The **color** package*

D. P. Carlisle

2024/06/23

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 colour, for those `dvi` drivers that can produce coloured 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 v1.3e, last revised 2024/06/23.

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

```
16 \DeclareOption{monochrome}{%
17   \def\c@lor@error#1{\PackageInfo{color}{Undefined color #1}}%
18   \AtEndOfPackage{%
19     \colors@false
20     \def\set@color{%
21       \c@lor@special\m@ne
22         {color push \current@color}\aftergroup\reset@color}%
23     \def\reset@color{\c@lor@special\m@ne{color pop}}%
24     \def\set@page@color{%
25       \c@lor@special\m@ne{background \current@color}}%
26     \def\define@color#1#2{%
27       \c@lor@special\m@ne{define #1 #2}}}}
```

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

```
28 \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.)

```
29 \newif\ifGin@setpagesize\Gin@setpagesizetrue
30 \DeclareOption{setpagesize}{\Gin@setpagesizetrue}
31 \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`.

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

`\ds@dvipdf` Sergey Lesenko's dvipdf driver.

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

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

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

`\ds@dvipdfmx` The driver for the dvipdfmx project.

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

`\ds@pdftex` Han The Thanh's TeX variant.

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

`\ds@luatex` LuaTeX TeX variant.

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

`\ds@luatex` dvisvgm driver.

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

`\ds@xetex` Jonathan Kew's TeX variant.

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

2

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

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

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

```
44 \DeclareOption{emtex}{\def\Gin@driver{emtex.def}}
45 \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.

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

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

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

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

```
48 \DeclareOption{pctexps}{\def\Gin@driver{pctexps.def}}
49 \DeclareOption{pctexwin}{\def\Gin@driver{pctexwin.def}}
50 \DeclareOption{pctexhp}{\def\Gin@driver{pctexhp.def}}
51 \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.

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

`\ds@vtex`	VTeX driver.

```
54 \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.

```
55 %\DeclareOption{dvi2ps}{\def\Gin@driver{dvi2ps.def}}
56 %\DeclareOption{dvialw}{\def\Gin@driver{dvialw.def}}
57 %\DeclareOption{dvilaser}{\def\Gin@driver{dvilaser.def}}
58 %\DeclareOption{dvitops}{\def\Gin@driver{dvitops.def}}
59 %\DeclareOption{psprint}{\def\Gin@driver{psprint.def}}
60 %\DeclareOption{pubps}{\def\Gin@driver{pubps.def}}
61 %\DeclareOption{ln}{\def\Gin@driver{ln.def}}
```

`\ds@dvipsnames` `\ds@nodvipsnames`	By default the named colour model has no pre-declared names. The `dvipsnames` option predeclares all the names in the colour 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.

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

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

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

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

3   Using Colours

3.1   Declarative form

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

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

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

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

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

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

```
76 \def\@declaredcolor#1{%
77   \@ifundefined{\@backslashchar color@#1}%
78     {\c@lor@error{'#1'}}%
79     {\expandafter\let\expandafter\current@color
80      \csname\@backslashchar color@#1\endcsname
81      \set@color}%
82   \ignorespaces}
```

3.2   Command (Argument) Form

\textcolor \textcolor{*declared-colour*}{*text*} and \textcolor[*model*]{*colour-spec*}{*text*} are just alternative syntax for \color, in which the groups are added implicitly. Thus *text* appears in the specified colour, but then the colour 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.

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.

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

4

3.3 Background (Page) Colour

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

```
85 \protected\def\pagecolor{%
86   \begingroup
87       \let\ignorespaces\endgroup
88       \let\set@color\set@page@color
89       \color}
```

\nopagecolor \nopagecolor (suggested by Heiko Oberdiek) removes any currently specified page colour 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.

```
90 \protected\def\nopagecolor{%
91   \@ifundefined{no@page@color}{%
92     \PackageInfo{color}{\@backslashchar nopagecolor\space is not supported}%
93   }{%
94     \no@page@color
95   }%
96 }
```

4 Defining Colours

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

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

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

```
97 \protected\def\definecolor#1#2#3{%
98   \@ifundefined{color@#2}%
99     {\c@lor@error{model '#2'}}%
100    {\@ifundefined{\@backslashchar color@#1}{}%
101      {\PackageInfo{color}{Redefining color #1}}%
102     \csname color@#2\expandafter\endcsname
103         \csname\@backslashchar color@#1\endcsname{#3}}}
```

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

The 'color1' drivers (dvips) currently ignore the specification of the colour and, once a name is defined, just put the colour 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 colour to use.

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

Normally after a colour, say `JungleGreen`, has been declared with:
`\DefineNamedColor{named}{JungleGreen}{cmyk}{1,2,3,4}`
it is available in the 'named' colour 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 colour name is declared as a user-colour as well as a name in the 'named' model.

```
104 \protected\def\DefineNamedColor#1#2#3#4{%
105   \@ifundefined{define@color@#1}%
106     {\c@lor@error{model '#1'}}%
107     {\@ifundefined{color@#3}%
108       {\c@lor@error{model '#3'}}%
109       {\@ifundefined{col@#2}{}%
110         {\PackageInfo{color}{Redefining color #2 in named color model}}%
111       \csname color@#3\endcsname\@tempa{#4}%
112       \csname define@color@#1\endcsname{#2}\@tempa
113       \c@lor@usename{#2}}}}
114 \@onlypreamble\DefineNamedColor
```

5   Colour Switch

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

```
115 \newif\ifcolors@
116 \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`).

```
117 \def\c@lor@special#1#2{%
118   \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.

```
119 \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.

```
120 \ProcessOptions

121 \if!\Gin@driver!
122   \PackageError{color}
123     {No driver specified}
124     {You should make a default driver option in a file \MessageBreak
125      color.cfg\MessageBreak
126      eg: \protect\ExecuteOptions{dvips}%
127      }
128 \else
129   \PackageInfo{color}{Driver file: \Gin@driver}
130   \@ifundefined{ver@\Gin@driver}{\input{\Gin@driver}}{}
131 \fi

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

8   Default Colour

\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 colour to the colour 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 colour. It will have no effect until the color package is also loaded.

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

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

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

\current@color contains an internal representation of the colour 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 colours 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* initialise \current@color to a specification for Black. This initialisation can not occur here, as the possible colour models (and thus the syntax for 'black') are not known at this point.

9   Higher Level Commands

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

9.1 Colour Block

`\color@block` `\color@block{`*width*`}{`*height*`}{`*depth*`}`

Should take up no space for TeX, but produce a block in the current colour of the specified size. It is mainly used for producing box backgrounds.

The definition here works by selecting a colour, and then drawing a TeX rule (unless `\ifcolors@false`). This allows the 'driver independent' colour 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).

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

9.2 Coloured Boxes

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

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

`\color@box`

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

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

`\fcolorbox`

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

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

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

```
142 \long\def\color@b@x#1#2#3{%
143 \leavevmode
144 \setbox\z@\hbox{\kern\fboxsep{\set@color#3}\kern\fboxsep}%
145 \dimen@\ht\z@\advance\dimen@\fboxsep\ht\z@\dimen@
146 \dimen@\dp\z@\advance\dimen@\fboxsep\dp\z@\dimen@
147 {#1{#2\color@block{\wd\z@}{\ht\z@}{\dp\z@}%
148     \box\z@}}}
```

9.3 Providing `\mathcolor`

This is shared coded between different packages, so external.

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

10   Extra Groups

Turning on extra groups in the standard LaTeX commands, so that colour 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 'colour safe'. The commands will not have any effect unless a user also uses this colour package, when the 'active definitions' here will take effect and keep colour 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 colour commands. A matching \color@endgroup should be used at the end of the text.

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

\color@begingroup   This is to be used at the start of contexts that may contain colour commands, but where it is not necessary to save the current colour. 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 colour commands could intervene. A matching \color@endgroup should be used at the end of the text.

```
151 \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).

```
152 \def\color@endgroup{\endgraf\@endpefalse\endgroup}
```

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

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

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

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

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

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

11   Predefining Colours

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

black   Black and white 'colours'.
white
```
156 \ifx\color@gray\@undefined
157   \ifx\color@rgb\@undefined
158   \else
159     \definecolor{black}{rgb}{0,0,0}
160     \definecolor{white}{rgb}{1,1,1}
161   \fi
```

```
162 \else
163   \definecolor{black}{gray}{0}
164   \definecolor{white}{gray}{1}
165 \fi
```

red Additive primaries.

green
blue
```
166 \ifx\color@rgb\@undefined\else
167   \definecolor{red}{rgb}{1,0,0}
168   \definecolor{green}{rgb}{0,1,0}
169   \definecolor{blue}{rgb}{0,0,1}
170 \fi
```

cyan Subtractive primaries.

magenta
yellow
```
171 \ifx\color@cmyk\@undefined\else
172   \definecolor{cyan}{cmyk}{1,0,0,0}
173   \definecolor{magenta}{cmyk}{0,1,0,0}
174   \definecolor{yellow}{cmyk}{0,0,1,0}
175 \fi
176 ⟨/package⟩
```

12   And Finally

Restore Catcodes
```
177 \Gin@codes
178 \let\Gin@codes\relax
```