

The `colortbl` package*

David Carlisle†

2024/02/20

Abstract

This package implements a flexible mechanism for giving colored ‘panels’ behind specified columns in a table. This package requires the `array` and `color` packages.

1 Introduction

This package is for coloring tables (i.e., giving colored panels behind column entries). In that it has many similarities with Timothy Van Zandt’s `colortab` package. The internal implementation is quite different though, also `colortab` works with the table constructs of other formats besides \LaTeX . This package requires \LaTeX (and its `color` and `array` packages).

First, a standard `tabular`, for comparison.

```
\begin{tabular}{|l|c|}
one&two\
three&four
\end{tabular}
```

one	two
three	four

2 The `\columncolor` command

The examples below demonstrate various possibilities of the `\columncolor` command introduced by this package. The vertical rules specified by `|` are kept in all the examples, to make the column positioning clearer, although possibly you would not want colored panels *and* vertical rules in practice.

The package supplies a `\columncolor` command, that should (only) be used in the argument of a `>` column specifier, to add a colored panel behind the specified column. It can be used in the main ‘preamble’ argument of `array` or `tabular`, and also in `\multicolumn` specifiers.

The basic format is:

```
\columncolor[color model]{color} [left overhang] [right overhang]
```

The first argument (or first two if the optional argument is used) are standard `color` package arguments, as used by `\color`.

The last two arguments control how far the panel overlaps past the widest entry in the column. If the *right overhang* argument is omitted then it defaults to

*This file has version number v1.0g, last revised 2024/02/20.

†Report issues to <https://github.com/davidcarlisle/dpctex/issues>

left overhang. If they are both omitted they default to `\tabcolsep` (in tabular) or `\arraycolsep` (in array).

If the overhangs are both set to `0pt` then the effect is:

```
|>{\columncolor[gray]{.8}[0pt]}1|
>{\color{white}%
\columncolor[gray]{.2}[0pt]}1|
```

one	two
three	four

The default overhang of `\tabcolsep` produces:

```
|>{\columncolor[gray]{.8}}1|
>{\color{white}%
\columncolor[gray]{.2}}1|
```

one	two
three	four

You might want something between these two extremes. A value of `.5\tabcolsep` produces the following effect

```
|>{\columncolor[gray]{.8} [.5\tabcolsep]}1|
>{\color{white}%
\columncolor[gray]{.2} [.5\tabcolsep]}1|
```

one	two
three	four

This package should work with most other packages that are compatible with the array package syntax. In particular it works with `longtable` and `dcolumn` as the following example shows.

Before starting give a little space: `\setlength\minrowclearance{2pt}`

A long table example		
First two columns	p-type	Third column
		D-type (dcolumn)
P-column	and another one	12:34
Total	(wrong)	100:6
Some long text in the first column	bbb	1:2
aaa	and some long text in the second column	1:345
Total	(wrong)	100:6
aaa	bbb	1:345
Note that the colored rules in all columns stretch to accomodate large entries in one column.	bbb	1:345
Continued...		

A long table example (continued)		
First two columns p-type		Third column D-type (dcolumn)
aaa	bbb	100
aaa	Depending on your driver you may get unsightly gaps or lines where the 'screens' used to produce different shapes interact badly. You may want to cause adjacent panels of the same color by specifying a larger overhang or by adding some negative space (in a <code>\noalign</code> between rows.	12·4
aaa	bbb	45·3

The End

This example shows rather poor taste but is quite colorful! Inspect the source file, `colortbl.dtx`, to see the full code for the example, but it uses the following column types.

```

\newcolumnntype{A}{%
  >{\color{white}\columncolor{red} [.5\tabcolsep]%
    \raggedright}%
  p{2cm}}
\newcolumnntype{B}{%
  >{\columncolor{blue} [.5\tabcolsep]%
    \color{yellow}\raggedright}
  p{3cm}}
\newcolumnntype{C}{%
  >{\columncolor{yellow} [.5\tabcolsep]}%
  D{.}\dot{3.3}}
\newcolumnntype{E}{%
  >{\large\bfseries
    \columncolor{cyan} [.5\tabcolsep]}c}
\newcolumnntype{F}{%
  >{\color{white}
    \columncolor{magenta} [.5\tabcolsep]}c}
\newcolumnntype{G}{%
  >{\columncolor[gray]{0.8} [.5\tabcolsep] [\tabcolsep]}1}
\newcolumnntype{H}{>{\columncolor[gray]{0.8}}1}
\newcolumnntype{I}{%
  >{\columncolor[gray]{0.8} [\tabcolsep] [.5\tabcolsep]}%

```

D{.}{\cdot}{3.3}}

3 Using the ‘overhang’ arguments for `tabular*`

The above is all very well for `tabular`, but what about `tabular*`?

Here the problem is rather harder. Although TeX’s `\leader` mechanism which is used by this package to insert the ‘stretchy’ colored panels is rather like *glue*, the `\tabskip` glue that is inserted between columns of `tabular*` (and `longtable` for that matter) has to be ‘real glue’ and not ‘leaders’.

Within limits the overhang options may be used here. Consider the first table example above. If we use `tabular*` set to 3 cm with a preamble setting of

```
\begin{tabular*}{3cm}{%
@{\extracolsep{\fill}}
>{\columncolor[gray]{.8}[0pt][20mm]}1
>{\columncolor[gray]{.8}[5mm][0pt]}1
@{}}
```

one	two
three	four

Changing the specified width to 4 cm works, but don’t push your luck to 5 cm...

one	two	one	two
three	four	three	four

4 The `\rowcolor` command

As demonstrated above, one may change the color of specified rows of a table by the use of `\multicolumn` commands in each entry of the row. However if your table is to be marked principally by *rows*, you may find this rather inconvenient. For this reason a new mechanism, `\rowcolor`, has been introduced¹.

`\rowcolor` takes the same argument forms as `\columncolor`. It must be used at the *start* of a row. If the optional overhang arguments are not used the overhangs will default to the overhangs specified in any `\columncolor` commands for that column, or `\tabcolsep` (`\arraycolsep` in `array`).

If a table entry is in the scope of a `\columncolor` specified in the table preamble, and also a `\rowcolor` at the start of the current row, the color specified by `\rowcolor` will take effect. A `\multicolumn` command may contain `>{\rowcolor...}` which will override the default colors for both the current row and column.

```
\begin{tabular}{|l|c|}
\rowcolor[gray]{.9}
one&two\\
\rowcolor[gray]{.5}
three&four
\end{tabular}
```

one	two
three	four

5 The `\rowcolors` command

The `\rowcolors` command and its documentation originate in the `xcolor` package by Dr. Uwe Kern.

`\rowcolors` [*commands*]{*row*}{*odd-row color*}{*even-row color*}

`\rowcolors*` [*commands*]{*row*}{*odd-row color*}{*even-row color*}

One of these commands has to be executed *before* a table starts. *row* tells the number of the first row which should be colored according to the *odd-row color* and *even-row color* scheme. Each of the color arguments may also be left empty (= no color). In the starred version, *commands* are ignored in rows with inactive *rowcolors status* (see below), whereas in the non-starred version, *commands* are applied to every row of the table. Such optional commands may be `\hline` or `\noalign{stuff}`.

`\showrowcolors` The *rowcolors status* is activated (i.e., use coloring scheme) by default and/or `\hiderowcolors` `\showrowcolors`, it is inactivated (i.e., ignore coloring scheme) by the command `\rownum` `\hiderowcolors`. The counter `\rownum` (or L^AT_EX counter `rownum`) may be used within such a table to access the current row number.

At the present time, the `rownum` counter is only incremented in tables using `\rowcolors`.

```

\rowcolors[\hline]{3}{green}{yellow} \arrayrulecolor{red}
\begin{tabular}{ll}
test & row \therownum\\
test & row \therownum\\
test & row \therownum\\
test & row \therownum\\
\arrayrulecolor{black}
test & row \therownum\\
test & row \therownum\\
\rowcolor{blue}
test & row \therownum\\
test & row \therownum\\
\hiderowcolors
test & row \therownum\\
test & row \therownum\\
\showrowcolors
test & row \therownum\\
test & row \therownum\\
\multicolumn{1}{%
  >{\columncolor{red}}1}{test} & row \therownum\\
\end{tabular}

```

test row 1	test row 1
test row 2	test row 2
test row 3	test row 3
test row 4	test row 4
test row 5	test row 5
test row 6	test row 6
test row 7	test row 7
test row 8	test row 8
test row 9	test row 9
test row 10	test row 10
test row 11	test row 11
test row 12	test row 12
test row 13	test row 13

6 The `\cellcolor` command

A background color can be applied to a single cell of a table by beginning it with `\multicolumn{1}{>{\rowcolor...}`, (or `\columncolor` if no row-color is in effect) but this has some deficiencies: 1) It prevents data within the cell from triggering the coloration; 2) The alignment specification must be copied from the top of the tabular, which is prone to errors, especially for `p{}` columns; 3) `\multicolumn{1}` is just silly. Therefore, there is the `\cellcolor` command, which works like `\columncolor` and `\rowcolor`, but over-rides both of them; `\cellcolor` can be placed anywhere in the tabular cell to which it applies.

¹At some cost to the internal complexity of this package

7 Coloring rules.

So you want colored rules as well?

One could do vertical rules without any special commands, just use something like `!\color{green}\vline` where you'd normally use `|`. The space between `||` will normally be left white. If you want to color that as well, either increase the overhang of the previous column (to `\tabcolsep + \arrayrulewidth + \doublerulesep`) Or remove the inter rule glue, and replace by a colored rule of the required thickness. So

```
!\color{green}\vline}
@{\color{yellow}\vrule width \doublerulesep}
!\color{green}\vline}
```

Should give the same spacing as `||` but more color.

However coloring `\hline` and `\cline` is a bit more tricky, so extra commands are provided (which then apply to vertical rules as well).

8 \arrayrulecolor

`\arrayrulecolor` takes the same arguments as `\color`, and is a global declaration which affects all following horizontal and vertical rules in tables. It may be given outside any table, or at the start of a row, or in a `>` specification in a table preamble. You should note however that if given mid-table it only affects rules that are specified after this point, any vertical rules specified in the preamble will keep their original colors.

9 \doublerulesepcolor

Having colored your rules, you'll probably want something other than white to go in the gaps made by `||` or `\hline\hline`. `\doublerulesepcolor` works just the same way as `\arrayrulecolor`. The main thing to note that if this command is used, then `longtable` will not 'discard' the space between `\hline\hline` at a page break. (\TeX has a built-in ability to discard space, but the colored 'space' which is used once `\doublerulesep` is in effect is really a third rule of a different color to the two outer rules, and rules are rather harder to discard.)

```
\setlength\arrayrulewidth{2pt}\arrayrulecolor{blue}
\setlength\doublerulesep{2pt}\doublerulesepcolor{yellow}
\begin{tabular}{||l||c||}
  \hline\hline
  one&two\\
  three&four\\
  \hline\hline
\end{tabular}
```

one	two
three	four

10 More fun with \hhline

The above commands work with `\hhline` from the `hhline` package, however if `hhline` is loaded in addition to this package, a new possibility is added. You may use `>{...}` to add declarations that apply to the following - or = column

rule. In particular you may give `\arrayrulecolor` and `\doublerulesepcolor` declarations in this argument.

Most manuals of style warn against over use of rules in tables. I hate to think what they would make of the following rainbow example:



```

\newcommand\rainbowline[1]{%
\hline{%
>{\arrayrulecolor {red}\doublerulesepcolor[rgb]{.3,.3,1}}%
|#1:=%
>{\arrayrulecolor{orange}\doublerulesepcolor[rgb]{.4,.4,1}}%
=%
>{\arrayrulecolor{yellow}\doublerulesepcolor[rgb]{.5,.5,1}}%
=%
>{\arrayrulecolor {green}\doublerulesepcolor[rgb]{.6,.6,1}}%
=%
>{\arrayrulecolor {blue}\doublerulesepcolor[rgb]{.7,.7,1}}%
=%
>{\arrayrulecolor{indigo}\doublerulesepcolor[rgb]{.8,.8,1}}%
=%
>{\arrayrulecolor{violet}\doublerulesepcolor[rgb]{.9,.9,1}}%
=:#1|}%
}}
\arrayrulecolor{red}
\doublerulesepcolor[rgb]{.3,.3,1}%
\begin{tabular}{||*7>{\columncolor[gray]{.9}}c||}
\rainbowline{t}%
\arrayrulecolor{violet}\doublerulesepcolor[rgb]{.9,.9,1}
Richard&of&York&gave&battle&in&
\multicolumn{1}{>{\columncolor[gray]{.9}}c||}{vain}\
\rainbowline{}%
1&2&3&4&5&6&
\multicolumn{1}{>{\columncolor[gray]{.9}}c||}{7}\
\rainbowline{b}%
\end{tabular}

```

11 Less fun with `\cline`

Lines produced by `\cline` are colored if you use `\arrayrulecolor` but you may not notice as they are covered up by any color pannels in the following row. This is a ‘feature’ of `\cline`. If using this package you would probably better using the `-rule` type in a `\hhrule` argument, rather than `\cline`.

12 The `\minrowclearance` command

As this package has to box and measure every entry to figure out how wide to make the rules, I thought I may as well add the following feature. ‘Large’ entries in tables may touch a preceding `\hline` or the top of a color panel defined by this style. It is best to increase `\extrarowsep` or `\arraystretch` sufficiently to ensure this doesn’t happen, as that will keep the line spacing in the table regular. Sometimes however, you just want to L^AT_EX to insert a bit of extra space above a large entry. You can set the length `\minrowclearance` to a small value. (The height of a capital letter plus this value should not be greater than the normal height of table rows, else a very uneven table spacing will result.)

Donald Arseneau’s `tabls` packages provides a similar `\tablinesep`. I was going to give this the same name for compatibility with `tabls`, but that is implemented quite differently and probably has different behaviour. So I’ll keep a new name for now.

13 The Code

```
1 (*package)
   Nasty hacky way used by all the graphics packages to include debugging code.
2 \edef\@tempa{%
3   \noexpand\AtEndOfPackage{%
4     \catcode'\noexpand^^A\the\catcode'\^^A\relax}}
5 \@tempa
6 \catcode'\^^A=\catcode'\%
7 \DeclareOption{debugshow}{\catcode'\^^A=9 }
   All the other options are handled by the color package.
8 \DeclareOption*{\PassOptionsToPackage\CurrentOption{color}}
9 \ProcessOptions
```

I need these so load them now. Actually Mark Wooding’s `mdwtab` package could probably work instead of `array`, but currently I assume `array` package internals so...

```
10 \RequirePackage{array,color}
\@classz First define stub for new array package code.
11 \ifx\do@row@strut\undefined\let\do@row@strut\relax\fi
   \@classz is the main function in the array package handling of primitive column types: It inserts the code for each of the column specifiers, ‘clrpmb’. The other classes deal with the other preamble tokens such as ‘@’ or ‘>’.
12 \def\@classz{\@classx
13   \@tempcnta \count@
14   \prepnext@tok
```

At this point the color specification for the background panel will be in the code for the ‘`>`’ specification of this column. This is saved in `\toks\@temptokena` but `array` will insert it too late (well it would work for `c`, but not for `p`) so fish the color stuff out of that token register by hand, and then insert it around the entry.

Of course this is a terrible hack. What is really needed is a new column type that inserts stuff in the right place (rather like `!` but without the spacing that that does). The `\newcolumntype` command of `array` only adds ‘second class’

column types. The re-implementations of `\newcolumn` in my `blkarray` or Mark Wooding's `mdwtab` allow new 'first class' column types to be declared, but stick with `array` for now. This means we have to lift the stuff out of the register before the register gets emptied in the wrong place.

```
15 \expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil
```

Save the entry into a box (using a double group for color safety as usual).

```
16 \@addtopreamble{%
17   \setbox\z@\hbox\bgroup\bgroup
18     \CT@everycr{}%
19     \ifcase \@cnum
```

`c` code: This used to use twice as much glue as `l` and `r` (1fil on each side). Now modify it to use 1fil total. Also increase the order from 1fil to 1fill to dissuade people from putting stretch glue in table entries.

```
20   \hskip\stretch{.5}\kern\z@
21   \d@llarbegin
22   \insert@column
23   \d@llarend\do@row@strut\hskip\stretch{.5}\or
```

`l` and `r` as before, but using fill glue.

```
24   \d@llarbegin \insert@column \d@llarend\do@row@strut \hfill
25   \or
26   \hfill\kern\z@ \d@llarbegin \insert@column \d@llarend\do@row@strut
27   \or
```

`m`, `p` and `b` as before, but need to take account of array package update.

```
28   \ifx\ar@align@mcell\@undefined
29     $\vcenter
30     \@startpbox{\@nextchar}\insert@column \@endpbox $
31   \else
32     \setbox\ar@mcellbox\vbox
33     \@startpbox{\@nextchar}\insert@column \@endpbox
34     \ar@align@mcell
35     \do@row@strut
36   \fi
37   \or
38   \vtop \@startpbox{\@nextchar}\insert@column \@endpbox\do@row@strut
39   \or
40   \vbox \@startpbox{\@nextchar}\insert@column \@endpbox\do@row@strut
41   \fi
```

Close the box register assignment.

```
42 \egroup\egroup
```

The main new stuff.

```
43 \begingroup
```

Initialise color command and overhands.

```
44 \CT@setup
```

Run any code resulting from `\columncolor` commands.

```
45 \CT@column@color
```

Run code from `\rowcolor` (so this takes precedence over `\columncolor`).

```
46 \CT@row@color
```

Run code from `\cellcolor` (so this takes precedence over both `\columncolor` and `\rowcolor`).

```
47 \CT@cell@color
```

This is `\relax` unless one of the three previous commands has requested a color, in which case it will be `\CT@@do@color` which will insert `\leaders` of appropriate color.

```
48 \CT@do@color
```

```
49 \endgroup
```

Nothing to do with color this bit, since we are boxing and measuring the entry anyway may as well check the height, so that large entries don't bump into horizontal rules (or the top of the color panels).

```
50 \@tempdima\ht\z@
```

```
51 \advance\@tempdima\minrowclearance
```

```
52 \vrule\@height\@tempdima\@width\z@
```

It would be safer to leave this boxed, but unboxing allows some flexibility. However the total glue stretch should either be finite or fill (which will be ignored). There may be fill glue (which will not be ignored) but it should *total ofill*. If this box contributes fill glue, then the leaders will not reach the full width of the entry. In the case of `\multicolumn` entries it is actually possible for this box to contribute *shrink* glue, in which case the colored panel for that entry will be too wide. Tough luck.

```
53 \unhbox\z@}%
```

```
54 \prepnext@tok}
```

`\CT@setup` Initialise the overhang lengths and the color command.

```
55 \def\CT@setup{%
```

```
56 \@tempdimb\col@sep
```

```
57 \@tempdimc\col@sep
```

```
58 \def\CT@color{%
```

```
59 \global\let\CT@do@color\CT@@do@color
```

```
60 \color}}
```

`\CT@@do@color` The main point of the package: Add the color panels.

Add a leader of the specified color, with natural width the width of the entry plus the specified overhangs and `1fill` stretch. Surround by negative kerns so total natural width is not affected by overhang.

```
61 \def\CT@@do@color{%
```

```
62 \global\let\CT@do@color\relax
```

```
63 \@tempdima\wd\z@
```

```
64 \advance\@tempdima\@tempdimb
```

```
65 \advance\@tempdima\@tempdimc
```

```
66 \kern-\@tempdimb
```

```
67 \leaders\vrule
```

For quick debugging with `xdvi` (which can't do colors). Limit the size of the rule, so I can see the text as well.

```
68 ^^A \@height\p@\@depth\p@
```

```
69 \hskip\@tempdima\@plus 1fill
```

```
70 \kern-\@tempdimc
```

Now glue to exactly compensate for the leaders.

```
71      \hskip-\wd\z@ \@plus -1fill }
```

`\CT@extract` Now the code to extract the `\columncolor` commands.

```
72 \def\CT@extract#1\columncolor#2#3\@nil{%  
73   \if!\noexpand#2%
```

! is a fake token inserted at the end.

```
74   \let\CT@column@color\@empty  
75   \else
```

If there was an optional argument

```
76   \if[\noexpand#2%  
77     \CT@extractb{#1}#3\@nil  
78   \else
```

No optional argument

```
79     \def\CT@column@color{%  
80       \CT@color{#2}}%  
81     \CT@extractd{#1}#3\@nil  
82     \fi  
83   \fi}
```

`\CT@extractb` Define `\CT@column@color` to add the right color, and save the overhang lengths. Finally reconstitute the saved ‘>’ tokens, without the color specification. First grab the color spec, with optional arg.

```
84 \def\CT@extractb#1#2]#3{%  
85   \def\CT@column@color{%  
86     \CT@color[#2]{#3}}%  
87   \CT@extractd{#1}}%
```

`\CT@extractd` Now look for left-overhang (default to `\col@sep`).

```
88 \def\CT@extractd#1{\@testopt{\CT@extracte{#1}}\col@sep}
```

`\CT@extracte` Same for right-overhang (default to left-overhang).

```
89 \def\CT@extracte#1[#2]{\@testopt{\CT@extractf{#1}[#2]}{#2}}
```

`\CT@extractf` Add the overhang info to `\CT@do@color`, for excuting later.

```
90 {\catcode'\!\active  
91 \gdef\CT@extractf#1[#2][#3]#4\columncolor#5\@nil{%  
92   \@tempdimb#2\relax  
93   \@tempdimc#3\relax  
94   \edef!\string!}%  
95   \edef\CT@column@color{%  
96     \CT@column@color  
97     \@tempdimb\the\@tempdimb\@tempdimc\the\@tempdimc\relax}%  
98   \toks\@tempcnta{#1#4}}%
```

`\CT@everycr` Steal `\everypar` to initialise row colors

```
99 \let\CT@everycr\everycr  
100 \newtoks\everycr  
101 \CT@everycr{\noalign{\global\let\CT@row@color\relax}\the\everycr}
```

```

\CT@start
102 \def\CT@start{%
103 \let\CT@arc@save\CT@arc@
104 \let\CT@drsc@save\CT@drsc@
105 \let\CT@row@color@save\CT@row@color
106 \let\CT@cell@color@save\CT@cell@color
107 \global\let\CT@cell@color\relax}

\CT@end
108 \def\CT@end{%
109 \global\let\CT@arc@\CT@arc@save
110 \global\let\CT@drsc@\CT@drsc@save
111 \global\let\CT@row@color@\CT@row@color@save
112 \global\let\CT@cell@color@\CT@cell@color@save}

\shortstack \shortstack
113 \gdef\@ishortstack#1{%
114 \CT@start\ialign{\mb@l {##}\unskip\mb@r\cr #1\cr}\CT@end\egroup}

\@tabarray array and tabular (delayed for delarray)
115 \AtBeginDocument{%
116 \expandafter\def\expandafter\@tabarray\expandafter{%
117 \expandafter\CT@start\@tabarray}}

\endarray
118 \def\endarray{%
119 \cr\egroup \egroup \@arrayright\gdef\@preamble{}\CT@end}

\multicolumn \multicolumn Patch \multicolumn to restore color settings. Done this way to
work wth different versions depending on the age of the array package.
120 \def\@tempa#1\@arstrut#2\relax{
121 \long\def\multicolumn##1##2##3{%
122 #1%
row@color
123 \let\CT@cell@color\relax
124 \let\CT@column@color\relax
125 \let\CT@do@color\relax
126 \@arstrut
127 #2}}
128 \expandafter\@tempa\multicolumn{#1}{#2}{#3}\relax
129 \let\@temp\relax

\@classvi Colored rules and rule separations.
130 \def\@classvi{\ifcase \@lastchclass
131 \@acol \or
132 \ifx\CT@drsc@\relax
133 \@addtopreamble{\hskip\doublerulesep}%
134 \else
135 \@addtopreamble{\CT@drsc@\vrule\@width\doublerulesep}}%
136 \fi\or
137 \@acol \or
138 \@classvii
139 \fi}

```

```

\doublerulesepcolor
140 \def\doublerulesepcolor#1#\CT@drs{#1}}

\CT@drs
141 \def\CT@drs#1#2{%
142 \ifdim\baselineskip=\z@\noalign\fi
143 {\gdef\CT@drsc@{\color#1{#2}}}}

\CT@drsc@
144 \let\CT@drsc@\relax

\arrayrulecolor
145 \def\arrayrulecolor#1#\CT@arc{#1}}

\CT@arc
146 \def\CT@arc#1#2{%
147 \ifdim\baselineskip=\z@\noalign\fi
148 {\gdef\CT@arc@{\color#1{#2}}}}

\CT@arc@
149 \let\CT@arc@\relax

\hline

\@arrayrule
150 \def\@arrayrule{\@addtopreamble {\CT@arc@\vline}}

\hline
151 \def\hline{%
152 \noalign{\ifnum0='}\fi
153 \let\hskip\vskip
154 \let\vrule\hrule
155 \let\@width\@height
156 {\CT@arc@\vline}%
157 \futurelet
158 \reserved@a\@xhline}

\@xhline
159 \def\@xhline{\ifx\reserved@a\hline
160 {\ifx\CT@drsc@\relax
161 \vskip
162 \else
163 \CT@drsc@\hrule\@height
164 \fi
165 \doublerulesep}%
166 \fi
167 \ifnum0='{ \fi}}

\cline \cline doesn't really work, as it comes behind the colored panels, but at least
make it the right color (the bits you can see, anyway).
168 \def\@cline#1-#2\@nil{%
169 \omit
170 \@multicnt#1%

```

```

171 \advance\@multispan\m@ne
172 \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
173 \@multicnt#2%
174 \advance\@multicnt-#1%
175 \advance\@multispan\@ne
176 {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}%
177 \cr
178 \noalign{\vskip-\arrayrulewidth}}

```

`\minrowclearance` The row height fudge length.

```

179 \newlength\minrowclearance
180 \minrowclearance=0pt

```

`\@mkpream` While expanding the preamble array passes tokens through an `\edef`. It doesn't use `\protection` as it thinks it has full control at that point. As the redefinition above adds `\color`, I need to add that to the list of commands made safe.

```

181 \let\@mkpreamarray\@mkpream
182 \def\@mkpream{%
183     \let\CT@setup\relax
184     \let\CT@color\relax
185     \let\CT@do@color\relax
186     \let\color\relax
187     \let\CT@column@color\relax
188     \let\CT@row@color\relax
189     \let\CT@cell@color\relax
190     \@mkpreamarray}

```

`\CT@do@color` For similar reasons, need to make this non-expandable

```

191 \let\CT@do@color\relax

```

`\rowcolor`

```

192 \def\rowcolor{%
193     \noalign{\ifnum0='}\fi
194     \global\let\CT@do@color\CT@do@color
195     \@ifnextchar[\CT@rowa\CT@rowb}

```

`\CT@rowa`

```

196 \def\CT@rowa[#1]#2{%
197     \gdef\CT@row@color{\CT@color[#1]{#2}}%
198     \CT@rowc}

```

`\CT@rowb`

```

199 \def\CT@rowb#1{%
200     \gdef\CT@row@color{\CT@color{#1}}%
201     \CT@rowc}

```

`\CT@rowc`

```

202 \def\CT@rowc{%
203     \@ifnextchar[\CT@rowd{\ifnum'={0}\fi}}

```

`\CT@rowd`

```

204 \def\CT@rowd[#1]{\@testopt{\CT@rowe[#1]}{#1}}

```

```

\CT@rowe
205 \def\CT@rowe[#1][#2]{%
206 \@tempdimb#1%
207 \@tempdimc#2%
208 \xdef\CT@row@color{%
209 \expandafter\noexpand\CT@row@color
210 \@tempdimb\the\@tempdimb
211 \@tempdimc\the\@tempdimc
212 \relax}%
213 \ifnum0='{\fi}}

\@ifxempty {<arg>}{<empty>}{<non-empty>}
Tests without expanding, whether the argument {<arg>} is empty and executes
the following code accordingly; {<arg>} must not start with the token \XC@@. Can
also be used within \edef.
214 \def\@ifxempty#1{\@ifxempty#1\@ifxempty\XC@@}
215 \def\@ifxempty#1#2\XC@@
216 {\ifx#1\@ifxempty
217 \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}

\rowcolors [[<commands>]{<row>}{<odd-row color>}{<even-row color>}
\rowcolors* Defines alternating colors for the next tabular environment. Starting with row
<row>, odd and even rows get their respective colors. The color arguments
may also be left empty (= no color). Optional commands may be hline or
noalign{<stuff>}.
In the starred version, <commands> are ignored in rows with inactive rowcolors
status (see below), whereas in the non-starred version, <commands> are applied to
every row of the table.
218 \def\rowcolors
219 {\@ifstar{\@rowcmdfalse\rowc@lors}{\@rowcmdtrue\rowc@lors}}
220 \def\rowc@lors{\@testopt{\rowc@l@rs}{}}
221 \def\rowc@l@rs[#1]#2#3#4%
222 {\global\rownum=\z@
223 \global\@rowcolorstrue
224 \@ifxempty{#3}%
225 {\def\@oddrowcolor{\@norowcolor}}%
226 {\def\@oddrowcolor{\gdef\CT@row@color{\CT@color{#3}}}}%
227 \@ifxempty{#4}%
228 {\def\@evenrowcolor{\@norowcolor}}%
229 {\def\@evenrowcolor{\gdef\CT@row@color{\CT@color{#4}}}}%
230 \if@rowcmd
231 \def\@rowcolors
232 {#1\if@rowcolors
233 \noalign{\relax\ifnum\rownum<#2\@norowcolor\else
234 \ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi}%
235 \fi}%
236 \else
237 \def\@rowcolors
238 {\if@rowcolors
239 \ifnum\rownum<#2\noalign{\@norowcolor}\else
240 #1\noalign{\ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi}\fi
241 \fi}%

```

```

242 \fi
243 \CT@everycr{\@rowcolor\the\everycr}%
244 \ignorespaces}

245 \def\@rowcolor{\noalign{\global\advance\rownum\@ne}\@rowcolor}
246 \let\@rowcolor\@empty

\showrowcolors Switch coloring mode on/off.
\hiderowcolors 247 \def\showrowcolors{\noalign{\global\@rowcolorstrue}\@rowcolor}
248 \def\hiderowcolors{\noalign{\global\@rowcolorstrue\@norowcolor}}
249 \def\@norowcolor{\global\let\CT@rowcolor\relax}
250 \@norowcolor

\if@rowcolors
\if@rowcmd 251 \newif\if@rowcolors
252 \newif\if@rowcmd

\rownum Reserve a counter register. Also alias as a LATEX counter (but not via \newcounter
\c@rownum as should not be in the reset list.)
253 \@ifundefined{rownum}{%
254 \ifundefined{c@rownum}{%
255 {\newcount\rownum\let\c@rownum\rownum}%
256 {\let\rownum\c@rownum}%
257 }%
258 {\let\c@rownum\rownum}
259 \providecommand\therownum{\arabic{rownum}}

\cellcolor \cellcolor applies the specified color to just its own tabular cell. It is defined ro-
bust, but without using \DeclareRobustCommand or \newcommand{} [] [] because
those forms are not used elsewhere, and would not work in very old LATEX.
260 \edef\cellcolor{\noexpand\protect
261 \expandafter\noexpand\csname cellcolor \endcsname}
262 \@namedef{cellcolor }{%
263 \@ifnextchar[{\CT@cellc@firstofone}{\CT@cellc@gobble[]}%
264 }
265 \def\CT@cellc#1[#2]#3{%
266 \expandafter\gdef\expandafter\CT@cellc@color\expandafter{%
267 \expandafter\CT@color#1{[#2]}{#3}%
268 \global\let\CT@cellc@color\relax
269 }}
270 \global\let\CT@cellc@color\relax

\DC@endright dcolumn support. the D column sometimes internally converts a c column to an r
one by squashing the supplied glue. This is bad news for this package, so redefine
it to add negative glue to one side and positive to the other to keep the total added
zero.
271 \AtBeginDocument{%
272 \def\@tempa{\$ \hfil \egroup \box \z@ \box \tw@}%
273 \ifx\@tempa\DC@endright

New version of dcolumn, only want to fudge it in the D{.}{.}{3} case, not
the new D{.}{.}{3.3} possibility. \hfill has already been inserted, so need to
remove 1fill's worth of stretch.
274 \def\DC@endright{%

```



```

275     $\hfil\egroup
276     \ifx\DC@rl\bgroup
277         \hskip\stretch{-.5}\box\z@\box\tw@\hskip\stretch{-.5}%
278     \else
279         \box\z@\box\tw@
280     \fi}%
281 \else
282     \def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
283     \ifx\@tempa\DC@endright

```

Old dcolumn code.

```

284     \def\DC@endright{%
285         $\hfil\egroup%
286         \hskip\stretch{.5}\box\z@\box\tw@\hskip\stretch{-.5}}%
287     \fi
288 \fi}

```

hhline support (almost the whole package, repeated, sigh).

```

289 \AtBeginDocument{%
290     \ifx\hhline\undefined\else
291     \def\HH@box#1#2{\vbox{%
292         \ifx\CT@drsc@relax\else
293             \global\dimen\thr@@\tw@\arrayrulewidth
294             \global\advance\dimen\thr@@\doublerulesep
295             {\CT@drsc@
296                 \hrule \@height\dimen\thr@@
297                 \vskip-\dimen\thr@@}%
298         \fi
299         \CT@arc@
300         \hrule \@height \arrayrulewidth \@width #1
301         \vskip\doublerulesep
302         \hrule \@height \arrayrulewidth \@width #2}}
303 \def\HH@loop{%
304     \ifx\@tempb'\def\next##1{\the\toks@\cr}\else\let\next\HH@let
305     \ifx\@tempb|\if@tempswa
306         \ifx\CT@drsc@relax
307             \HH@add{\hskip\doublerulesep}%
308         \else
309             \HH@add{{\CT@drsc@vrule\@width\doublerulesep}}%
310         \fi
311         \fi\@tempswatrue
312         \HH@add{{\CT@arc@vline}}\else
313     \ifx\@tempb:\if@tempswa
314         \ifx\CT@drsc@relax
315             \HH@add{\hskip\doublerulesep}%
316         \else
317             \HH@add{{\CT@drsc@vrule\@width\doublerulesep}}%
318         \fi
319         \fi\@tempswatrue
320         \HH@add{\@tempc\HH@box\arrayrulewidth\arrayrulewidth\@tempc}\else
321     \ifx\@tempb#\if@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatrue
322         \HH@add{{\CT@arc@vline\copy\@ne\@tempc\hrule}}\else
323     \ifx\@tempb~\@tempswafalse
324         \if@firststamp\@firststampfalse\else\HH@add{&\omit}\fi

```

```

325         \ifx\CT@drsc@\relax
326         \HH@add{\hfil}\else
327         \HH@add{%
328         \CT@drsc@\leaders\hrule\@height\HH@height\hfil}}%
329         \fi
330         \else
331 \ifx\@tempb-\@tempwafalse
332         \gdef\HH@height{\arrayrulewidth}%
333         \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
334         \HH@add{%
335         \CT@arc@\leaders\hrule\@height\arrayrulewidth\hfil}}%
336         \else
337 \ifx\@tempb=\@tempwafalse
338         \gdef\HH@height{\dimen\thr@@}%
339         \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
340         \HH@add
341         {\rlap{\copy\@ne}\leaders\copy\@ne\hfil\llap{\copy\@ne}}\else

```

Stop the backspacing for t and b, it messes up the underlying color.

```

342 \ifx\@tempb t\HH@add{%
343 \def\HH@height{\dimen\thr@@}%
344 \HH@box\doublerulesep\z@\@tempwafalse\else
345 \ifx\@tempb b\HH@add{%
346 \def\HH@height{\dimen\thr@@}%
347 \HH@box\z@\doublerulesep\@tempwafalse\else
348 \ifx\@tempb>\def\next##1##2{%
349 \HH@add{%
350 {\baselineskip\p@\relax
351 ##2%
352 \global\setbox\@ne\HH@box\doublerulesep\doublerulesep}}%
353 \HH@let!}\else
354 \ifx\@tempb\@sptoken\let\next\HH@spacelet\else
355 \PackageWarning{hhline}%
356 {\meaning\@tempb\space ignored in \noexpand\hhline argument%
357 \MessageBreak}%
358 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
359 \next}
360 \lowercase{\def\HH@spacelet}{\futurelet\@tempb \HH@loop}
361 \fi}

```

longtable support.

```

362 \AtBeginDocument{
363 \ifx\longtable\@undefined\else
364 \def\LT@@hline{%
365 \ifx\LT@next\hline
366 \global\let\LT@next\@gobble
367 \ifx\CT@drsc@\relax
368 \gdef\CT@LT@sep{%
369 \noalign{\penalty-\@medpenalty\vskip\doublerulesep}}%
370 \else
371 \gdef\CT@LT@sep{%
372 \multispan\LT@cols{%
373 \CT@drsc@\leaders\hrule\@height\doublerulesep\hfill}\cr}%
374 \fi

```

```

375     \else
376         \global\let\LT@next\empty
377         \gdef\CT@LT@sep{%
378             \noalign{\penalty-\@lowpenalty\vskip-\arrayrulewidth}}%
379     \fi
380     \ifnum0='{\fi}%
381     \multispan\LT@cols
382     {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}\cr
383     \CT@LT@sep
384     \multispan\LT@cols
385     {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}\cr
386     \noalign{\penalty\@M}%
387     \LT@next}
388     \fi}
389 \end{package}

```