

The colortbl package*

David Carlisle

2018/12/12

Abstract

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

1 Introduction

This package is for colouring tables (i.e., giving coloured 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 L^AT_EX. This package requires L^AT_EX (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 coloured 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 coloured 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]{colour}[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`.

*This file has version number v1.0d, last revised 2018/12/12.

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 *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		Third column
p-type	D-type (dcolumn)	
P-column	and another one	12.34
Total	(wrong)	100.6
Some long text in the first column aaa	bbb	1.2
	and some long text in the second column	1.345
Total	(wrong)	100.6
aaa	bbb	1.345
Note that the coloured rules in all columns stretch to accomodate large entries in one column.	bbb	1.345
Continued...		

A long table example (continued)		
First two columns		Third column
p-type		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 colour by specifying a larger overhang or by adding some negative space (in a <code>\noalign</code>	12·4
aaa	bbb	45·3
The End		

This example shows rather poor taste but is quite colourful! 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{.}{\cdot}{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 $\text{T}_{\text{E}}\text{X}$ ’s `\leader` mechanism which is used by this package to insert the ‘stretchy’ coloured 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 colour 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 colour specified by `\rowcolor` will take effect. A `\multicolumn` command may contain `>\rowcolor...` which will override the default colours for both the current row and column.

¹At some cost to the internal complexity of this package

```

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

```

one	two
three	four

5 The `\cellcolor` command

A background colour can be applied to a single cell of a table by beginning it with `\multicolumn{1}{>{\rowcolor...}`, (or `\columncolor` if no row-colour is in effect) but this has some deficiencies: 1) It prevents data within the cell from triggering the colouration; 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.

6 Colouring rules.

So you want coloured 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 colour 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 coloured 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 colour.

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

7 `\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 colours.

8 \doublerulesepcolor

Having coloured 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 coloured 'space' which is used once `\doublerulesep` is in effect is really a third rule of a different colour 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|l|c|}
\hline\hline
one&two\\
three&four\\
\hline\hline
\end{tabular}
```

one	two
three	four

9 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:

Richard	of	York	gave	battle	in	vain
1	2	3	4	5	6	7

```
\newcommand\rainbowline[1]{%
\hhline{%
>{\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{f}%
1&2&3&4&5&6&
\multicolumn{1}{>{\columncolor[gray]{.9}}c||}{7}\
\rainbowline{b}%
\end{tabular}

```

10 Less fun with `\cline`

Lines produced by `\cline` are coloured if you use `\arrayrulecolor` but you may not notice as they are covered up by any colour 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 `\hhline` argument, rather than `\cline`.

11 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 colour 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 \LaTeX 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.

12 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 }
8 \DeclareOption*{\PassOptionsToPackage\CurrentOption{color}}
9 \ProcessOptions

```

All the other options are handled by the color package.

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, `'clrpm'`. 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 colour 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 colour 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 `\newcolumntype` 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 colour 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 1fill 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     \dollarbegin

```



```

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 \or
25     \hfill\kern\z@ \d@llarbegin \insert@column \d@llarend\do@row@strut \or
m, p and b as before, but need to take account of array package update.
26     \ifx\ar@align@mcell\@undefined
27     $ \vcenter
28     \@startpbox{\@nextchar}\insert@column \@endpbox $
29     \else
30     \setbox\ar@mcellbox\vbox
31     \@startpbox{\@nextchar}\insert@column \@endpbox
32     \ar@align@mcell
33     \do@row@strut
34     \fi
35     \or
36     \vtop \@startpbox{\@nextchar}\insert@column \@endpbox\do@row@strut \or
37     \vbox \@startpbox{\@nextchar}\insert@column \@endpbox\do@row@strut
38     \fi
Close the box register assignment.
39 \egroup\egroup
The main new stuff.
40 \begingroup
Initialise colour command and overhands.
41 \CT@setup
Run any code resulting from \columncolor commands.
42 \CT@column@color
Run code from \rowcolor (so this takes precedence over \columncolor).
43 \CT@row@color
Run code from \cellcolor (so this takes precedence over both \columncolor
and \rowcolor).
44 \CT@cell@color
This is \relax unless one of the three previous commands has requested a colour,
in which case it will be \CT@@do@color which will insert \leaders of appropriate
colour.
45 \CT@do@color
46 \endgroup
Nothing to do with colour this bit, since we are boxing and measuring the en-
try anyway may as well check the height, so that large entries don't bump into
horizontal rules (or the top of the colour panels).
47     \@tempdima\ht\z@
48     \advance\@tempdima\minrowclearance
49     \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 0fill*. 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 coloured panel for that entry will be too wide. Tough luck.

```
50      \unhbox\z@}%
51  \prepnext@tok}
```

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

```
52 \def\CT@setup{%
53  \@tempdimb\col@sep
54  \@tempdimc\col@sep
55  \def\CT@color{%
56    \global\let\CT@do@color\CT@@do@color
57    \color}}
```

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

Add a leader of the specified colour, 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.

```
58 \def\CT@@do@color{%
59  \global\let\CT@do@color\relax
60      \@tempdima\wd\z@
61      \advance\@tempdima\@tempdimb
62      \advance\@tempdima\@tempdimc
63      \kern-\@tempdimb
64      \leaders\vrule
```

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

```
65  ^^A      \@height\p@\@depth\p@
66      \hskip\@tempdima\@plus 1fill
67      \kern-\@tempdimc
```

Now glue to exactly compensate for the leaders.

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

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

```
69 \def\CT@extract#1\columncolor#2#3\@nil{%
70  \if!#2%
    ! is a fake token inserted at the end.
71  \let\CT@column@color\@empty
72  \else
```

If there was an optional argument

```
73 \if[#2%  
74 \CT@extractb{#1}#3\@nil  
75 \else
```

No optional argument

```
76 \def\CT@column@color{%  
77 \CT@color{#2}}%  
78 \CT@extractd{#1}#3\@nil  
79 \fi  
80 \fi}
```

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

```
81 \def\CT@extractb#1#2]#3{%  
82 \def\CT@column@color{%  
83 \CT@color[#2]{#3}}%  
84 \CT@extractd{#1}}%
```

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

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

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

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

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

```
87 \def\CT@extractf#1[#2][#3]#4\columncolor#5\@nil{%  
88 \@tempdimb#2\relax  
89 \@tempdimc#3\relax  
90 \edef\CT@column@color{%  
91 \CT@column@color  
92 \@tempdimb\the\@tempdimb\@tempdimc\the\@tempdimc\relax}%  
93 \toks\@tempcnta{#1#4}}%
```

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

```
94 \let\CT@everycr\everycr  
95 \newtoks\everycr  
96 \CT@everycr{\noalign{\global\let\CT@row@color\relax}\the\everycr}
```

\CT@start

```
97 \def\CT@start{%  
98 \let\CT@arc@save\CT@arc@  
99 \let\CT@drsc@save\CT@drsc@  
100 \let\CT@row@color@save\CT@row@color  
101 \let\CT@cell@color@save\CT@cell@color  
102 \global\let\CT@cell@color\relax}
```

```

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

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

\@tabarray array and tabular (delayed for delarray)
110 \AtBeginDocument{%
111   \expandafter\def\expandafter\@tabarray\expandafter{%
112     \expandafter\CT@start\@tabarray}}

\endarray
113 \def\endarray{%
114   \cr\egroup \group \@arrayright\gdef\@preamble{}\CT@end}

\multicolumn \multicolumn
115 \long\def\multicolumn#1#2#3{%
116   \multispan{#1}\begingroup
117   \def\@addamp{\if@firstamp \@firstampfalse \else
118     \@preamerr 5\fi}%
119   \@mkpream{#2}\@addtopreamble\@empty
120   \endgroup
121   \def\@sharp{#3}%
122   \let\CT@cell@color\relax
   row@color
123   \let\CT@column@color\relax
124   \let\CT@do@color\relax
125   \@arstrut \@preamble
126   \null
127   \ignorespaces}

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

```

```

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

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

\CT@drsc@
142 \let\CT@drsc@\relax

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

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

\CT@arc@
147 \let\CT@arc@\relax

\hline

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

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

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

```

`\cline` `\cline` doesn't really work, as it comes behind the coloured panels, but at least make it the right colour (the bits you can see, anyway).

```
166 \def\@cline#1-#2\@nil{%
167   \omit
168   \@multicnt#1%
169   \advance\@multispan\m@ne
170   \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
171   \@multicnt#2%
172   \advance\@multicnt-#1%
173   \advance\@multispan\@ne
174   {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}%
175   \cr
176   \noalign{\vskip-\arrayrulewidth}}
```

`\minrowclearance` The row height fudge length.

```
177 \newlength\minrowclearance
178 \minrowclearance=0pt
```

`\@mkpream` While expanding the preamble array passes tokens through an `\edef`. It doesn't
`\@mkpreamarray` 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.

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

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

```
189 \let\CT@do@color\relax
```

`\rowcolor`

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

`\CT@rowa`

```
194 \def\CT@rowa[#1]#2{%
195   \gdef\CT@row@color{\CT@color[#1]{#2}}%
196   \CT@rowc}
```

`\CT@rowb`

```
197 \def\CT@rowb#1{%
198   \gdef\CT@row@color{\CT@color{#1}}%
199   \CT@rowc}
```

```

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

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

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

\cellcolor \cellcolor applies the specified colour to just its own tabular cell. It is de-
fined robust, but without using \DeclareRobustCommand or \newcommand{}[] []
because those forms are not used elsewhere, and would not work in very old LATEX.
212 \edef\cellcolor{\noexpand\protect
213 \expandafter\noexpand\csname cellcolor \endcsname}
214 \@namedef{cellcolor }{%
215 \@ifnextchar[{\CT@cellc\@firstofone}{\CT@cellc\@gobble[]}%
216 }
217 \def\CT@cellc#1[#2]#3{%
218 \expandafter\gdef\expandafter\CT@cell@color\expandafter{%
219 \expandafter\CT@color#1{[#2]}{#3}%
220 \global\let\CT@cell@color\relax
221 }}
222 \global\let\CT@cell@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.
223 \AtBeginDocument{%
224 \def\@tempa{\$hfil\egroup\box\z@\box\tw@}%
225 \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.
226 \def\DC@endright{%
227 \$hfil\egroup
228 \ifx\DC@rl\bgroup
229 \hskip\stretch{- .5}\box\z@\box\tw@\hskip\stretch{- .5}%
230 \else

```

```

231     \box\z@\box\tw@
232     \fi}%
233 \else
234     \def\@tempa{\$ \hfil\egroup\hfill\box\z@\box\tw@}%
235     \ifx\@tempa\DC@endright

```

Old dcolumn code.

```

236     \def\DC@endright{%
237         \$ \hfil\egroup%
238         \hskip\stretch{.5}\box\z@\box\tw@\hskip\stretch{-.5}}%
239     \fi
240 \fi}

```

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

```

241 \AtBeginDocument{%
242     \ifx\hhline\@undefined\else
243     \def\HH@box#1#2{\vbox{%
244         \ifx\CT@drsc@\relax\else
245             \global\dimen\thr@@\tw@\arrayrulewidth
246             \global\advance\dimen\thr@@\doublerulesep
247             {\CT@drsc@
248             \hrule \@height\dimen\thr@@
249             \vskip-\dimen\thr@@}%
250         \fi
251         \CT@arc@
252         \hrule \@height \arrayrulewidth \@width #1
253         \vskip\doublerulesep
254         \hrule \@height \arrayrulewidth \@width #2}}
255 \def\HH@loop{%
256     \ifx\@tempb'\def\next##1{\the\toks@\cr}\else\let\next\HH@let
257     \ifx\@tempb|\if@tempswa
258         \ifx\CT@drsc@\relax
259             \HH@add{\hskip\doublerulesep}%
260         \else
261             \HH@add{{\CT@drsc@\vrule\@width\doublerulesep}}%
262         \fi
263         \fi\@tempswatrue
264         \HH@add{{\CT@arc@\vline}}\else
265     \ifx\@tempb:\if@tempswa
266         \ifx\CT@drsc@\relax
267             \HH@add{\hskip\doublerulesep}%
268         \else
269             \HH@add{{\CT@drsc@\vrule\@width\doublerulesep}}%
270         \fi
271         \fi\@tempswatrue
272         \HH@add{\@tempc\HH@box\arrayrulewidth\arrayrulewidth\@tempc}\else
273     \ifx\@tempb#\if@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatrue
274         \HH@add{{\CT@arc@\vline\copy\@ne\@tempc\vline}}\else
275     \ifx\@tempb^\@tempswafalse
276         \if@firststamp\@firststampfalse\else\HH@add{&\omit}\fi

```



```

323             \CT@drsc@\leaders\hrule\@height\doublerulesep\hfill}\cr}%
324         \fi
325     \else
326         \global\let\LT@next\empty
327         \gdef\CT@LT@sep{%
328             \noalign{\penalty-\@lowpenalty\vskip-\arrayrulewidth}}%
329     \fi
330     \ifnum0={\fi}%
331     \multispan\LT@cols
332     {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}\cr
333     \CT@LT@sep
334     \multispan\LT@cols
335     {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}\cr
336     \noalign{\penalty\@M}%
337     \LT@next}
338 \fi}
339 </package>

```