The 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 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 \LaTeX. This package requires \LaTeX (and its color and array packages).

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

\begin{tabular}{|l|c|}
\hline
one & two \\
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 \textit{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:
\texttt{\textbackslash columncolor[(color model)]\{(colour)\} [(left overhang)][(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}.

*This file has version number v0.1i, last revised 1999/03/24.
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]}l
>{\color{white}\columncolor[gray]{.2}[0pt]}l
```

The default overhang of `\tabcolsep` produces:

```
>{\columncolor[gray]{.8}}l
>{\color{white}\columncolor[gray]{.2}}l
```

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

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

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}`

```
\begin{longtable}{l|l|l}
\hline
\textbf{A long table example} & \textbf{First two columns} & \textbf{Third column} \\
& \textbf{p-type} & \textbf{D-type (dcolumn)} \\
\hline
\textbf{P-column} & \textbf{and another one} & 12\cdot34 \\
\hline
\textbf{Total} & \textbf{(wrong)} & 100\cdot6 \\
\hline
\textbf{Some long text in the first column} & \textbf{bbb} & 1\cdot2 \\
\textbf{aaa} & \textbf{and some long text in the second column} & 1\cdot345 \\
\hline
\textbf{Total} & \textbf{(wrong)} & 100\cdot6 \\
\hline
\textbf{aaa} & \textbf{Note that the coloured rules in all columns stretch to accomodate large entries in one column.} & \\
\hline
\end{longtable}
```

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}{{\columncolor{blue}[.5\tabcolsep]\color{yellow}\raggedright}p{3cm}}
\newcolumntype{C}{{\columncolor{yellow}[.5\tabcolsep]\bfseries D{.}{\cdot}{3.3}}}c
\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]l}}
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}{% \texttt{@{extracolsep{\fill}}}
 >\{\columncolor[gray]{.8}[0pt][20mm]\}\texttt{l}
 >\{\columncolor[gray]{.8}[5mm][0pt]\}\texttt{l}
 \texttt{@{}}
\end{tabular*}
\end{verbatim}

one two
three four

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

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

one two
three four

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\(^1\).

\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} (\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.

\(^1\)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 Colouring rules.

So you want coloured rules as well?

One could do vertical rules without any special commands, just use something like \texttt{!\{\textcolor{green}{\vline}\}} where you’d normally use \texttt{|}. The space between \texttt{||} 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{!\{\textcolor{green}{\vline}\}@\textcolor{yellow}{\vrule width \doublerulesep}}

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).

### 6 \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.

### 7 \texttt{\doublerulesepcolor}

Having coloured your rules, you’ll probably want something other than white to go in the gaps made by || or \texttt{\hline\hline}. \texttt{\doublerulesepcolor} works just the same way as \texttt{\arrayrulecolor}. The main thing to note that if this command is used, then \texttt{longtable} will not ‘discard’ the space between \texttt{\hline\hline} at a page break. (\TeX\ has a built-in ability to discard space, but the coloured ‘space’ which is used once \texttt{\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.)
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 \{...\} to add declarations that apply to the following - or = column rule. In particular you may give \arrayrulecolor and \doublerulesepcolor declarations in this argument.

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

\newcommand\rainbowline[1]{%\begin{tabular}{||*7{>{\columncolor[gray]{.9}}c}||}\rainbowline{t}%%
\end{tabular}}

\begin{tabular}{||l||c||}
\hline\hline
Richard & York & gave & battle & in & vain \\
\hline
1        & 2       & 3     & 4     & 5     & 6     & 7
\end{tabular}
Richard&of&York&gave&battle&in&vain
\cline{1-7}
\hline
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 \hhline 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 \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.

11 The Code

\begin{verbatim}
\edef\@tempa{%
\noexpand\AtEndOfPackage{%
\catcode'\noexpand\^^A\the\catcode'\^^A\relax}}
\@tempa
\catcode'\^^A=9
\DeclareOption{debugshow}{\catcode'\^^A=9}
\ProcessOptions
\end{verbatim}
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, ‘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 \texttt{\toks\@temptokena} but \texttt{array} will insert it too late (well it would work for \texttt{c}, but not for \texttt{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 \texttt{\texttt{\texttt{\newcolumntype}}} command of \texttt{array} only adds ‘second class’ column types. The re-implementations of \texttt{\texttt{\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.
\end{verbatim}

\begin{verbatim}
\expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil
Save the entry into a box (using a double group for colour safety as usual).
\end{verbatim}

\begin{verbatim}
\@addtopreamble{%
\setbox\z@\hbox\bgroup\bgroup
\ifcase \@chnum
\texttt{c} code: This used to use twice as much glue as \texttt{l} and \texttt{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@
\dollarbegin \insert@column \dollarend \hfill \or
\hfill \kern\z@ \dollarbegin \insert@column \dollarend \or
\texttt{m}, \texttt{p} and \texttt{b} as before.
\$vcenter
\vcenter\@startpbox{\@nextchar}\insert@column \@endpbox \$\or
\vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
\vbox \@startpbox{\@nextchar}\insert@column \@endpbox
\fi
Close the box register assignment.
\end{verbatim}
The main new stuff.
\begingroup

Initialise colour command and overhands.
\CT@setup

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

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

This is \relax unless one of the two 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

\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).
\@tempdima\ht\z@\advance\@tempdima\minrowclearance\vrule\@height\@tempdima\@width\z@

It would be safer to leave this boxed, but unboxing allows some flexibilty. 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.
\unhbox\z@}%

\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\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.
^^A\@height\p@\@depth\p@
\hskip\@tempdima\@plus 1fill
\kern-\@tempdimc
Now glue to exactly compensate for the leaders.
\hskip-\wd\z@ \@plus -1fill }
\CT@extract
Now the code to extract the \columncolor commands.
\def\CT@extract#1\columncolor#2#3\@nil{%
  \if!#2%
    ! is a fake token inserted at the end.
  \let\CT@column@color\@empty
  \else
  If there was an optional argument
  \if[#2]
  \CT@extractb{#1}#3\@nil
  \else
  No optional argument
  \def\CT@columncolor{%
  \CT@color{#2}}%
  \CT@extractd{#1}%
  \fi
  \fi
\CT@extractb
Define \CT@columncolor 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\@nil{%
  \def\CT@columncolor{%
  \if\CT@columncolor{#2}{#3}{#3}\@nil
  \fi
\CT@extractd
\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}}
\texttt{\CT@extractf} Add the overhang info to \texttt{\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)%
def\CT@everycr Steal \texttt{\everypar} to initialise row colours
\let\CT@everycr\everycr
\newtoks\everycr
\CT@everycr\{\noalign{\global\let\CT@row@color\relax}\the\everycr}
def\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}
def\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}
def\@ishortstack#1{% 
\CT@start\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\CT@end\egroup}
def\multicolumn#1#2#3{% 
\multispan{#1}\begingroup 
\def\@addamp\if@firstamp\@firstampfalse\else\@preamerr 5\fi% 
\@mkpream{#2}\@addtopreamble\@empty 
\endgroup 
\let\CT@row@color\relax 
\let\CT@column@color\relax 
def\@tabarray array and tabular (delayed for delarray)
def\AtBeginDocument{% 
\expandafter\def\expandafter\@tabarray\expandafter{% 
\expandafter\CT@start\@tabarray}}
def\endarray 
def\multicol Multicolumn 
def\multicolumn#1#2#3{% 
\multispan(#1)\begingroup 
\def\@addamp\if@firststamp\@firststampfalse\else\@preamerr 5\fi\% 
\@mkpream(#2)\@addtopreamble\@empty 
\endgroup 
\def\@sharp#3\% 
\let\CT@row@color\relax 
\let\CT@column@color\relax
Coloured rules and rule separations.

\def\@classvi{\ifcase\@lastchclass
\@acol \or
\ife\CT@drsc\relax
\@addtopreamble{\hskip\doublerulesep}\%
\else
\@addtopreamble{{\CT@drsc\vrule\@width\doublerulesep}}\%
\fi\or
\@acol \or
\@classvii
\fi}

doublerulesepcolor
\def\doublerulesepcolor#1#{\CT@drs{#1}}
\def\CT@drs#1#2{%\ifdim\baselineskip=\z@\noalign\fi
\gdef\CT@drsc@{\color#1{#2}}}
\let\CT@drsc@\relax
arrayrulecolor
\def\arrayrulecolor#1#{\CT@arc{#1}}
\def\CT@arc#1#2{%\ifdim\baselineskip=\z@\noalign\fi
\gdef\CT@arc@{\color#1{#2}}}
\let\CT@arc@\relax

\hline

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

The row height fudge length.
newlength\minrowclearance
\minrowclearance=0pt
\let\CT@do@color\relax

For similar reasons, need to make this non-expandable
\let\CT@do@color\relax
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\box2\box0}\box2%}
\ifx\@tempa\DC@endright
\hfill
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.
\def\DC@endright{%}
$\hfill\egroup$
\ifx\DC@rl\bgroup
\hskip\stretch{-.5}\box2\box0\hskip\stretch{-.5}%

Old dcolumn code.
\def\DC@endright{% $\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
    \def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\fi}

\AtBeginDocument{%
\ifx\hhline\@undefined\else
\def\HH@box#1#2{\vbox{{%\CT@drsc@\dimen@\tw@\arrayrulewidth\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\@tempswatrue\fi\HH@add{{\CT@arc@\vline}}\else\ifx\@tempb:\if@tempswa\HH@add{&\omit}\fi\if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi\ifx\CT@drsc@\relax\HH@add{\hfil}\else\HH@add{\hfil}\fi\@tempswatrue\fi\HH@add{{\CT@arc@\vline\copy\@one\@tempc\vline}}\else\ifx\@tempb~\@tempswafalse\if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi\ifx\CT@drsc@\relax\HH@add{\hfil}\else\HH@add{\hfil}\fi\@tempswatrue\fi\HH@add{{\CT@arc@\vline\copy\@one\@tempc\vline}}\else\ifx\@tempc\@empty\def\next##1{\the\toks@\cr}\else\let\next\HH@let\fi\HH@loop\HH@loop}}%
\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\@tempswatrue\fi\HH@add{{\CT@arc@\vline}}\else\ifx\@tempb:\if@tempswa\HH@add{&\omit}\fi\if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi\ifx\CT@drsc@\relax\HH@add{\hfil}\else\HH@add{\hfil}\fi\@tempswatrue\fi\HH@add{{\CT@arc@\vline\copy\@one\@tempc\vline}}\else\ifx\@tempb~\@tempswafalse\if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi\ifx\CT@drsc@\relax\HH@add{\hfil}\else\HH@add{\hfil}\fi\@tempswatrue\fi\HH@add{{\CT@arc@\vline\copy\@one\@tempc\vline}}\else\ifx\@tempc\@empty\def\next##1{\the\toks@\cr}\else\let\next\HH@let\fi\HH@loop\HH@loop}}%
Stop the backspacing for \textit{t} and \textit{b}, it messes up the underlying colour.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
\textbf{Column 1} & \textbf{Column 2} & \textbf{Column 3} \\
\hline
A & B & C \\
\hline
D & E & F \\
\hline
\end{tabular}
\end{table}

\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\hrule\@height\doublerulesep\hfill}\cr}%
\fi}
\else
\global\let\LT@next\empty
\gdef\CT@LT@sep{\noalign{\penalty-\@lowpenalty\vskip-\arrayrulewidth}}%
\fi}
\fi}

\end{document}
\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