The \texttt{colortbl} package*

David Carlisle

1999/03/24

Abstract

This package implements a flexible mechanism for giving colored ‘panels’ behind specified columns in a table. This package requires the \texttt{array} and \texttt{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 \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|}
  one & two \\
  three & four \\
\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 colored 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{\textgreater} column specifier, to add a colored 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:

\texttt{\textbackslash columncolor[\langle color model\rangle]\{\langle colour\}\} \{\langle left overhang\}\} \{\langle right overhang\}\} 

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 \texttt{right overhang} argument is omitted then it defaults to \texttt{left overhang}. If they are both omitted they default to \texttt{\textbackslash tabcolsep} (in \texttt{tabular}) or \texttt{\textbackslash arraycolsep} (in \texttt{array}).

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

\*This file has version number v0.1i, last revised 1999/03/24.
The default overhang of `\tabcolsep` produces:

<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
</tr>
</thead>
<tbody>
<tr>
<td>three</td>
<td>four</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
</tr>
</thead>
<tbody>
<tr>
<td>three</td>
<td>four</td>
</tr>
</tbody>
</table>

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}

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

Continued...
A long table example (continued)

<table>
<thead>
<tr>
<th>First two columns</th>
<th>Third column</th>
<th>D-type (dcolumn)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-type</td>
<td></td>
<td>12-4</td>
</tr>
<tr>
<td>Depending on your</td>
<td></td>
<td></td>
</tr>
<tr>
<td>driver you may get</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsightly gaps or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lines where the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'screens' used to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>produce different</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shapes interact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>badly. You may</td>
<td></td>
<td></td>
</tr>
<tr>
<td>want to cause</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adjacent panels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of the same colour</td>
<td></td>
<td></td>
</tr>
<tr>
<td>by specifying a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>larger overhang</td>
<td></td>
<td></td>
</tr>
<tr>
<td>or by adding some</td>
<td></td>
<td></td>
</tr>
<tr>
<td>negative space (in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a \noalign</td>
<td></td>
<td></td>
</tr>
<tr>
<td>before rows.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The End</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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}{%p{2cm}}\newcolumntype{B}{%p{3cm}}\newcolumntype{C}{%p{3cm}}\newcolumntype{E}{%p{3cm}}\newcolumntype{F}{%p{3cm}}\newcolumntype{G}{%p{3cm}}\newcolumntype{H}{%p{3cm}}\newcolumntype{I}{%p{3cm}}
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 \LaTeX{}'s \texttt{\char101leader} mechanism which is used by this package to insert the ‘stretchy’ coloured panels is rather like \texttt{glue}, the \texttt{\char101tabskip} 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}{% \\
@{\extracolsep{\fill}}>{\columncolor{gray}{.8}{20mm}}l
>{\columncolor{gray}{5mm}}l} \\
\end{tabular*}
\end{verbatim}

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

4 The \texttt{\char101rowcolor} command

As demonstrated above, one may change the colour of specified rows of a table by the use of \texttt{\char101multicolumn} 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{\char101rowcolor}, has been introduced\footnote{At some cost to the internal complexity of this package}.

\texttt{\char101rowcolor} takes the same argument forms as \texttt{\char101columncolor}. 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{\char101columncolor} command for that column, or \texttt{\char101tabcolsep (arraycolsep in array)}.

If a table entry is in the scope of a \texttt{\char101columncolor} specified in the table preamble, and also a \texttt{\char101rowcolor} at the start of the current row, the colour specified by \texttt{\char101rowcolor} will take effect. A \texttt{\char101multicolumn} command may contain >\texttt{\char101rowcolor}… which will override the default colours for both the current row and column.

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

5 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}\line} where you’d normally use \texttt{\line}. The space between \texttt{\line} will normally be left white. If you want to colour that as well, either increase the overhang of the previous column (to \texttt{\char101tabcolsep + \char101arrayrulewidth}}
Or remove the inter rule glue, and replace by a coloured rule of the required thickness. So

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

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

\section*{6 \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.

\section*{7 \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}
\doublerulesepcolor{yellow}

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

\section*{8 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 $\ast$ 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.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>

9 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 \hline argument, rather than \cline.

10 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 \texttt{\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 \texttt{tabs} packages provides a similar \texttt{\tablinesep}. I was going to give this the same name for compatibility with \texttt{tabs}, but that is implemented quite differently and probably has different behaviour. So I’ll keep a new name for now.

11 The Code

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

\begin{verbatim}
\edef\@tempa{\noexpand\AtEndOfPackage{\catcode\noexpand\^^A\the\catcode\^^A\relax}}
\@tempa
\catcode\^^A=\catcode\%
\DeclareOption{debugshow}{\catcode\^^A=9 }
\end{verbatim}

All the other options are handled by the \texttt{color} package.

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

\begin{verbatim}
\RequirePackage{array,color}
\@classz
\@classz is the main function in the \texttt{array} package handling of primitive column types: It inserts the code for each of the column specifiers, ‘\texttt{clrpmb}’. The other classes deal with the other preamble tokens such as ‘@’ or ‘>’.
\end{verbatim}

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 \texttt{\toks\@tempptokena} but \texttt{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 does). The \texttt{\newcolumntype} command of \texttt{array} only adds ‘second class’ column types. The re-implementations of \texttt{\newcolumntype} in my \texttt{blkarray} or Mark Wooding’s \texttt{mdwtab} allow new ‘first class’ column types to be declared, but stick with \texttt{array} for now. This means we have to lift the stuff out of the register before the register gets emptied in the wrong place.

\begin{verbatim}
\expandafter\CT@extract\the\toks\@tempptokena\columncolor!\@nil
\@addtopreamble{\setbox\z@\hbox{bgroup\bgroup}
\ifcase\@chnum
\@addtopreamble{\expandafter\CT@extract\the\toks\@tempptokena\columncolor!\@nil
\everypar{\ifcase\@chnum
\savebox\z@\hbox{bgroup\bgroup}
\ifcase\@chnum
\end{verbatim}

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.

```
\hskip\stretch{.5}\kern\z@ \tdollarbegin \insert@column \tdollarbegin \hskip\stretch{.5}\or
\hfill \kern\z@ \tdollarbegin \insert@column \tdollarend \hfill \or
\vcenter \@startpbox{\@nextchar}\insert@column \@endpbox \fi
```

Close the box register assignment.

```
\begingroup
\CT@setup
Run any code resulting from \texttt{\columncolor} commands.
```

```
\CT@column@color
Run code from \texttt{\rowcolor} (so this takes precedence over \texttt{\columncolor}).
```

```
\CT@do@color
This is \texttt{\relax} unless one of the two previous commands has requested a colour, in which case it will be \texttt{\CT@do@color} which will insert \texttt{\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 flexibilily. 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 \texttt{total 0fill}. If this box contributes fill glue, then the leaders will not reach the full width of the entry. In the case of \texttt{\multicolumn} entries it is actually possible for this box to contribute \texttt{shrink} glue, in which case the coloured panel for that entry will be too wide. Tough luck.

```
\unhbox\z@\%
\prepnext@tok}
```
\CT@setup  Initialise the overhang lengths and the colour command.

\def\CT@setup{%
  \@tempdimb\col@sep
  \@tempdimc\col@sep
  \def\CT@color{%
    \global\let\CT@do@color\CT@@do@color
    \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.

\def\CT@@do@color{%
  \global\let\CT@do@color\relax
  \@tempdima\wd\z@
  \advance\@tempdima\@tempdimb
  \advance\@tempdima\@tempdimc
  \kern-\@tempdimb
  \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.

```
```

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

\def\CT@extract#1\columncolor#2#3\@nil{%
  \if!#2%
    \let\CT@column@color\@empty
  \else
    \if\[#2%
      \CT@extractb{#1}#3\@nil
    \else
      \def\CT@column@color{\CT@color[#2]{#3}}
      \CT@extractd{#1}\@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)#3\@nil
  }\fi
  \fi}
\CT@extractd Now look for left-overhang (default to \col@sep).
74 \def\CT@extractd#1{\@testopt{\CT@extracte{#1}}\col@sep}

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

\CT@extractf Add the overhang info to \CT@do@color, for executing later.
76 \def\CT@extractf#1[#2][#3]#4\columncolor#5\@nil{% 
77 \@tempdimb#2\relax 
78 \@tempdimc#3\relax 
79 \edef\CT@column@color{% 
80 \CT@column@color 
81 \@tempdimb\the\@tempdimb\@tempdimc\the\@tempdimc\relax}% 
82 \toks\@tempcnta{#1#4}}% 

\CT@everycr Steal \everypar to initialise row colours
83 \let\CT@everycr\everycr 
84 \newtoks\everycr 
85 \CT@everycr{\noalign{\global\let\CT@row@color\relax}\the\everycr}

\CT@start
86 \def\CT@start{% 
87 \let\CT@arc@save\CT@arc@ 
88 \let\CT@drsc@save\CT@drsc@ 
89 \let\CT@row@color@save\CT@row@color}

\CT@end
90 \def\CT@end{% 
91 \global\let\CT@arc@\CT@arc@save 
92 \global\let\CT@drsc@\CT@drsc@save 
93 \global\let\CT@row@color\CT@row@color@save}

\shortstack 
94 \edef@ishortstack#1{% 
95 \CT@start\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\CT@end\egroup}

\@tabarray array and tabular (delayed for delarray)
96 \AtBeginDocument{% 
97 \expandafter\def\expandafter\@tabarray\expandafter{\% 
98 \expandafter\CT@start\@tabarray\expandafter\CT@end\expandafter}}

\endarray
99 \def\endarray{\crcr \egroup \egroup \gdef\@preamble{}\CT@end}

\multicolumn \multicolumn
100 \def\multicolumn#1#2#3{% 
101 \multispan{#1}\begingroup 
102 \def\@addamp{\if@firstamp \@firstampfalse \else 
103 \@preamerr 5\fi}% 
104 \@mkpream{#2}\@addtopreamble\@empty 
105 \endgroup 
106 \def\@sharp{#3}}% 
107 \let\CT@row@color\relax
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).

The row height fudge length.

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.

For similar reasons, need to make this non-expandable
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

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

\def\DC@endright{\hfil\egroup\hskip\stretch{.5}\box\z@\box\tw@}}\fi

Old \texttt{dcolumn} code.

\def\DC@endright{\hfil\egroup\hskip\stretch{-.5}\box\z@\box\tw@\hskip\stretch{-.5}}\else

\else

\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}}\ifx\@tempa\DC@endright

\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}}\ifx\@tempa\DC@endright

\def\DC@endright{\hfil\egroup%}
\else
\hskip\stretch{-.5}\box\z@\box\tw@\hskip\stretch{-.5}}\else

\def\DC@endright{\hfil\egroup%}
\else

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

\AtBeginDocument{%
  \ifx\hhline\@undefined\else
    \def\HH@box#1#2{\vbox{{%
      \advance\dimen@\doublerulesep
      \hrule \@height\dimen@ \vskip\dimen@
      \CT@arc@
    \hrule \@height \arrayrulewidth \@width #1
    \vskip\doublerulesep
      \hrule \@height \arrayrulewidth \@width #2}}}
  \def\HH@loop{%
    \ifx\@tempb`\def\next##1{\the\toks@\cr}\else\let\next\HH@let
      \ifx\@tempb|\if@tempswa
        \ifx\CT@drsc@\relax
          \HH@add{\hskip\doublerulesep}\
        \else
          \HH@add{{\CT@drsc@\vrule\@width\doublerulesep}}\fi
        \fi\@tempswatrue
        \HH@add{{\CT@arc@\vline}}\else
        \ifx\@tempb:\if@tempswa
          \ifx\CT@drsc@\relax
            \HH@add{\hskip\doublerulesep}\
          \else
            \HH@add{{\CT@drsc@\vrule\@width\doublerulesep}}\fi
          \fi\@tempswatrue
          \HH@add{\@tempc\HH@box\arrayrulewidth\@tempc}\else
          \ifx\@tempb~\@tempswafalse
            \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
            \ifx\CT@drsc@\leaders\hrule\@height\doublerulesep\hfil}\else
          \ifx\@tempb\@tempswafalse
            \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
            \HH@add{%\CT@arc@\leaders\hrule\@height\doublerulesep\hfil}\%}
          \else
            \ifx\@tempb=#\@tempswafalse
              \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
              \HH@add{%\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfil}\%
            \else
              \ifx\@tempb=\@tempswafalse
                \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
                \HH@add{%

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

\if\@tempswa\tl@hhadd{\HH@box\doublerulesep\z@}{\@tempswafalse}\else
\if\@tempswa\b\HH@add{\HH@box\z@\doublerulesep}{\@tempswafalse}\else
\if\@tempswa\def\next##1##2{%
\HH@add{%
{\baselineskip\p@\relax
##2%
\global\setbox\@ne\HH@box\doublerulesep\doublerulesep}}%
\HH@let!\else
\PackageWarning{hhline}%
{\meaning\@tempb\space ignored in \noexpand\hhline argument%
\MessageBreak}\i\fi\fi\fi\fi\fi\fi\fi\fi

\next}

\fi}

longtable support.
\AtBeginDocument{
\ifx\longtable\@undefined\else
\def\LT@hline{%
\ifx\LT@next\hline
\global\let\LT@next\@gobble
\ifx\CT@drsc@\relax
\gdef\CT@LT@sep{%
\noalign{\penalty-\@medpenalty\vskip\doublerulesep}}%
\else
\gdef\CT@LT@sep{%
\multispan\LT@cols{%
\CT@drsc@\leaders\hline\@height\doublerulesep\hfill}\cr}%
\noalign{\penalty\@M}%
\LT@next}
\fi
\else
\global\let\LT@next\empty
\gdef\CT@LT@sep{%
\multispan\LT@cols{%
\CT@arc@\leaders\hline@\@height\doublerulesep\hfill}\cr}%
\fi
\else
\global\let\LT@next\empty
\gdef\CT@LT@sep{%
\noalign{\penalty-\@medpenalty\vskip\doublerulesep}}%
\fi
\ifnum0=`{\fi}%
\multispan\LT@cols
{%\CT@arc@\leaders\hline\@height\arrayrulewidth\hfill}\cr
\CT@LT@sep
\multispan\LT@cols
{%\CT@arc@\leaders\hline\@height\arrayrulewidth\hfill}\cr
\noalign{\penalty\@M}%
\LT@next}
\fi}

\fi}

/*package*/