first activation'.
```

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

`\restep` is identical with `\step`, but is displayed at the same time as the previous `\step`.

```
\@onlyinstepwise@TP\restep

\newcommand{\proper@restep@TP}
{%
  \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
  \proper@step@TP%                      Go on with \step.
  }
```

`\reswitch` is identical with `\switch`, but is displayed at the same time as the previous `\switch`.

```
\@onlyinstepwise@TP\reswitch

\newcommand{\proper@reswitch@TP}
{%
  \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
  \proper@switch@TP%                      Go on with \switch.
  }
```

`\bstep` is a variant of `\step` which is always boxed.

```
\@onlyinstepwise@TP\bstep

\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`.

```
\newcommand{\@bstep@TP}[1]{{\boxedsteps\@step@TP{#1}}}
```

`\rebstep` is identical with `\bstep`, but is displayed at the same time as the previous `\bstep`.

```
\@onlyinstepwise@TP\rebstep

\newcommand{\proper@rebstep@TP}
{%
  \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
  \proper@bstep@TP%                      Go on with \bstep.
  }
```

`\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).

```
\ifthenelse{\boolean{colormath@TP}}% Should we color math?
```

```
{% Yes.
  \newcommand{\commitcolors@TP}
  {\color{\ifmmode mathcolor\else textcolor\fi}}% Switch to mathcolor if in math mode, to textc
  }
{% No.
  \newcommand{\commitcolors@TP}{\color{textcolor}}% Switch to textcolor.
}

\@onlyinstepwise@TP\dstep

\newcommand{\proper@dstep@TP}{\deactivate@inner@TP\let\@do@s@TP=\@dstep@TP\pickup@s@optargs@TP}

\ifthenelse{\boolean{TPcolor}}% Can we use colors at all?
{% Yes.
  \newcommand{\@dstep@TP}{\@@switch@TP{\dimcolors\commitcolors@TP}{\set@color}}
  }
{\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.

```
\@onlyinstepwise@TP\redstep

\newcommand{\proper@redstep@TP}
{%
  \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
  \proper@dstep@TP%                    Go on with \dstep.
  }
```

\vstep is a variant of \step which takes **no** argument, but switches all colors to \vanishcolor.

```
\@onlyinstepwise@TP\vstep

\newcommand{\proper@vstep@TP}{\deactivate@inner@TP\let\@do@s@TP=\@vstep@TP\pickup@s@optargs@TP}

\ifthenelse{\boolean{TPcolor}}% Can we use colors at all?
{% Yes.
  \newcommand{\@vstep@TP}{\@@switch@TP{\vanishcolors\color{textcolor}}{\set@color}}
  }
{\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.

```
\@onlyinstepwise@TP\revstep

\newcommand{\proper@revstep@TP}
{%
  \global\advance\c@stepcommand by -1% This is done by simply counting \value{stepcommand} back
  \proper@vstep@TP%                    Go on with \vstep.
  }
```

\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.

```
\@onlyinstepwise@TP\steponce

\newcommand{\proper@steponce@TP}
{\@ifnextchar[{\brack@steponce@TP}{\@steponce@TP}}% Optional argument in square brackets?

\newcommand{\@steponce@TP}
{%
  \@ifnextchar(%%          Optional argument in parentheses?
  {\paren@steponce@TP}%
  {\brack@steponce@TP[]}%  [] is the default if no optional argument is given.
  }

\def\brack@steponce@TP[#1]%
{%
  \def\optarg@so@TP{[#1]}% Store optional argument for later re-insertion.
  \@@steponce@TP%          Proceed.
  }%

\def\paren@steponce@TP(#1){\def\optarg@so@TP{(#1)}\@@steponce@TP}
```

Main body of `\steponce`.

```
\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>}{<ste` expands to a sequence of `<n>` commands of the form `\switch[<activatefirst>][\boolean{firstactivat` 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.

```
\newcounter{substep}
```

User interface for `\multistep`.

```
\@onlyinstepwise@TP\multistep

\newcommand{\proper@multistep@TP}
{%
  \let\ns@ms@TP=\normalstep@ms@TP% Placeholder for 'every step but the last one'.
  \let\nshook@ms@TP=\relax%       These hooks are used by \movie.
  \let\lshook@ms@TP=\relax
```

```
  \@ifstar%                         Starred version?
  {%
    \let\ls@ms@TP=\laststep@ms@TP% Last step acts differently.
    \multistep@TP%                Collect optional argument and proceed.
    }
  {%
    \let\ls@ms@TP=\normalstep@ms@TP% Last step acts like all other steps.
    \multistep@TP%                  Collect optional argument and proceed.
    }%
  }

\newcommand{\multistep@TP}% Collect optional argument.
{\@ifnextchar[{\brack@multistep@TP}{\@multistep@TP}}% Optional argument in square brackets?

\newcommand{\@multistep@TP}
{%
  \@ifnextchar(%%          Optional argument in parentheses?
  {\paren@multistep@TP}%
  {\brack@multistep@TP[]}% [] is the default if no optional argument is given.
  }

\def\brack@multistep@TP[#1]%
{%
  \def\optarg@ms@TP{[#1]}% Store optional argument for later re-insertion.
  \@@multistep@TP%         Proceed.
  }%

\def\paren@multistep@TP(#1){\def\optarg@ms@TP{(#1)}\@@multistep@TP}

```

Execute one step.

```
\newcommand{\normalstep@ms@TP}[1]
{%
  \expandafter\switch\optarg@ms@TP%  Re-insert optional argument.
  [\boolean{firstactivation}]{}{#1}% 'normal' steps appear only once.
  }

\newcommand{\laststep@ms@TP}[1]
{\expandafter\switch\optarg@ms@TP{}{#1}}% In the starred version, the last step doesn't disappe
```

Main body of `\multistep`.

```
\newcommand{\@@multistep@TP}[2]
{%
  \setcounter{substep}{0}%       Initialize substep counter.
  \whiledo{\value{substep}<#1}%  Iterate <n> times.
  {%
    \stepcounter{substep}%
    \ifthenelse{\value{substep}=#1}% Last step?
    {\ls@ms@TP{\lshook@ms@TP#2}}%    Execute single step (together with \movie hooks).
    {\ns@ms@TP{\nshook@ms@TP#2}}%
    }%
```

```
}
```

`\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`.

```
\@onlyinstepwise@TP\movie

\newcommand{\proper@movie@TP}
{%
  \let\ns@ms@TP=\normalstep@ms@TP% Placeholder for 'every step but the last one'.
  \def\nshook@ms@TP{\afterstep{\pageDuration{\dur@ms@TP}}}% Page duration to be used between st
  \def\lshook@ms@TP{\afterstep{\end@ms@TP}}%               Page duration setting after last st
  \@ifstar%                         Starred version?
  {%
    \let\ls@ms@TP=\laststep@ms@TP% Last step acts differently.
    \movie@TP%                     Collect optional argument and proceed.
    }
  {%
    \let\ls@ms@TP=\normalstep@ms@TP% Last step acts like all other steps.
    \movie@TP%                       Collect optional argument and proceed.
    }%
  }

\newcommand{\movie@TP}% Collect optional argument.
{\@ifnextchar[{\brack@movie@TP}{\@movie@TP}}%       Optional argument in square brackets?

\newcommand{\@movie@TP}
{\@ifnextchar({\paren@movie@TP}{\brack@movie@TP[]}}% Optional argument in parentheses?

\def\brack@movie@TP[#1]{\def\optarg@ms@TP{[#1]}\@@movie@TP}% Store optional argument and procee
\def\paren@movie@TP(#1){\def\optarg@ms@TP{(#1)}\@@movie@TP}

\newcommand{\@@movie@TP}[2]% Collect <n> and <dur> arguments.
{\gdef\dur@ms@TP{#2}\gdef\nosteps@ms@TP{#1}\@@@movie@TP}

\newcommand{\@@@movie@TP}[1][\stopAdvancing]% Collect second optional argument and call body of
{\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`.

```
\@onlyinstepwise@TP\overlays

\providecommand{\proper@overlays@TP}
{\@ifnextchar[{\brack@overlays@TP}{\@overlays@TP}}% Optional argument in square brackets?

\newcommand{\@overlays@TP}
{%
  \@ifnextchar(%%          Optional argument in parentheses?
  {\paren@overlays@TP}%
  {\brack@overlays@TP[]}%  [] is the default if no optional argument is given.
  }

\def\brack@overlays@TP[#1]%
{%
  \def\optarg@ov@TP{[#1]}% Store optional argument for later re-insertion.
  \@@overlays@TP%          Proceed.
  }%

\def\paren@overlays@TP(#1){\def\optarg@ov@TP{(#1)}\@@overlays@TP}
```

Main body of `\overlays`.

```
\newcommand{\@@overlays@TP}[2]
{%
  \setcounter{substep}{1}%       Initialize substep counter.
  \leavevmode%                   Make sure that \rlap doesn't insert anything in the vertical l
  \whiledo{\value{substep}<#1}%  Iterate <n-1> times (since we start at 1).
  {%
    \expandafter\switch\optarg@ov@TP{}{\ifthenelse{\boolean{firstactivation}}{\mbox{#2}}{\rlap{
    \stepcounter{substep}%
  }%
  \expandafter\switch\optarg@ov@TP{}{\mbox{#2}}% Always using \mbox for last overlay.
}
```

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).

```
\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.

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

One new counter for saving the value of firststep.

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

\stepwise user interface.

```
\newcommand{\stepwise}
{%
  \global\c@o@fs@TP=\c@firststep\relax%     Save the default value of counter firststep.
  \@ifstar%                                 Using the starred version?
  {% Yes.
    \c@firststep=1\relax%                   Start with counter step at number 1.
    \@stepwise@TP%                          Collect optional argument and proceed.
    }
  {% No.
    \@stepwise@TP%                          Use the default.
    }% End of \@ifstar.
  }%
```

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.

```
\newcommand{\list@stepcapsule}
{%
```

```
  \renewenvironment{stepcapsule}%                   stepcapsule is to put an invisible rule on
  {\vspace*{\parskip}\hrule \@height\z@\relax}{}
  }%

\let\par@stepcapsule=\list@stepcapsule%             Identical for \parstepwise.
```

User interface for `\liststepwise`.

```
\newcommand{\liststepwise}
{%
  \@ifstar%                             Starred version?
  {\def\star@TP{*}\@liststepwise@TP}% Save star in \star@TP, collect optional argument and proc
  {\def\star@TP{}\@liststepwise@TP}%  Collect optional argument and proceed (non-starred versio
  }

\newcommand{\@liststepwise@TP}[1][]%  Collect optional argument.
{%
  \expandafter\stepwise\star@TP%                            Re-insert the star (if given).
  [%
    \list@stepcapsule%                                      Activate special stepcapsule.
    #1%                                                     Insert optional argument of \lists
    ]%
  }
```

User interface for `\parstepwise`.

```
\newcommand{\parstepwise}
{%
  \@ifstar%                             Starred version?
  {\def\star@TP{*}\@parstepwise@TP}%  Save star in \star@TP, collect optional argument and proc
  {\def\star@TP{}\@parstepwise@TP}%   Collect optional argument and proceed (non-starred versio
  }

\newcommand{\@parstepwise@TP}[1][]%  Collect optional argument.
{%
  \expandafter\stepwise\star@TP%     Re-insert the star (if given).
  [%
    \boxedsteps%                     Activate \boxedsteps.
    \par@stepcapsule%                Activate special stepcapsule.
    #1%                              Insert optional argument of \parstepwise.
    ]%
  }
```

`\count@em@TP` is used by `\stepwise` as a redefinition of `\@@switch@TP` for counting `\step` commands.

```
\newcommand{\count@em@TP}[2]
{%
  \global\advance\c@stepcommand by 1\relax% We simply advance the number of \step commands...
  #2%                                      ... and execute the second argument (to find nested u
}
```

`\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'.

```
\def\@nb@TPfalse{\global\let\if@nb@TP\iffalse}
\def\@nb@TPtrue{\global\let\if@nb@TP\iftrue}
\newtoks\ep@TP
\def\savecounters@TP{%
  \begingroup
    \def\@elt##1{\global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
    \xdef\restorecounters@TP{\cl@@ckpt}%
  \endgroup
  \if@nobreak\@nb@TPtrue\else\@nb@TPfalse\fi
  \global\ep@TP\everypar
  }
```

`\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.

```
\def\saveTPcounters@TP{%
  \begingroup
    \def\@elt##1{\global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
    \xdef\restoreTPcounters@TP{\cl@@ckpt@TP}%
  \endgroup
  }
```

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

```
\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.

```
\newcommand{\releasecounter}[1]%
{\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.

```
\newcommand{\disable@counting@TP}
{%
  \let\afterstep=\@gobble
  \renewcommand{\backgroundstyle}[2][]{}%
  \renewcommand{\@@vgradrule@TP}[3][0pt]{}%
  \renewcommand{\@@@@dblvgradrule@TP}[3][0pt]{}%
  \renewcommand{\@@hgradrule@TP}[3][0pt]{}%
  \renewcommand{\@@@@dblhgradrule@TP}[3][0pt]{}%
}
```

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

```
\newboolean{instepwise@TP}
```

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

```
\newboolean{instep@TP}
```

`\@stepwise@TP` implements the functionality of `\stepwise`. It is called by `\stepwise` after checking for the star.

```
\newcommand{\@stepwise@TP}[2][]
{%
  \save@TP%                              Save the current contents of the page.
  \savecounters@TP%                      Save the values of all counters.
  \dumpcolorset{stwcolors}%              Make a copy of all color definitions.
  \begingroup%                           A group makes redefinitions local.
    \setboolean{instepwise@TP}{true}%    Set indicatior.
    \let\step\proper@step@TP%            By default, \step executes \proper@step@TP.
    \let\restep\proper@restep@TP%
    \let\bstep\proper@bstep@TP%
    \let\rebstep\proper@rebstep@TP%
    \let\dstep\proper@dstep@TP%
    \let\redstep\proper@redstep@TP%
    \let\vstep\proper@vstep@TP%
    \let\revstep\proper@revstep@TP%
    \let\steponce\proper@steponce@TP%
    \let\multistep\proper@multistep@TP%
    \let\movie\proper@movie@TP%
    \let\overlays\proper@overlays@TP%
    \let\switch\proper@switch@TP%        By default, \switch executes \proper@switch@TP.
    \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.

```
    \let\orig@mathchoice@TP=\mathchoice% Save the current definition of \mathchoice...
    \def\mathchoice##1##2##3##4%         ... and redefine.
    {%
      \orig@mathchoice@TP%               The original definition of \mathchoice is called...
      {##1}%                             ... with the first argument untouched and in all othe
      {\let\step\restep\let\bstep\rebstep\let\dstep\redstep\let\vstep\revstep\let\switch\reswit
      {\let\step\restep\let\bstep\rebstep\let\dstep\redstep\let\vstep\revstep\let\switch\reswit
      {\let\step\restep\let\bstep\rebstep\let\dstep\redstep\let\vstep\revstep\let\switch\reswit
      }%
    %
    \c@stepcommand=0\relax%              Initialize the counter for \step commands.
    \let\@@switch@TP=\count@em@TP%        Next, we count the \step commands in <contents>...
    \setbox\tempbox@TP%                   ... by putting <contents> into a \vbox (which is the
    =\vbox
    {%
      \hfuzz\maxdimen\hbadness\@M\relax% No bogus 'overfull \hbox' warnings.
      \disable@counting@TP#2%            Inside the \vbox, some commands are redefined for saf
```

```
}%
\c@totalsteps=\c@stepcommand%         Now, we know the total number of \step commands.
%
% Next, we have to reset \first@TP@<n> for <n>=0...\value{totalsteps}.
\c@step=0\relax
\loop
 \ifnum\c@step<\c@totalsteps
  \advance\c@step by 1\relax
  \expandafter\let\csname first@TP@\the\c@step\endcsname=\@undefined% \first@TP@<n>=\@undef
                                                                    % hasn't yet been activ
\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>`.

```
#1%
%
\c@step=\c@firststep%                 Set the counter for the current step to its startin
\c@stepsperformed=0\relax%            ... and also the counter for the \step commands whi
\let\@@switch@TP=\@switch@TP%          \step will now act normally.
\ifthenelse{\boolean{verbose@TP}}%    Some accounting info (if verbose option is set).
{\PackageInfo{texpower}{Total number of step commands: \the\c@totalsteps}}
{}%
%
\ifthenelse{\boolean{display}}%       Are dynamic features enabled?
{}% Yes.
{%  No. Do only one loop.
  \c@stepsperformed=\c@totalsteps%    Set everything up for the last loop.
  \c@step=0\relax%                    Set \first@TP@<n> for <n>=0...\value{totalsteps}.
  \loop
   \ifnum\c@step<\c@totalsteps
    \advance\c@step by 1\relax
    \expandafter\let\csname first@TP@\the\c@step\endcsname=\@empty% \first@TP@<n>=\@empty m
                                                                  % has already been activa
  \repeat
  \advance\c@step by 1\relax%         This way, the last step won't think it's 'first act
  }%
\let\insertdup@TP=\insertfirstduplicate@TP% Setup command to restore page contents for the
\loop%                                This is the central loop.
  \c@stepcommand=0\relax%             Initialize the counter for the current \step comman
  \saveTPcounters@TP%                 Save the 'stepwise-specific' counters.
  \restorecounters@TP%                Restore the 'original' values of all counters...
  \restoreTPcounters@TP%              ... and the current values of the 'stepwise-specifi
  \let\@afterstep@TP=\relax%          Reset the container for \afterstep.
  %
  \insertdup@TP%                      Insert saved page contents.
  \begin{stepcapsule}%                This is usually empty, but may start a minipage (or
    \usecolorset{stwcolors}%          Restore colors to state at the beginning of \stepwi
    \if@nb@TP\@nobreaktrue\else\@nobreakfalse\fi
    \global\everypar\ep@TP
```

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

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.

```
\newenvironment{fragilesteps}{%
  \def\texpower@verbatimfilename{\jobname-texpower.vrb}%
  \immediate\openout\texpower@verbatimfileout=\texpower@verbatimfilename%
  \texpower@verbatimreadframe%
}
{%
  \immediate\closeout\texpower@verbatimfileout%
  \stepwise{\input{\texpower@verbatimfilename}}%
}
```

Internals used in the fragilesteps environment.

```
\newwrite\texpower@verbatimfileout

\def\texpower@verbatimreadframe{%
  \begingroup%
  \let\do\@makeother\dospecials%
  \count@=127%
  \@whilenum\count@<255 \do{%
    \advance\count@ by 1%
    \catcode\count@=11%
  }%
  \@makeother\^^L% and whatever other special cases
  \endlinechar`\^^M \catcode`\^^M=12 \texpower@processframefirstline}

{\catcode`\^^M=12\endlinechar=-1%
  \long\gdef\texpower@processframefirstline#1^^M{%
    \def\texpower@test{#1}%
    \ifx\texpower@test\texpower@stopframefirst%
      \let\next=\texpower@endfragilesteps%
    \else
      \ifx\texpower@test\@empty%
      \else%
```

```
        \@temptokena={#1}%
        \immediate\write\texpower@verbatimfileout{\the\@temptokena}%
      \fi%
      \let\next=\texpower@processframeline%
    \fi%
    \next%
  }
  \long\gdef\texpower@processframeline#1^^M{%
    \def\texpower@test{#1}%
    \ifx\texpower@test\texpower@stopframe%
      \let\next=\texpower@endfragilesteps%
    \else
      \immediate\write\texpower@verbatimfileout{#1}%
    \fi%
    \next%
  }
}

{
  \escapechar=-1\relax%
  \xdef\texpower@stopframe{\string\\end\string\{fragilesteps\string\}}
  \xdef\texpower@stopframefirst{\noexpand\end\string\{fragilesteps\string\}}
}

\def\texpower@endfragilesteps{\endgroup\end{fragilesteps}}
```

3.9.16 Input system-specific settings

If file exists.

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

Change History

added \bstep, \switch, \rebstep, \reswitch. 2

v0.0.7a

General: \pause and \stepwise now use \leaders for inserting duplicated parts of pages. This way, processing of whatsits is turned off in the duplicates so that table of contents entries are no longer duplicated when a section occurs on a page where \pause or \stepwise is used (spotted by heiner richter). Thanks to Heiko Oberdiek for his suggestion how \leaders can be successfully applied for this purpose. \stepwise now does the right thing if no \step command occurs in contents. There was a bug in \save@TP which would become apparent if \stepwise was the first thing on a page. Spotted and fixed by Ross Moore (thanks). 2

v0.0.7b

General: Changed hyperref version check from 2000/03/22 to 2000/03/23. Spotted by Ross Moore. 2

v0.0.7c

General: \eject changed to \newpage in \stepwise to cure some problems with the foils package. Spotted by Ross Moore. 2

v0.0.7d

General: \everydisplay finally removed from colormath option because it only causes trouble. Maybe I should look for a less fragile solution for the whole thing. Fixed a bug in texpower's definition of \set@page@color (used only if pdftex.def doesn't define it) which would cause a fatal error if two \pagecolor commands occur. Added command \replacecolor. 2

v0.0.8a

General: Fixed a bug in the code which disables \pageDuration if the pdfpageduration key doesn't exist (spotted by Friedrich Eisenbrand). 2

v0.0.8b

General: Added additional 'dimmed' and 'enhanced' color sets for all standard colors, with corresponding commands \dimcolors and \enhancecolors. Added a 'color stack correction' option fixcolorstack, which should avoid that the duplication of "color push" and "color pop" specials confuses the driver's color stack (spotted by Ross Moore). Added new display commands \hidedimmed, \hidevanish, \highlightenhanced. Added \step variants \dstep and \vstep. Added patches for \[, equation, eqnarray, and eqnarray* when the colormath option is used. Now also saving and restoring footnotes at \pause and \stepwise. Added a command \releasecounter to keep a counter from 'freezing' during the execution of \stepwise. . . 2

v0.0.8c

General: The default duplication method will now (only) attack \protected@write. There's a new option hackwrite which restores the former default method (of attacking \write). Corrected a bug newly introduced into \switch with version 0.0.8b. Corrected a bug in the color correction code (spotted by Ross Moore). 2

v0.0.8d

General: Corrected a minor quirk in \hidetext. Added a command \addTPcolor 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 robustduplicates and hackwrite are obsolete now. Thanks to Martin Schroeder for permission to use his everyshi code. Fixed a small quirk in `\dstep` and `\vstep`. `\darkbackground` and relatives now set both page and text color. 2

v0.0.8f

General: A small change in the definition of `\liststepwise` 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. `\dstep` and `\vstep` now understand the usual optional arguments. 2

v0.0.9

General: Added support for structured backgrounds (command `\backgroundstyle`). New commands for gradient rules and boxes. Added a hack to keep hyperref from producing duplicate page anchors (suggested by Thomas Emmel). Some slight changes in the mode of accounting in `\step` to hopefully give better results for 'complicated' orders of activating steps using `\step`'s optional arguments. Added (experimental) commands `\multistep` and `\movie` for aiding in (simple) animations. `\pagecolor` hack removed, as pdftex.def on CTAN now supports `\pagecolor`. `\set@color` hack for seminar removed (made unneccessary by enhancements to powersem). 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 texpower, texpower's color management is no longer activated automatically. Definitions of standard colors moved to file tpcolors.cfg. Option 'slifonts' is obsolete now. The code is now part of the much more sophisticated package "tpslifonts". Now the 'colormath' option cooperates with array.sty (and thus colortbl.sty). New option "nineminutes" to circumvent a strange behaviour of acrobat/acroread v4.05 and later wrt. page duration. Option 'fixcolorstack' now checks also for VTeX. `\hidedimmed`, `\highlightenhanced` and `\dstep` now check for math mode. Now using ifpdf package if available. 2

v0.0.9b

General: colormath adapted to different handling of 'equation' by amsmath 2.x. Adapted for new version of soul package. Added another patch to colormath for handling array package's "m" columns without color change. 'Turn on' seminar parameters in panel boxes even before `\begin{document}`. Added a command `\overlays`, sibling of `\multistep`, which prints all steps over each other. Added a command `\steponce`, sibling of `\step`, which is active only for one step. Removed a

Index

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

Symbols

A

B

C

E

F

G

H

I

J

K

L

N

O

P

R

S

T

U

V

W

X

Y

Z