The color package*

D. P. Carlisle

2026-05-17

This file is maintained by the LaTeX Project team. Bug reports can be opened (category graphics) at https://latex-project.org/bugs.html.

1 Introduction

This package implements LaTeX support for color, for those dvi drivers that can produce colored text.

The user level documentation of this package is contained in the document *Packages in the 'graphics' bundle* in the file grfguide.tex.

2 Options

1 ⟨*package⟩

First we save the catcodes of some characters, and set them to fixed values whilst this file is being read.

```
2 \edef\Gin@codes{%
3  \catcode`\noexpand\^^A\the\catcode`\^^A\relax
4  \catcode`\noexpand\"\the\catcode`\"\relax
5  \catcode`\noexpand\*\the\catcode`\*\relax
6  \catcode`\noexpand\!\the\catcode`\!\relax
7  \catcode`\noexpand\:\the\catcode`\:\relax}
8 \catcode`\^^A=\catcode`\%
9 \catcode`\"=12
10 \catcode`\*=11
11 \catcode`\!=12
12 \catcode`\:=12
```

\Gin@driver Initialize the macro to hold the driver file name.

```
13 \providecommand\Gin@driver{}
```

\c@lor@error Helper macro for error handling (redefined by the monochrome option to make errors warnings).

```
14 \def\c@lor@error#1{%
15   \@latex@error{Undefined color #1}\@ehd}
```

*This file has version number v1.3f, last revised 2026-05-17.

\ds@monochrome The `monochrome` option should be used *in addition* to one of the standard driver options.

```
\DeclareOption{monochrome}{%
  \def\c@lor@error#1{\PackageInfo{color}{Undefined color #1}}%
  \AtEndOfPackage{%
    \colors@false
    \def\set@color{%
      \c@lor@special\m@ne
        {color push \current@color}\aftergroup\reset@color}%
    \def\reset@color{\c@lor@special\m@ne{color pop}}%
    \def\set@page@color{%
      \c@lor@special\m@ne{background \current@color}}%
    \def\define@color#1#2{%
      \c@lor@special\m@ne{define #1 #2}}}}
```

\ds@debugshow The `debugshow` option turns on debugging info (perhaps).

```
\DeclareOption{debugshow}{\catcode`\^^A=9 \let\GDebug\typeout}
```

\ds@setpagesize \ds@nosetpagesize The `setpagesize` option requests that the driver option sets the page size. (Whichever option is used, the page size is not set by this package if `\mag` has been changed from its default value.)

```
\newif\ifGin@setpagesize\Gin@setpagesizetrue
\DeclareOption{setpagesize}{\Gin@setpagesizetrue}
\DeclareOption{nosetpagesize}{\Gin@setpagesizefalse}
```

Now the options for supported drivers.

\ds@dvips \ds@xdvi Tom Rokicki's dvips driver, and the X Windows previewer, xdvi, which uses (a subset of) the same `\specials`.

```
\DeclareOption{dvips}{\def\Gin@driver{dvips.def}%
                      \def\c@lor@namefile{dvipsnam.def}}
\DeclareOption{xdvi}{\ExecuteOptions{dvips,monochrome}}
```

\ds@dvipdf Sergey Lesenko's dvipdf driver.

```
\DeclareOption{dvipdf}{\def\Gin@driver{dvipdf.def}}
```

\ds@dvipdfm Mark Wick's dvipdfm driver (now merged with xdvipdfmx).

```
\DeclareOption{dvipdfm}{\def\Gin@driver{dvipdfmx.def}}
```

\ds@dvipdfmx The driver for the dvipdfmx project.

```
\DeclareOption{dvipdfmx}{\def\Gin@driver{dvipdfmx.def}}
```

\ds@pdftex Han The Thanh's TEX variant.

```
\DeclareOption{pdftex}{\def\Gin@driver{pdftex.def}}
```

\ds@luatex LuaTEX TEX variant.

```
\DeclareOption{luatex}{\def\Gin@driver{luatex.def}}
```

\ds@luatex dvisvgm driver.

```
\DeclareOption{dvisvgm}{\def\Gin@driver{dvisvgm.def}}
```

\ds@xetex Jonathan Kew's TEX variant.

```
\DeclareOption{xetex}{\def\Gin@driver{xetex.def}}
```

\ds@dvipsone \ds@dviwindo The drivers of the Y&Y TEX system. (Which use the same \specials.)

```
\DeclareOption{dvipsone}{\def\Gin@driver{dvipsone.def}}
\DeclareOption{dviwindo}{\ExecuteOptions{dvipsone}}
```

\ds@emtex \ds@dviwin Freely available drivers for PCs.

```
\DeclareOption{emtex}{\def\Gin@driver{emtex.def}}
\DeclareOption{dviwin}{\def\Gin@driver{dviwin.def}}
```

\ds@oztex The OzTEX system for a Macintosh. Since release 3 of OzTEX, merge with dvips back end.

```
\DeclareOption{oztex}{\ExecuteOptions{dvips}}
```

\ds@textures Blue sky's Textures system on a Macintosh.

```
\DeclareOption{textures}{\def\Gin@driver{textures.def}}
```

\ds@pctexps \ds@pctexwin \ds@pctexhp \ds@pctex32 The drivers for PTI's TEX system on PCs.

```
\DeclareOption{pctexps}{\def\Gin@driver{pctexps.def}}
\DeclareOption{pctexwin}{\def\Gin@driver{pctexwin.def}}
\DeclareOption{pctexhp}{\def\Gin@driver{pctexhp.def}}
\DeclareOption{pctex32}{\def\Gin@driver{pctex32.def}}
```

\ds@truetex \ds@tcidvi The drivers of the Kinch TEX system on PCs, and its version with extra \special handling dll's as shipped with TCI's Scientific Word.

```
\DeclareOption{truetex}{\def\Gin@driver{truetex.def}}
\DeclareOption{tcidvi}{\def\Gin@driver{tcidvi.def}}
```

\ds@vtex VTEX driver.

```
\DeclareOption{vtex}{\def\Gin@driver{vtex.def}}
```

\ds@dvi2ps \ds@dvialw \ds@dvilaser \ds@dvitops \ds@psprint \ds@pubps \ds@ln Old, probably obsolete, drivers commented out. See the section on 'Driver support' in grfguide to see how to re-enable these options in color.cfg if you need them.

```
%\DeclareOption{dvi2ps}{\def\Gin@driver{dvi2ps.def}}
%\DeclareOption{dvialw}{\def\Gin@driver{dvialw.def}}
%\DeclareOption{dvilaser}{\def\Gin@driver{dvilaser.def}}
%\DeclareOption{dvitops}{\def\Gin@driver{dvitops.def}}
%\DeclareOption{psprint}{\def\Gin@driver{psprint.def}}
%\DeclareOption{pubps}{\def\Gin@driver{pubps.def}}
%\DeclareOption{ln}{\def\Gin@driver{ln.def}}
```

\ds@dvipsnames \ds@nodvipsnames By default the named color model has no pre-declared names. The dvipsnames option predeclares all the names in the color prologue of dvips. The dvips option automatically implies dvipsnames unless this choice is overruled with the nodvipsnames option. For other drivers, eg textures, you may use this option to explicitly request that these names be declared.

```
\DeclareOption{dvipsnames}{\def\c@lor@namefile{dvipsnam.def}}
\DeclareOption{nodvipsnames}{\let\c@lor@namefile\relax}
```

\ds@usenames The usenames option modifies the behavior of \DefineNamedColor so that it declares the same name as a "user's color" for use in a \color command, as well as a name in the named color model. The normal behavior is just to declare the name in the named color model.

```
\let\c@lor@usename\@gobble
```

```
\DeclareOption{usenames}{%
  \def\c@lor@usename#1{%
    \expandafter\color@named\csname\@backslashchar color@#1\endcsname{#1}}}
```

3 Using Colors

3.1 Declarative form

`\color` `\color{`*declared-color*`}` switches to the color *declared-color*, which must previously have been defined using `\definecolor`. This color will stay in effect until the end of the current TEX group.

`\color[`*model*`]{`*color-specification*`}` is similar to the above, but uses a color not declared by `\definecolor`. The allowed *model*'s vary depending on the driver. The syntax of the *color-specification* argument depends on the model.

```
\DeclareRobustCommand\color{%
  \@ifnextchar[\@undeclaredcolor\@declaredcolor}
```

`\@undeclaredcolor` Call the driver-dependent command `\color@`⟨*model*⟩ to define `\current@color`, then call `\set@color` to change the current color accordingly.

```
\def\@undeclaredcolor[#1]#2{%
  \@ifundefined{color@#1}%
    {\c@lor@error{model `#1'}}%
    {\csname color@#1\endcsname\current@color{#2}%
     \set@color}%
  \ignorespaces}
```

`\@declaredcolor` `\let \current@color` to the internal representation of the color if the color has been declared, otherwise generate an error. Finally call `\set@color` to effect the color change.

```
\def\@declaredcolor#1{%
  \@ifundefined{\@backslashchar color@#1}%
    {\c@lor@error{`#1'}}%
    {\expandafter\let\expandafter\current@color
     \csname\@backslashchar color@#1\endcsname
     \set@color}%
  \ignorespaces}
```

3.2 Command (Argument) Form

`\textcolor` `\textcolor{`*declared-color*`}{`*text*`}` and `\textcolor[`*model*`]{`*color-spec*`}{`*text*`}` are just alternative syntax for `\color`, in which the groups are added implicitly. Thus *text* appears in the specified color, but then the color reverts to its previous value. The naming is by analogy with `\textrm` (as opposed to `\rm` and `\rmfamily`) although it is slightly a misnomer as the command also works in math-mode. In contrast to `\textrm`, the command gobbles spaces at the begin of its argument, so `Hello\textcolor{red}{  World}` will output HelloWorld.

Since December 95, in fact this command has one other difference from `\color`. It calls `\leavevmode` to ensure the start of horizontal mode. Specifically this means that a construction such as

```
xxx\parbox[t]{1cm}{\textcolor{red}{a}.....
```

now works as expected, with the xxx and the red a lining up correctly.

```
\protected\def\textcolor#1#{\@textcolor{#1}}
\def\@textcolor#1#2#3{\protect\leavevmode{\color#1{#2}#3}}
```

3.3 Background (Page) Color

\pagecolor `\pagecolor`, which has the same argument forms as `\color`, specifies the background color for the current, and all following, pages. It is a global declaration which does not respect TEX groups.

```
\protected\def\pagecolor{%
  \begingroup
      \let\ignorespaces\endgroup
      \let\set@color\set@page@color
      \color}
```

\nopagecolor `\nopagecolor` (suggested by Heiko Oberdiek) removes any currently specified page color returning to the default transparent background. It is not yet supported by all driver options and so generates a warning if there is no definition in the driver file.

```
\protected\def\nopagecolor{%
  \@ifundefined{no@page@color}{%
    \PackageInfo{color}{\@backslashchar nopagecolor\space is not supported}%
  }{%
    \no@page@color
  }%
}
```

4 Defining Colors

\definecolor `\definecolor{`*name*`}{`*model*`}{`*color-spec*`}` defines the color *name*, which may then be used in subsequent `\color` or `\textcolor` commands to refer to a color specified by *color-spec* in the color model *model*.

`\definecolor` associates the *name* to a color in *model*. So `\color{`*name*`}` would check *name* then issue a `\special` for the color model *model*.

`\definecolor` just calls an internal macro that defines the color for a particular model. This macro must have been defined by the driver file that supports the requested *model*.

```
\protected\def\definecolor#1#2#3{%
  \@ifundefined{color@#2}%
    {\c@lor@error{model '#2'}}%
    {\@ifundefined{\@backslashchar color@#1}{}%
      {\PackageInfo{color}{Redefining color #1}}%
     \csname color@#2\expandafter\endcsname
         \csname\@backslashchar color@#1\endcsname{#3}}}
```

\DefineNamedColor Driver files may opt to define a 'named' color model. Placing color names rather than numeric specifications into the dvi file has certain advantages, in that post processing software can tune the color specifications to the particular technology being used, and more easily make color separations for that color. The disadvantage is that the driver must 'know' the color names.

The 'color1' drivers (dvips) currently ignore the specification of the color and, once a name is defined, just put the color name in the dvi file. For dvips, the header file `color.pro` is used to give the cmyk equivalents of the names.

The 'color2' drivers (textures) use a special postscript operator that takes both the name and the cmyk equivalent, so if the names are not being used, 'fall back' definitions in the cmyk model are available. These drivers also allow a numeric value to affect the 'density' of the color to use.

Drivers based on 'color3' do not support named colors at the dvi level, but to ease document portability, the named model is still defined, but the `\special`'s just put the color specifications, not the names, into the dvi file.

Normally after a color, say `JungleGreen`, has been declared with:
`\DefineNamedColor{named}{JungleGreen}{cmyk}{1,2,3,4}`
it is available in the 'named' color model, for instance by
`\color[named]{JungleGreen}`
A user may give a more convenient name, using
`\definecolor{mygreen}{named}{JungleGreen}`
If however you are happy with the original names, and want to use them directly, as in `\color{JungleGreen}` without specifying `[named]` all the time, just give the package option `usenames`, which will redefine `\DefineNamedColor`, so that the color name is declared as a user-color as well as a name in the 'named' model.

```
\protected\def\DefineNamedColor#1#2#3#4{%
  \@ifundefined{define@color@#1}%
    {\c@lor@error{model '#1'}}%
    {\@ifundefined{color@#3}%
      {\c@lor@error{model '#3'}}%
      {\@ifundefined{col@#2}{}%
        {\PackageInfo{color}{Redefining color #2 in named color model}}%
      \csname color@#3\endcsname\@tempa{#4}%
      \csname define@color@#1\endcsname{#2}\@tempa
      \c@lor@usename{#2}}}}
\@onlypreamble\DefineNamedColor
```

5 Color Switch

`\ifcolors@` This boolean can be tested by higher level macros that may want to alter their behavior if a monochrome driver is being used.

```
\newif\ifcolors@
\colors@true
```

6 Whatsit...

`\c@lor@special` Some drivers can not support all the features of this package. They should always put a ⟨*whatsit*⟩ in the current list though. The following macro has most of the features of `\special`, but does not put anything into the `dvi` file. It does write to the log file or the terminal (depending on the value of `#1`).

```
\def\c@lor@special#1#2{%
  \edef\@tempa{\write#1{#2}}\@tempa}
```

7 Processing Options

A local configuration file may declare more options. It should also make one driver option the default, by calling `\ExecuteOptions` with the appropriate option.

```
\InputIfFileExists{color.cfg}{}{}
```

After the options are processed, load the appropriate driver file. If a site wants a default driver (eg `dvips`) it just needs to put `\ExecuteOptions{dvips}` in a `color.cfg` file.

```
\ProcessOptions
```

```
\if!\Gin@driver!
  \PackageError{color}
    {No driver specified}
    {You should make a default driver option in a file \MessageBreak
     color.cfg\MessageBreak
     eg: \protect\ExecuteOptions{dvips}%
     }
\else
  \PackageInfo{color}{Driver file: \Gin@driver}
  \@ifundefined{ver@\Gin@driver}{\input{\Gin@driver}}{}
\fi
```

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

8 Default Color

`\normalcolor` Early versions of this package redefined `\reset@font` to reset the color as well. Current versions do not do this (since there are too many `\reset@font` commands hidden in strange places) and so they define a separate command, `\normalcolor`, to reset the color to the color in effect at the start of the document.

`\normalcolor` is defined (to `\relax`) in the LaTeX kernel, so it is safe to use this in macros that may possibly be used in conjunction with color. It will have no effect until the `color` package is also loaded.

```
\protected\def\normalcolor{\let\current@color\default@color\set@color}
```

`\default@color` Internal macro to store the 'default' color used by `\normalcolor`.

```
\AtBeginDocument{\let\default@color\current@color}
```

`\current@color` contains an internal representation of the color at this point in the document. (This can only be an approximation to the truth as the 'macro layer' of TeX does not know where the output routine is going to re-insert floats. This is why drivers must maintain their own stack of colors in order to fully support these commands.)

For dvips, the `\current@color` is something like '`Black`' or '`rgb 0 1 0`', but other packages should not rely on any particular format for this macro.

The driver file *must* initialize `\current@color` to a specification for Black. This initialisation can not occur here, as the possible color models (and thus the syntax for 'black') are not known at this point.

9 Higher Level Commands

With the basic color primitives specified above we may define a few higher level commands for colored boxes etc. This is still quite a low level and presumably packages and classes making use of color will define more appropriate document-level commands.

9.1 Color Block

`\color@block` `\color@block{`*width*`}{`*height*`}{`*depth*`}`
Should take up no space for TEX, but produce a block in the current color of the specified size. It is mainly used for producing box backgrounds.

The definition here works by selecting a color, and then drawing a TEX rule (unless `\ifcolors@false`). This allows the 'driver independent' color specials to be used. However it is defined using `\providecommand`, so that this file will not over-write any other definition of this command. A graphics package may want to define it using a special to produce (for example) a PostScript line. Producing the line in the `\special` has the advantage that on a preview that does not understand `\specials`, the line is automatically omitted, without needing to modify the source of the document (for instance by adding the `monochrome` option).

```
\def\color@block#1#2#3{%
  {\set@color\rlap{\ifcolors@\vrule\@width#1\@height#2\@depth#3\fi}}}
```

9.2 Colored Boxes

`\colorbox` `\colorbox` takes the same argument forms as `\textcolor`, but the color specifies the *background* color of the box.

```
\protected\def\colorbox#1#{\color@box{#1}}
```

`\color@box`

```
\def\color@box#1#2{\color@b@x\relax{\color#1{#2}}}
```

`\fcolorbox` `\fcolorbox` has an extra *color-spec* argument, and puts a frame of the first color around a box with a background specified by the second color. If an optional argument is given, it specifies the color model for both colors.

`\fcolorbox`

```
\protected\def\fcolorbox#1#{\color@fbox{#1}}
```

```
\def\color@fbox#1#2#3{%
  \color@b@x{\fboxsep\z@\color#1{#2}\fbox}{\color#1{#3}}}
```

`\color@b@x` Internal macro for `\colorbox` and `\fcolorbox`.

```
\long\def\color@b@x#1#2#3{%
 \leavevmode
 \setbox\z@\hbox{\kern\fboxsep{\set@color#3}\kern\fboxsep}%
 \dimen@\ht\z@\advance\dimen@\fboxsep\ht\z@\dimen@
 \dimen@\dp\z@\advance\dimen@\fboxsep\dp\z@\dimen@
 {#1{#2\color@block{\wd\z@}{\ht\z@}{\dp\z@}%
       \box\z@}}}
```

9.3 Providing \mathcolor

This is shared coded between different packages, so external.

```
\input{mathcolor.ltx}
```

10 Extra Groups

Turning on extra groups in the standard LaTeX commands, so that color commands are scoped correctly.

Like \normalcolor, the following five commands are defined in the kernel, with empty definitions (\relax). This means that they can be used to make macros in packages 'color safe'. The commands will not have any effect unless a user also uses this color package, when the 'active definitions' here will take effect and keep color commands correctly scoped.

\color@setgroup This is to be used in contexts (eg 'lrbox') where text is to be saved and used after some other, unknown, text that may contain color commands. A matching \color@endgroup should be used at the end of the text.

```
\def\color@setgroup{\begingroup\set@color}
```

\color@begingroup This is to be used at the start of contexts that may contain color commands, but where it is not necessary to save the current color. Examples of this are in the box commands of ltboxes.dtx where user-supplied text is saved internally in a box between \color@begingroup, \color@endgroup, but the box is used before any other color commands could intervene. A matching \color@endgroup should be used at the end of the text.

```
\let\color@begingroup\begingroup
```

\color@endgroup To be used to close the 'group' started by one of the above two commands. The \endgraf in its definition is required in the case of groups of text in vertical 'par' mode, but doesn't do any harm in horizontal 'LR' contexts. The \@endpefalse is required for the newer @endpe handling, again it is harmless if an older kernel is used (because there it was a local assignment).

```
\def\color@endgroup{\endgraf\ifvmode\@endpefalse\fi\endgroup}
```

\color@hbox To be used to open a 'colored hbox'

```
\def\color@hbox{\hbox\bgroup\color@begingroup}
```

\color@vbox To be used to open a 'colored vbox'

```
\def\color@vbox{\vbox\bgroup\color@begingroup}
```

\color@endbox To be used to close a 'colored hbox'

```
\def\color@endbox{\color@endgroup\egroup}
```

11 Predefining Colors

As long as the driver file has defined sufficient color models, we define a few colors, just to get people started.

black Black and white ‘colors’.

white
```
\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
```

red Additive primaries.

green
blue
```
\ifx\color@rgb\@undefined\else
  \definecolor{red}{rgb}{1,0,0}
  \definecolor{green}{rgb}{0,1,0}
  \definecolor{blue}{rgb}{0,0,1}
\fi
```

cyan Subtractive primaries.

magenta
yellow
```
\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
```

⟨/package⟩

12 And Finally

Restore Catcodes

```
\Gin@codes
\let\Gin@codes\relax
```