The \texttt{colortbl} package\footnote{This file has version number v1.0d, last revised 2018/12/12.}

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 \texttt{array} and \texttt{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 \texttt{colortab} package. The internal implementation is quite different though, also \texttt{colortab} works with the table constructs of other formats besides \LaTeX. This package requires \LaTeX{} (and its \texttt{color} and \texttt{array} packages).

First, a standard \texttt{tabular}, for comparison.

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

2 The \texttt{\textbackslash{columncolor}} command

The examples below demonstrate various possibilities of the \texttt{\textbackslash{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 \texttt{\textbackslash{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 \texttt{array} or \texttt{tabular}, and also in \texttt{\textbackslash{multicolumn}} specifiers.

The basic format is:

\begin{verbatim}
\texttt{\textbackslash{columncolor}}[(\texttt{color model})]{{(colour)}} [\texttt{(left overhang)}][\texttt{(right overhang)}]
\end{verbatim}

The first argument (or first two if the optional argument is used) are standard \texttt{color} package arguments, as used by \texttt{\textbackslash{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 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:

\begin{verbatim}
|>{\columncolor[gray]{.8}{0pt}}l|
|>{\color{white} % \columncolor[gray]{.2}{0pt}}l|
\end{verbatim}

The default overhang of \tabcolsep produces:

\begin{verbatim}
|>{\columncolor[gray]{.8}}l|
|>{\color{white} % \columncolor[gray]{.2}}l|
\end{verbatim}

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

\begin{verbatim}
|>{\columncolor[gray]{.8}{.5\tabcolsep}}l|
|>{\color{white} % \columncolor[gray]{.2}{.5\tabcolsep}}l|
\end{verbatim}

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

<table>
<thead>
<tr>
<th>First two columns</th>
<th>Third column</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-type</td>
<td>D-type (dcolumn)</td>
</tr>
<tr>
<td>P-column and another one</td>
<td>12:34</td>
</tr>
<tr>
<td>Total (wrong)</td>
<td>100:6</td>
</tr>
<tr>
<td>Some long text in the first column bbb</td>
<td>1:2</td>
</tr>
<tr>
<td>and some long text in the second column</td>
<td>1:345</td>
</tr>
<tr>
<td>Total (wrong)</td>
<td>100:6</td>
</tr>
<tr>
<td>aaaa</td>
<td>bbb</td>
</tr>
<tr>
<td>Note that the coloured rules in all columns stretch to accommodate large entries in one column.</td>
<td>bbb</td>
</tr>
</tbody>
</table>

Continued...
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.

\newcolumntype{A}{% \color{white}\columncolor{red}[.5\tabcolsep]\raggedright\p{2cm}}
\newcolumntype{B}{% \color{blue}[.5\tabcolsep]\columncolor{yellow}\raggedright\p{3cm}}
\newcolumntype{C}{% \columncolor{yellow}[.5\tabcolsep]\D{.}{\cdot}{3.3}}
\newcolumntype{E}{% \large\bfseries\columncolor{cyan}[.5\tabcolsep]\c}
\newcolumntype{F}{% \color{white}\columncolor{magenta}[.5\tabcolsep]\c}
\newcolumntype{G}{% \columncolor{gray}[0.8][.5\tabcolsep]\c}
3 Using the ‘overhang’ arguments for \texttt{tabular*}

The above is all very well for \texttt{tabular}, but what about \texttt{tabular*}?

Here the problem is rather harder. Although \TeX's \texttt{leader} mechanism which is used by this package to insert the ‘stretchy’ coloured panels is rather like \texttt{glue}, the \texttt{tabskip} glue that is inserted between columns of \texttt{tabular*} (and \texttt{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 \texttt{tabular*} set to 3 cm with a preamble setting of

\begin{verbatim}
\begin{tabular*}{3cm}{% \\
@{}l@{}}
one & two \\
three & four \\
\end{tabular*}
\end{verbatim}

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

\begin{verbatim}
\begin{tabular*}{4cm}{% \\
@{}l@{}}
one & two \\
three & four \\
\end{tabular*}
\end{verbatim}

4 The \texttt{\rowcolor} command

As demonstrated above, one may change the colour of specified rows of a table by the use of \texttt{\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, \texttt{\rowcolor}, has been introduced\textsuperscript{1}.

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

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

\textsuperscript{1}At 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}

5 **The \texttt{\textbackslash cellcolor} command**

A background colour can be applied to a single cell of a table by beginning it with \multicolumn{1}{c}{\rowcolor{\ldots}} (or \texttt{\textbackslash 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) \texttt{\textbackslash multicolumn{1}{c}|} is just silly. Therefore, there is the \texttt{\textbackslash cellcolor} command, which works like \texttt{\textbackslash columncolor} and \texttt{\rowcolor}, but over-rides both of them; \texttt{\textbackslash 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 \texttt{\tabcolsep} + \texttt{\arrayrulewidth} + \texttt{\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 \texttt{\textbackslash hline} and \texttt{\textbackslash cline} is a bit more tricky, so extra commands are provided (which then apply to vertical rules as well).

7 **\texttt{\textbackslash arrayrulecolor}**

\texttt{\textbackslash arrayrulecolor} takes the same arguments as \texttt{\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 \texttt{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.)

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

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 \{\ldots\} 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:

\begin{verbatim}
\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}}%
\end{verbatim}
Richard of York gave battle in vain

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 panels in the following row. This is a ‘feature’ of \cline. If using this package you would probably better using the \hline 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 tabs provides a similar \tablinesep. I was going to give this the same name for compatibility with tabs, but that is implemented quite differently and probably has different behaviour. So I’ll keep a new name for now.

12 The Code
Nasty hacky way used by all the graphics packages to include debugging code.

```latex
\edef\@tempa{% 
\noexpand\AtEndOfPackage{% \catcode`\noexpand\^^A\the\catcode`\^^A\relax}}
\@tempa
\catcode`\^^A=\catcode`\%
\DeclareOption{debugshow}{\catcode`\^^A=9 } 
All the other options are handled by the color package.
\DeclareOption*{\PassOptionsToPackage\CurrentOption{color}}
\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...
\RequirePackage{array,color}
\@classz First define stub for new array package code.
\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 ‘>’.
\def\@classz{\@classx
\@tempcnta \count@
\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.
\expandafter\CT@extract\the\toks@temptokena\columncolor!\@nil
Save the entry into a box (using a double group for colour safety as usual).
\@addtopreamble{% 
\setbox\z@\hbox{\bgroup\bgroup
\CT@everycr{}
\ifcase \@chnum  
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.
\hskip\stretch{.5}\kern\z@
\d@llarbegin
8
1 and r as before, but using fill glue.

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

The main new stuff.

Initialise colour command and overhands.

Run any code resulting from \columncolor commands.

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

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

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.

Nothing to do with colour 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 colour panels).
It would be safer to leave this boxed, but unboxing allows some flexibility. However the total glue stretch should either be finite or fin (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@column@color\@empty
60 \@tempdima\wd\z@%s
61 \advance\@tempdima\@tempdimb
62 \advance\@tempdima\@tempdimc
63 \kern-\@tempdimb
64 \leaders\vrule
65 \^A
66 \@height\p@\@depth\p@
67 \hskip\@tempdima\@plus \ifill
68 \kern-\@tempdimc
Now glue to exactly compensate for the leaders.
69 \hskip\wd\z@ \@plus -\ifill }
\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@columncolor\@empty
72 \else
If there was an optional argument

\if\[#2%
\CT@extractb\#1\#3\@nil
\else
No optional argument
\def\CT@column@color{%\CT@color\#2}%
\CT@extractd\#1\#3\@nil
\fi
\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.

\def\CT@extractb#1#2\]#3{%
\def\CT@column@color{%\CT@color\#2}{#3}}%
\CT@extractd{#1}}%
\CT@extractb

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

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

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

\def\CT@extractf#1\[#2\][#3]#4\columncolor#5\@nil{%
\@tempdimb#2\relax
\@tempdimc#3\relax
\edef\CT@column@color{\CT@column@color\@tempdimb\the\@tempdimb\@tempdimc\the\@tempdimc}%
\toks\@tempcnta{#1#4}}%
\CT@extractf

\CT@everycr Steal \everypar to initialise row colours

\let\CT@everycr\everycr
\newtoks\everycr
\CT@everycr\noalign{\global\let\CT@row@color\relax}{\the\everycr}%
\CT@everycr

\CT@start

\def\CT@start{%
\let\CT@arc@save\CT@arc@
\let\CT@drsc@save\CT@drsc@
\let\CT@row@color@save\CT@row@color
\let\CT@cell@color@save\CT@cell@color
\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\crcr}\CT@end\egroup}

\@tabarray array and tabular (delayed for delarray)
110 \AtBeginDocument{%
111 \expandafter\def\expandafter\@tabarray\expandafter{%
112 \expandafter\CT@start\@tabarray}
113 \endarray
114 \def\endarray{%
115 \crcr \egroup \egroup \@arrayright\gdef\@preamble{}\CT@end}

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

\CT@drs
139 \def\CT@drsc#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\reserved@a\@xhline

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

\def\cline#1-#2\@nil{\omit\@multicnt#1\advance\@multispan\m@ne\ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi\@multicnt#2\advance\@multicnt-#1\advance\@multispan\@ne\{\CT@arc\leaders\hrule\@height\arrayrulewidth\hfill\cr\noalign{\vskip-\arrayrulewidth}}}

\minrowclearance The row height fudge length.
\newlength{\minrowclearance}
\minrowclearance=0pt

\@mkpream While expanding the preamble array passes tokens through an \edef. It doesn’t use \protect ion 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.
\let\@mkpreamarray\@mkpream
\def\@mkpream{\let\CT@setup\relax\let\CT@color\relax\let\CT@do@color\relax\let\color\relax\let\CT@column@color\relax\let\CT@row@color\relax\let\CT@cell@color\relax\@mkpreamarray}

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

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

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

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

\CT@rowc

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

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

\CT@rowe
\def\CT@rowe[#1][#2]{{% 
  \@tempdimb#1% 
  \@tempdimc#2% 
  \xdef\CT@row@color{\expandafter\noexpand\CT@row@color \@tempdimb\the\@tempdimb \@tempdimc\the\@tempdimc \relax}}% 
  \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 \TeX.
\edef\cellcolor{\noexpand\protect \expandafter\noexpand\csname cellcolor \endcsname}
\@namedef{cellcolor }{% 
  \@ifnextchar[\CT@cellc\@firstofone}{\CT@cellc\@gobble[\]}% 
}\CT@cellc#1[#2]#3{\expandafter\gdef\expandafter\CT@cell@color\expandafter{\expandafter\CT@color#1{[#2]}{#3} \global\let\CT@cell@color\relax}}\global\let\CT@cell@color\relax

\DC@endright\dcolumn\ support. the \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.
\AtBeginDocument{% 
  \def\@tempa{$\hfil\egroup\box\z@\box\tw@}% 
  \ifx\@tempa\DC@endright 
  \newversionof\dcolumn, only want to fudge it in the \D{.}{.}{3} case, not the new \D{.}{.}{3,3} possibility. \hfill \hskip has already been inserted, so need to remove 1fill’s worth of stretch.
\def\DC@endright{% 
  $\hfil\egroup \hskip\stretch{-.5}% 
  \ifx\DC@rf\bbox 
  \else 
  \hskip\stretch{-.5}
\def\DD@endright{$\hfil\egroup\hfill\box\z@\box\tw@}$
\iffalse\def\DC@endright{$\hfil\egroup\hfill\box\z@\box\tw@}$\fi}
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}\
\iffalse\def\DC@endright{$\hfil\egroup\hfill\box\z@\box\tw@}$\fi}
\def\DC@endright{\ifsame\@tempa\DC@endright\else
\hfill\box\tw@i
\let\DC@endright\DD@endright

Old dcolumn code.
\def\DC@endright{\ifsame\@tempa\DC@endright\else
\hfill\box\tw@i
\let\DC@endright\DD@endright

hhline support (almost the whole package, repeated, sigh).
\AtBeginDocument{%
\if\hhline\@undefined\else
\def\HH@box#1#2{\vbox{{\global\dimen\thr@@\tw@\arrayrulewidth
\global\advance\dimen\thr@@\doublerulesep
{\CT@drsc@
\hrule \@height\dimen\thr@@
\vskip-\dimen\thr@@}\
\CT@arc@\hrule \@height\arrayrulewidth \@width#1
\vskip\doublerulesep\hrule \@height\arrayrulewidth \@width#2}}}
\def\HH@loop{\if\@tempb\def\next##1{\the\toks@\cr}\else\let\next\HH@let
\if\@tempb|\if\@tempswa
\if\CT@drsc@\relax
\HH@add{\hskip\doublerulesep}\
\else
\HH@add{{\CT@drsc@\vrule\@width\doublerulesep}}\
\fi\@tempswatrue
\HH@add{{\CT@arc@\vline}}\else
\if\@tempb:\if\@tempswa
\HH@add{\hskip\doublerulesep}\fi\@tempswatrue
\HH@add{{\CT@arc@\vline\copy\@ne\@tempc\vline}}\else
\if\@tempb~\@tempswafalse
\if\@firstamp\@firstampfalse\else\HH@add{&\omit}\fi}
\else
\if\@tempswafalse
\if\@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
\else
\HH@add{{\CT@arc@\vline\copy\@one\@tempc\vline}}\else
\if\@tempswafalse
\HH@add{\@tempc}
\else
\HH@add{\hfill\box\tw@\hskip\stretch{-0.5}}%
Stop the backspacing for \texttt{t} and \texttt{b}, it messes up the underlying colour.

\AtBeginDocument{
\ifx\longtable\@undefined\else
  \def\LT@@hline{\ifx\LT@next\hline \global\let\LT@next\@gobble \else \gdef\CT@LT@sep{\noalign{\penalty-\@medpenalty\vskip\doublerulesep}} \fi}
\fi}

\longtable support.

\AtBeginDocument{
\ifx\longtable\@undefined\else
  \def\LT@hline{%
    \ifx\LT@next\hline \global\let\LT@next\@gobble \else \gdef\CT@LT@sep{\noalign{\penalty-\@medpenalty\vskip\doublerulesep}} \fi
  }
\fi}
\CT@drsc0\leaders\hrule\@height\doublerulesep\hfill\cr% 
\fi
\else
  \global\let\LT@next\empty
  \gdef\CT@LT@sep{%
    \noalign{\penalty-\@lowpenalty\vskip-\arrayrulewidth}}%
  \fi
⟨/package⟩