The \texttt{colortbl} package\footnote{This file has version number v1.0c, last revised 2018/05/02.}

David Carlisle

2018/05/02

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

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 \texttt{|} 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 \texttt{>} 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[\{color model\}][\{colour\}][\{left overhang\}][\{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}. 

\texttt{\begin{verbatim}
\begin{tabular}{|l|c|}
  one&two \\
  three&four
\end{tabular}
\end{verbatim}
The last two arguments control how far the panel overlaps past the widest entry in the column. If the \textit{right overhang} argument is omitted then it defaults to \textit{left overhang}. If they are both omitted they default to \texttt{\tabcolsep} (in \texttt{tabular}) or \texttt{\arraycolsep} (in \texttt{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 \texttt{\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\texttt{\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 \texttt{array} package syntax. In particular it works with \texttt{longtable} and \texttt{dcolumn} as the following example shows.

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

\begin{longtable}{|p{3cm}|p{2cm}|p{2cm}|p{2cm}|}
\hline
\textbf{A long table example} & First two columns & Third column & D-type (dcolumn) \\
\hline
\texttt{P-column} & \texttt{and another one} & & 12:34 \\
\texttt{Total} & & & 100:6 \\
\hline
\texttt{Some long text in the first column} & \texttt{bbb} & & 1:2 \\
\texttt{aaa} & \texttt{and some long text in the second column} & & 1:345 \\
\texttt{Total} & & & 100:6 \\
\hline
\texttt{aaa} & \texttt{bbb} & & 1:345 \\
\texttt{Note that the coloured rules in all columns stretch to accomodate large entries in one column.} & \texttt{bbb} & & 1:345 \\
\hline
\end{longtable}

Continued...
A long table example (continued)

<table>
<thead>
<tr>
<th>First two columns</th>
<th>Third column D-type (\texttt{dcolumn})</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-type</td>
<td>D-type (\texttt{dcolumn})</td>
</tr>
<tr>
<td>aaa</td>
<td>bbb</td>
</tr>
<tr>
<td>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 \texttt{noalign} between rows.</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>12.4</td>
</tr>
<tr>
<td>aaa</td>
<td>bbb</td>
</tr>
<tr>
<td>45.3</td>
<td></td>
</tr>
<tr>
<td>The End</td>
<td></td>
</tr>
</tbody>
</table>

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

\begin{verbatim}
\newcolumntype{A}{%r{\color{white}\columncolor{red}[.5\tabcolsep]\raggedright}p{2cm}}
\newcolumntype{B}{%r{\columncolor{blue}[.5\tabcolsep]\color{yellow}\raggedright}p{3cm}}
\newcolumntype{C}{%r{\columncolor{yellow}[.5\tabcolsep]}D{.}{\cdot}{3.3}}
\newcolumntype{E}{%r{\large\bfseries\columncolor{cyan}[.5\tabcolsep]}c}
\newcolumntype{F}{%r{\color{white}\columncolor{magenta}[.5\tabcolsep]}c}
\newcolumntype{G}{%r{\columncolor{gray}[0.8][.5\tabcolsep]\tabcolsep}l}
\end{verbatim}
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’ 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 \texttt{tabular*} set to 3 cm with a preamble setting of

```
\begin{tabular*}{3cm}{%}
  \extracolsep{\fill}\% \\
  >{\color{gray}\{0.8\}\{tabcolsep\}0.5\{tabcolsep\}}% \\
  \texttt{D\{.\}\{\texttt{c}\texttt{dott}\}3.3}}
\end{tabular*}
```

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

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\footnote{At some cost to the internal complexity of this package.}

\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} (\texttt{arraycolsep} 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.}
\begin{tabular}{|l|c|}
  \rowcolor{gray}{.9}  
  one & two \\
  \rowcolor{gray}{.5}  
  three & four \\
\end{tabular}

5  The \texttt{\cellcolor} command

A background colour can be applied to a single cell of a table by beginning it with \texttt{\multicolumn{1}{>{\rowcolor . . . , (or \texttt{\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 \texttt{p{}} columns; 3) \texttt{\multicolumn{1}} is just silly. Therefore, there is the \texttt{\cellcolor} command, which works like \texttt{\columncolor} and \texttt{\rowcolor}, but over-rides both of them; \texttt{\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 \texttt{!\{color\{green\}\vline}} where you’d normally use \texttt{|}. 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 + \arrayrulewidth + \doublerulesep}) Or remove the inter rule glue, and replace by a coloured rule of the required thickness. So

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

Should give the same spacing as || but more colour.

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

7  \texttt{\arrayrulecolor}

\texttt{\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 \texttt{\textbackslash doublerulesepcolor}

Having coloured your rules, you’ll probably want something other than white to
go in the gaps made by \textbackslash hline or \textbackslash hline. \texttt{\textbackslash doublerulesepcolor} works just
the same way as \texttt{\textbackslash arrayrulecolor}. The main thing to note that if this command
is used, then \texttt{\textbackslash longtable} will not ‘discard’ the space between \textbackslash hline at a
page break. (\TeX{} has a built-in ability to discard space, but the coloured ‘space’
which is used once \texttt{\textbackslash 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
\texttt{one} & \texttt{two} \\
\hline \texttt{three} & \texttt{four} \\
\hline\hline
\end{tabular}
\end{verbatim}

9 More fun with \texttt{\textbackslash hhline}

The above commands work with \texttt{\textbackslash hhline} from the \texttt{hhline} package, however if
\texttt{hhline} is loaded in addition to this package, a new possibility is added. You
may use \texttt{\{\ldots\}} to add declarations that apply to the following – or = column
rule. In particular you may give \texttt{\arrayrulecolor} and \texttt{\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\{0.3,0.3,1\}}\%
|\#1:=\%
>\{\arrayrulecolor {orange}\doublerulesepcolor {rgb\{0.4,0.4,1\}}\%
=\%
>\{\arrayrulecolor {yellow}\doublerulesepcolor {rgb\{0.5,0.5,1\}}\%
=\%
>\{\arrayrulecolor {green}\doublerulesepcolor {rgb\{0.6,0.6,1\}}\%
=\%
>\{\arrayrulecolor {blue}\doublerulesepcolor {rgb\{0.7,0.7,1\}}\%
\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 - 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.

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...

\classz First define stub for new array package code.

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.

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

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.
\"insert@column
\do@row@strut\hskip\stretch{.5}\or

l and r as before, but using fill glue.
\dollarbegin \insert@column \do@row@strut \hfill \or
\hfill\kern\z@ \dollarbegin \insert@column \do@row@strut \or

m, p and b as before, but need to take account of array package update.
\ife\ar@align@mcell\@undefined
\vcenter \else
\@startpbox{\@nextchar}\insert@column \@endpbox $ \ar@align@mcell
\setbox\ar@mcellbox\vbox \@startpbox{\@nextchar}\insert@column \@endpbox
\ar@align@mcell \do@row@strut
\fi
\or
\vtop \@startpbox{\@nextchar}\insert@column \@endpbox \do@row@strut
\vbox \@startpbox{\@nextchar}\insert@column \@endpbox \do@row@strut
\fi

Close the box register assignment.
\egroup\egroup

The main new stuff.
\begingroup

Initialise colour command and overhands.
\CT@setup

Run any code resulting from \columncolor commands.
\CT@column@color

Run code from \rowcolor (so this takes precedence over \columncolor).
\CT@row@color

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

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).
\@tempdima\ht\z@ \advance\@tempdima\minrowclearance \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 fil (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.

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

\CT@do@color\par
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.

\CT@extract\par
Now the code to extract the \columncolor commands.
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 ‘\textgreater’ 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@extractd Now look for left-overhang (default to \col@sep).
\def\CT@extractd#1{\@testopt{\CT@extracte{#1}}\col@sep}

\CT@extracte Same for right-overhang (default to left-overhang).
\def\CT@extracte#1\[#2\]{\@testopt{\CT@extractf{#1\[#2\]}{#2}}%

\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\relax}%
\toks\@tempcnta{#1#4}}%

\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@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
\def\CT@end{\global\let\CT@arc@\CT@arc@save
\global\let\CT@drsc@\CT@drsc@save
\global\let\CT@row@color\CT@row@color@save}
\shortstack
\gdef\@ishortstack#1{\CT@start\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\CT@end\egroup}
\@tabarray array and tabular (delayed for delarray)
\AtBeginDocument{%\expandafter\def\expandafter\@tabarray\expandafter{\CT@start\@tabarray}}
\endarray
\def\endarray{%\crcr\egroup\egroup\@arrayright\gdef\@preamble{}}
\multicolumn \multicolumn
\long\def\multicolumn#1#2#3{\multispan{#1}\begingroup\def\@addamp{\if@firstamp\@firstampfalse \else \@preamerr 5\fi}\def\@mkpream{#2}\@addtopreamble\@empty\endgroup\def\@sharp{#3}\let\CT@cell@color\relax\let\CT@column@color\relax\let\CT@do@color\relax\@arstrut \@preamble\null\ignorespaces}
\@classvi Coloured rules and rule separations.
\def\@classvi{\ifcase \@lastchclass \@acol \or \ifx\CT@drsc@\relax \@addtopreamble{\hskip\doublerulesep}% \else \@addtopreamble{{\CT@drsc@\vrule\@width\doublerulesep}}% \fi\or \@acol \or \@classvii \fi}
\doublerulesepcolor
138 \def\doublerulesepcolor#1#2{%CT\@drs{#1}{#2}}
\CT\@drs
139 \def\CT\@drs#1#2{%ifdim\baselineskip=\z@\noalign\fi
140 {\gdef\CT\@drsc@{\color#1{#2}}}}
\CT\@drsc@
141 \let\CT\@drsc@\relax
\arrayrulecolor
142 \def\arrayrulecolor#1#2{%CT\@arc{#1}{#2}}
\CT\@arc@
143 \let\CT\@arc@\relax
hline
\@arrayrule
144 \def\@arrayrule{\@addtopreamble {{\CT\@arc@\vline}}} hline
145 \def\hline{\noalign{\ifnum0='\fi}\let\hskip\vskip\let\vrule\hrule\let\@width\@height
146 {\CT\@arc@\vline}\futurelet\reserved@a\@xhline}
\@xhline
147 \def\@xhline{\ifx\reserved@a\hline\ifx\CT\@drsc@\relax
148 \vskip\else\CT\@drsc@\hrule\@height\doublerulesep\fi\fi\ifnum0='\fi}}
149 \def\hline{%\noalign{\ifnum0='\fi}\let\hskip\vskip\let\vrule\hrule
150 \let\vrule\hrule\let\@width\@height
151 {\CT\@arc@\vline}\%\futurelet\reserved@a\@xhline}
\@xhline
152 \def\@xhline{\ifx\reserved@a\hline{\ifx\CT\@drsc@\relax
153 \vskip\else\CT\@drsc@\hrule\@height\doublerulesep\fi\fi\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).

\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 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.
\expandafter\def\expandafter\@mkpream\expandafter#1\expandafter{%\expandafter
  \expandafter\let\expandafter\CT@setup\expandafter\relax
  \expandafter\let\expandafter\CT@color\expandafter\relax
  \expandafter\let\expandafter\CT@do@color\expandafter\relax
  \expandafter\let\expandafter\color\expandafter\relax
  \expandafter\let\expandafter\CT@column@color\expandafter\relax
  \expandafter\let\expandafter\CT@row@color\expandafter\relax
  \expandafter\let\expandafter\CT@cell@color\expandafter\relax
  \@mkpream{#1}}

@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 14
\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}}
\edef\cellcolor{\noexpand\protect\expandafter\noexpand\csname cellcolor \endcsname}
\@namedef{cellcolor }{\@ifnextchar[\CT@cellc\@firstofone}{\CT@cellc\@gobble}[\]}
\def\CT@cellc#1[#2]#3{\expandafter\gdef\expandafter\CT@cell@color\expandafter{\CT@color#1[#2]{#3}\global\let\CT@cell@color\relax}}
\global\let\CT@cell@color\relax
\DC@endright
dcolumn support. the \texttt{D} column sometimes internally converts a \texttt{c} column to an \texttt{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{$\hfil\egroup\box\z@\box\tw@$}}
\edef\DC@endright{$\hfil\egroup\ifx\DC@rl\bgroup\hskip\stretch{-.5}\box\z@\box\tw@\hskip\stretch{-.5}$\else\hskip\stretch{-.5}\box\z@\box\tw@\hskip\stretch{-.5}$\hfill\hspace{\stretch{-.5}}$\fi}}
Old \texttt{dcolumn} code.

\begin{verbatim}
def\DC@endright{%  
$hfil$\egroup%  
\hskip\stretch{.5}\box\z@\box\tw@\hskip\stretch{-.5}}%
\fi
\fi
\end{verbatim}

\texttt{hhline} support (almost the whole package, repeated, sigh).

\begin{verbatim}
\AtBeginDocument{%  
\if\hhline\@undefined\else
  \def\HH@box#1#2{\vbox{{%  
    \if\CT@drsc@\relax\else
      \global\dimen\thr@@\tw@\arrayrulewidth  
      \global\advance\dimen\thr@@\doublerulesep  
     \CT@drsc@
     \hrule\@height\dimen\thr@@  
     \vskip-\dimen\thr@@}  
    \fi
    \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@elax
        \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
      \HH@add{\CT@drsc@\vrule\@width\doublerulesep}%
    \else
      \if\@tempb##\if@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatrue
      \HH@add{\CT@drsc@\vrule\@width\doublerulesep}%
    \else
      \if\@tempb~\@tempswafalse
        \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
      \fi
    \fi
    \fi
  \fi\@tempswatrue
  \HH@add{\{\CT@arc@\vline\copy\@ne\@tempc\vline}}
  \if\@tempb:\if@tempswa
    \HH@add{\CT@drsc@\vrule\@width\doublerulesep}%
  \else
    \HH@add{\CT@drsc@\vrule\@width\doublerulesep}%
  \fi
  \fi
  \fi
  \fi
\immediate\write\@auxout{	he\toks@}
\end{verbatim}
Stop the backspacing for \texttt{t} and \texttt{b}, it messes up the underlying colour.

\begin{longtable}
\caption{\texttt{hhline} support.}
\end{longtable}
\CT@drsc\leaders\hrule\@height\doublerulesep\hfill\cr\fi
\else
\global\let\LT@next\empty
\gdef\CT@LT@sep{%
\noalign{\penalty-\@lowpenalty\vskip-\arrayrulewidth}}%
\fi
\ifnum0='{\fi}%
\multispan\LT@cols
{\CT@arc\leaders\hrule\@height\arrayrulewidth\hfill\cr
\CT@LT@sep
\multispan\LT@cols
{\CT@arc\leaders\hrule\@height\arrayrulewidth\hfill\cr
\noalign{\penalty\@M}%
\LT@next}
\fi}
{/package}