The **xtab** package[*]

Author: Peter Wilson, Herries Press
Maintainer: Will Robertson
will dot robertson at latex-project dot org

2009/09/04

Abstract

The xtab package enables long tables to be automatically broken at page boundaries. It is an extension of the **supertabular** package and also reduces or eliminates some of its weaknesses.

Contents

List of Tables

1 Introduction

Although the xtab package was originally developed as part of a suite for typesetting ISO international standards [Wil96], it is also applicable for use with the LaTeX standard classes. The package is an extension of the **supertabular** package developed by Johannes Braams and Theo Jurriens.[1] It reduces some of the weaknesses noted in the **supertabular** documentation and provides additional functionality.

Section 2 provides the user manual for the package which enables long tables to be automatically broken across multiple pages. Section 3 describes the implementation.

---

[*]This file (`xtab.dtx`) has version number v2.3d, last revised 2009/09/04.

[1]supertabular.sty, version 4.1c, 7 November 1997.

This manual is typeset according to the conventions of the LaTeX DOC-STRIP utility which enables the automatic extraction of the LaTeX macro source files [GMS94].

2 The **xtab** package

The supertabular package provides for the automatic breaking of a long table across page boundaries. The extension provided here enables the heading on the table on the last page to differ from those on earlier pages of the table. The downside of the extension is that LaTeX has to be run twice if the document contains a `supertabular`. However, LaTeX is usually run at least twice for any but the simplest document in order to get cross-references and Table of Contents, etc., resolved correctly.

The current version of the extension also either cures or reduces following weaknesses in the supertabular package.[2]

1. Sometimes the top caption of a `supertabular` is printed on one page and the body is printed on the following page(s). That is, there is a lonely caption.

2. Sometimes the last page of a `supertabular` consists of an empty table. That is, just the head and foot of the table are printed.

3. If the number of lines in the first header for the table differs from the number of lines in subsequent headers, then the continuation pages of the table may be too short or, more troubling, too long.

The weaknesses are caused by trying to guess where TeX will put a page break. The package has to guesstimate how long the next entry will be in the table and, if it is too long for the available space, it puts in its own page break. If its guess is off too much in one direction, TeX will break the page unexpectadly; if it's off in the other direction supertabular will put in an unnecessary page break.

The xtab package has reduced, but perhaps not entirely eliminated, these weaknesses. Some hand tuning may still be required.

The principal commands available are given in Table 1.

Table 1: The principal xtab package commands

Command	Effect
`\begin{xtabular}{...}`	This is equivalent to the normal `\begin{tabular}{...}` environment. You supply the specification of the columns just as for the normal tabular environment.
	Continued on next page

---

[2]I have corresponded with the authors of supertabular about these.

Table 1 – continued from previous page

Command	Effect
	All commands that can be used within a `tabular` environment can also be used within the `xtabular` environment.
	Unlike the `tabular` environment which prevents page breaking within the tabular, the `xtabular` allows page breaking, so that tabulars can extend automatically across several pages.
	`xtabular` starts off with a `tabular` environment and checks the amount of space left on the page as it adds each row to the tabulation. If the space left on the page is too short for another row, then it ends the current `tabular`, performs a page break and starts another `tabular` on the following page. This process is repeated until all the rows have been output.
	There are special commands for captioning an `xtabular` as a table, and also elements can be automatically inserted after each (internal) `\begin{tabular}` and immediately before each `\end{tabular}`.
	Do not put a `xtabular` in a `table` environment, as the `table` environment keeps its contents on a single page (presumably you are using `xtabular` because its contents are longer than one page).
`\end{xtabular}`	End the `xtabular` environment.
`\begin{mpxtabular}`	Like the `xtabular` environment except that each 'page' is put into a `minipage` first.
	Thus it is possible to have footnotes inside an `mpxtabular`. The footnote text is printed at the end of each page.
Continued on next page	

Table 1 – continued from previous page

Command	Effect
`\end{mpxtabular}`	End the `mpxtabular` environment.
	Note: If any of the following commands are used, then they should be placed before the particular `xtabular` environment that they apply to.
`\topcaption{...}`	A command to provide a caption for the table. The caption is placed at the top of the table.
`\bottomcaption{...}`	A command to provide a caption for the table. The caption is placed at the bottom of the table.
`\tablecaption{...}`	A command to provide a caption for the table. The caption is placed at the default position, which is at the top of the table.
	Notes: You cannot use the `\caption` command but you can put a label after any of these captioning commands. If you want captioning, the command must be specified before the start of the supertabular environment.
	The `\...caption{}` command(s) remain in effect until changed by another `\...caption` command.
`\tablefirsthead{...}`	Defines the contents of the first occurence of the tabular head. The tabular head is some special treatment of the first row in the table. This command is optional.
	If used, the header must be closed by the end of line command for tabulars (e.g., `\\`).
`\tablehead{...}`	Defines the contents of the table head on subsequent pages.
	For example, you might want to note that this is a continuation of the table on the previous page, as well as repeating any column headings that were given at the start of the `xtabular` by `\tablefirsthead`.
	Continued on next page

<div align="center">

Table 1 – concluded from previous page

</div>

Command	Effect
`\tablelasthead{...}`	The header must be closed like the `\tablefirsthead` command.
	Defines the contents of the table head on the last page of the table.
	For example, you might want to note that the table is concluded on this page.
	The header must be closed like the `\tablefirsthead` command.
`\notablelasthead`	Switches off the last `\tablelasthead`. A `\tablelasthead` stays in effect until overwritten by a new `\tablelasthead` or cancelled by this command.
`\tabletail{...}`	The contents of this command are inserted before the (internal) `\end{tabular}` on each page except for the last page of the table.
	For example, you might want to note that the table is continued on the next page.
`\tablelasttail{...}`	The contents of this command are inserted before the final (internal) `\end{tabular}` of the table.
	For example, you might want to note that this is where the table ends.

As well as the `xtabular` and `mpxtabular` environments there are the corresponding starred versions (i.e., `xtabular*` and `mpxtabular*` for use in two column mode where the table is meant to span both columns.

Table 1 was produced by code similar to the following:

```
\topcaption{The principal xtab package commands} \label{tab:xtab}
\tablefirsthead{\hline \multicolumn{1}{|c|}{\textbf{Command}} &
                \multicolumn{1}{c|}{\textbf{Effect}} \\ \hline }
\tablehead{\multicolumn{2}{c}%
          {{\captionsize\bfseries \tablename\ \thetable{} --
             continued from previous page}} \\
  \hline    \multicolumn{1}{|c|}{\textbf{Command}} &
          \multicolumn{1}{c|}{\textbf{Effect}} \\ \hline }
\tablelasthead{\multicolumn{2}{c}%
          {{\captionsize\bfseries \tablename\ \thetable{} --
             concluded from previous page}} \\
  \hline    \multicolumn{1}{|c|}{\textbf{Command}} &
          \multicolumn{1}{c|}{\textbf{Effect}} \\ \hline }
\tabletail{\hline \multicolumn{2}{|r|}{{Continued on next page}} \\ \hline}
\tablelasttail{\hline \hline}

\begin{center}
```

```
\begin{xtabular}{|l|p{0.5\textwidth}|}
\verb|\begin{xtabular}{...}| & This is equivalent to the normal
                             \verb|\begin{tabular}{...}| environment.
                             You supply the specification of the columns
                             just as for the normal \Lenv{tabular} environment.
 \\
 &

                             All commands that can be used within a \Lenv{tabular}
                             environment can also be used within
                             the \Lenv{xtabular} environment.
 \\
 &
    Unlike the \Lenv{tabular} environment which prevents page breaking
within the tabular, the \Lenv{xtabular} allows page breaking, so that
tabulars can extend automatically across several pages.
... ... ...
\verb|\tablelasttail{...}| & The contents of this command are inserted before
                             the final (internal) \verb|\end{tabular}| of the table.
 \\
 &
    For example, you might want to note that  this is where
the table ends.
 \\
\end{xtabular}
\end{center}
```

  The table is only broken between rows — a row will not be split across pages. This can lead to some bad page breaks, especially if there are rows with a large vertical height (like some in Table 1). It is best to keep rows not too tall.

  Unkike the `table` environment which floats, an `xtabular` environment is typeset at the point in the document where the environment is specified. It is best not to start an `xtabular` too close to the bottom of a page otherwise there might be an ugly page break.

\shrinkheight The command `\shrinkheight{`⟨*length*⟩`}` may be used after the first `\\` in the table to modify the allowed height of the table on that page. A positive ⟨*length*⟩ decreases the allowed space on the page and a negative ⟨*length*⟩ increases the allowed space.

  For example:

`\shrinkheight{2\baselineskip}` decreases the space per page by two lines.

`\shrinkheight{-\baselineskip}` increases the space per page by one line.

  Note that I have never tried using this command so I cannot comment on its efficacy. Instead, I use the `\xentrystretch` command when necessary.

\xentrystretch The command `\xentrystretch{`⟨*decimal-fraction*⟩`}` can be used before a table to modify the amount of vertical space apparently consumed by each entry in the subsequent table(s). The default is `\xentrystretch{0.1}` which specifies a 10% overestimate in the vertical space. Similarly, `\xentrystretch{0.25}` will overestimate the space by 25%. A different value may be used for each table in

order to eliminate, or at least reduce, bad page breaks. Increasing the value causes fewer entries to be put on a page, thus reducing the chance of TEX putting in a page break before the xtab package is prepared for one.

You may specify the font used for the \tablehead and \tablelasthead yourself.

Note: Within ISO documents, captions shall be in bold font. The iso class also provides a command for setting the size of the font used in captions, namely \captionsize. The default value for this is set by the iso class. For the curious, the default definition is:

```
\newcommand{\captionsize}{\normalsize}
```

2.1 Options

The xtab package has three options which control the amount of information that is written to the .log file. The options are:

1. The option errorshow (the default) does not write any extra information;

2. The option pageshow writes information about when and why xtab decides to produce a new page;

3. The option debugshow, which also includes pageshow, additionally writes information about each line that is added to the table.

Under normal circumstances xtab is used without invoking any option. The pageshow option may be useful when attempting to cure a bad page break. The debugshow option, as its name implies, is principally of use to the xtab developer.

Independently of the options, the command \sstraceon may be used at any point in the document to turn on printing of debugshow data. This can be turned off later by the \sstraceoff command, which will stop all . . . show printing.

3 The implementation

The xtab package provides an extension to the supertabular package written by Johannes Braams and Theo Jurriens.[3] The major portion of the following documentation is taken from supertabular.dtx. The package is designed to be used with the iso class in addition to the usual article, etc., classes.

The extension provided here enables the heading on the table on the last page to differ from those on earlier pages of the table. The implementation of the extension is based on ideas in David Carlisle's longtable package. The downside of the extension is that LaTeX has to be run twice if the document contains a supertabular. However, LaTeX is usually run at least twice for any but the simplest document in order to get cross-references and Table of Contents, etc., resolved correctly.

---

[3]supertabular.sty, version 4.1c, 7 November 1997.

The current version of the extension also either cures or reduces following weaknesses in the supertabular package.[4]

1. Sometimes the top caption of a supertabular is printed on one page and the body is printed on the following page(s). That is, there is a lonely caption.

2. Sometimes the last page of a supertabular consists of an empty table. That is, just the head and foot of the table are printed.

3. If the number of lines in the first header for the table differs from the number of lines in subsequent headers, then the continuation pages of the table may be too short or, more troubling, too long.

The first version of xtab imported much of the code from the supertabular package (version 3.7) but I found that this did not work well because there were incompatible coded versions of supertabular available on CTAN. Further, I found that there were some problems with the original supertabular code in any case.[5] I have to make the assumption that other users may have dissimilar or problematic versions, so include all the code here, and thus any errors can now be laid at my door.

The requirement for compatibilty with the iso class is achieved by modifications to the \ST@caption command only. Effectively this is orthogonal to the code required to implement the extension.

Now for the code itself. As syntactic sugar, all new macros for the extension have the prefix 'PWST' to distinguish them from the original macros. I have also denoted all extensions to the original supertabular by introducing them as *Extension:*.

Announce the name and version of the package, which requires LaTeX 2_ε.

```
1 ⟨*xtab⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{xtab}[2009/09/04 v2.3d Extended supertabular package]
4
```

\c@tracingst There are three options with the package which control the amount of information written to the log file:

1. errorshow (the default) no extra information

2. pageshow writes information about page breaking

3. debugshow adds information about each line that is added to the tabular

```
5 \newcount\c@tracingst
6 \DeclareOption{errorshow}{\c@tracingst\z@}
```

Extension: The next line in the original code did not do what the authors intended; the number should have been 3 rather than 2.

---

[4]Only the first two of these have been recognised by the authors of supertabular.
[5]I also found a bug in the 4.1b version which the authors kindly fixed in version 4.1c.

```
 7 %%%%\DeclareOption{pageshow}{\c@tracingst\tw@}
 8 \DeclareOption{pageshow}{\c@tracingst\thr@@}
 9 \DeclareOption{debugshow}{\c@tracingst5\relax}
10 \ProcessOptions
11
```

\if@topcaption The user-commands \topcaption and \bottomcaption set the flag @topcaption
\topcaption   to determine where to put the tablecaption. The default is to put the caption on
\bottomcaption the top of the table

```
12 \newif\if@topcaption \@topcaptiontrue
13 \def\topcaption{\@topcaptiontrue\tablecaption}
14 \def\bottomcaption{\@topcaptionfalse\tablecaption}
15
```

\PWST@thecaption *Extension:* \PWST@thecaption is used to store the text of the table's caption. The
\PWSTcapht   vertical space required by a caption is stored in \PWSTcapht.

```
16 \gdef\PWST@thecaption{}
17 \newdimen\PWSTcapht
```

\tablecaption   This command has to function exactly like \caption does except it has to store its
\@xtablecaption argument (and the optional argument) for later processing *within* the supertabular
                 environment.

```
18 \long\def\tablecaption{%
19   \refstepcounter{table}\@dblarg{\@xtablecaption}}
20 \long\def\@xtablecaption[#1]#2{%
```

Extension: I store the caption text for later measurement.

```
21   \long\gdef\PWST@thecaption{#2}%
```

Finish up with the original code.

```
22   \long\gdef\@process@tablecaption{\ST@caption{table}[#1]{#2}}}
23 \global\let\@process@tablecaption\relax
24
```

\ifST@star This switch is used in the internal macros to remember which kind of environment
            was started.

```
25 \newif\ifST@star
```

\ifST@mp This flag is used in the internal macros to remember if the tabular is to be put in
          a minipage.

```
26 \newif\ifST@mp
```

\ST@wd For the **supertabular*** environment it is necessary to store the intended width of
        the tabular.

```
27 \newdimen\ST@wd
```

\ST@rightskip   For the **mpsupertabular** environments we need special versions of \leftskip,
\ST@leftskip   \rightskip and \parfillskip.
\ST@parfillskip

```
28 \newskip\ST@rightskip
29 \newskip\ST@leftskip
30 \newskip\ST@parfillskip
31
```

\@initisotab   Required for ISO class, and check if class loaded.

```
32 \@ifundefined{@initisotab}{%
33   \newcommand{\@initisotab}{}%
34   \newif\ifinfloat}{\typeout{xtab using iso captions}}
35
```

\ST@caption   This is a redefinition of LaTeX's \@caption, \@makecaption is called within a group so as not to return to \normalsize globally. In the original a fix was made for the 'feature' of the \@makecaption of article.sty and friends that a caption *always* gets a \vskip 10pt at the top and *none* at the bottom; if a user wants to precede his table with a caption this results in a collision. This fix is not implemented here as I think it should be done by the user modifying \beforecaptionskip and \aftercaptionskip.

Extension: The ISO captioning is also initialised.

```
36 \long\def\ST@caption#1[#2]#3{\par%
37   \@initisotab
38   \addcontentsline{\csname ext@#1\endcsname}{#1}%
39                   {\protect\numberline{%
40                      \csname the#1\endcsname}{\ignorespaces #2}}%
41   \begingroup
42     \@parboxrestore
43     \normalsize
44 %%  \if@topcaption \vskip -10\p@ \fi
45     \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
46 %%  \if@topcaption \vskip 10\p@ \fi
47   \endgroup
```

Extension: The height of the caption is subtracted from the available space on the page.

```
48   \global\advance\ST@pageleft -\PWSTcapht
49   \ST@trace\tw@{Added caption. Space left for xtabular: \the\ST@pageleft}}
50
```

\tablehead   \tablehead activates the new tabular \cr commands.
\tablefirsthead
```
51 \newcommand\tablehead[1]{%
52   \gdef\@tablehead{%
53   \noalign{%
54     \global\let\@savcr=\\
55     \global\let\\=\org@tabularcr}%
56   #1%
57   \noalign{\global\let\\=\@savcr}}}
58 \tablehead{}
59 \newcommand\tablefirsthead[1]{\gdef\@table@first@head{#1}}
60
```

\c@PWSTtable
\PWSTlastpage
\PWSTcurpage
\PWSTpenultimate
\PWSTtempc
\PWSTlines
\PWSThead
\PWSTlasthead

Extension: These are counters for the supertabular extension. *c@PWSTtable* counts the number of supertabulars in case one or more are not captioned. *PWSTlastpage* is a counter holding the number of pages that a supertabular uses and *PWSTpenultimate* is the penultimate page. *PWSTcurpage* counts the current number of supertabular pages processed. *PWSTtempc* is a scratch counter for page processing.

```
61 \newcounter{PWSTtable}
62 \newcount\PWSTlastpage
63 \newcount\PWSTpenultimate
64 \newcount\PWSTcurpage
65 \newcount\PWSTtempc
```

Extension: *PWSTlines* is used to count the number of supertabular entry lines on a page. Estimates of the number of lines in the normal table heading is held by *PWSThead*, and similarly *PWSTlasthead* is for the number of lines in the last heading.

```
66 \newcount\PWSTlines
67 %%% \newcount\PWSThead
68 %%% \newcount\PWSTlasthead
```

\iffirstcall

Extension: This is used by the extension code to flag if the presumed last page overflows. If overflow occurs, then `firstcall` is set to false.

```
69 \newif\iffirstcall
```

\PWST@lastht
\PWST@generalht
\PWST@ht

Extension: The estimated height of a table header and tail (i.e., the height of an empty table) for the last page of a supertabular is stored in \PWSTlastht. Similarly, the corresponding height of an empty table on a general page (neither the first nor the last) is stored in \PWSTgeneralht. \PWST@ht is a scratch variable.

```
70 \newdimen\PWST@lastht
71 \newdimen\PWST@generalht
72 \newdimen\PWST@ht
73
```

\tablelasthead
\@table@last@head
\notablelasthead

Extension: \tablelasthead is the extension user command to specify the heading for the last page of a supertabular. The command \notablelasthead switches off the last heading. This has to be used if a last headed table precedes one that does not have a special last head.

```
74 \newcommand{\tablelasthead}[1]{\gdef\@table@last@head{#1}}
75 \newcommand{\notablelasthead}{\let\@table@last@head\relax}
```

Now initialize these commands.

```
76 \tablelasthead{}
77 \notablelasthead
```

\tabletail
\tablelasttail

\tabletail is the user command to specify the appearance of the bottom of each tabular on a page. Special treatment is given to the end of the supertabular via the \tablelasttail command.

If the user uses an extra amount of tabular-data (like \multicolumn) in \tabletail TeX starts looping because of the definition of \nextline. So make

\\ act like just a \cr inside this tail to prevent the loop. Save and restore the value of \\

```
78 \newcommand\tabletail[1]{%
79   \gdef\@tabletail{%
80     \noalign{%
81       \global\let\@savcr=\\
82       \global\let\\=\org@tabularcr}%
83     #1%
84     \noalign{\global\let\\=\@savcr}}}
85 \tabletail{}
86 \newcommand\tablelasttail[1]{\gdef\@table@last@tail{#1}}
87 \tablelasttail{}
88
```

\sttraceon The original supertabular included a tracing mechanism to follow the decisions
\sttraceoff supertabular made about page breaking. This is now also used as a debugging mechanism for the extension.

```
89 \newcommand\sttraceon{\c@tracingst5\relax}
90 \newcommand\sttraceoff{\c@tracingst\z@}
```

\ST@trace A macro that gets the trace message as its argument

```
91 \newcommand\ST@trace[2]{%
92   \ifnum\c@tracingst>#1\relax
93     \GenericWarning{(xtab)\@spaces\@spaces}{Package xtab: #2}%
94   \fi}
95
```

\ST@pageleft This register holds the estimate of the amount of space left over on the current page. This is used in the decision when to start a new page.

```
96 \newdimen\ST@pageleft
```

\shrinkheight \shrinkheight is a command to diminish the value of \ST@pageleft if necessary.
\setSTheight \setSTheight sets the value of \ST@pageleft if necessary.

```
97 \newcommand*\shrinkheight[1]{%
98   \noalign{\global\advance\ST@pageleft-#1\relax}}
99 \newcommand*\setSTheight[1]{%
100    \noalign{\global\ST@pageleft=#1\relax}}
```

\xentrystretch *Extension:* Provide a user and internal command for fudging the estimated space
\PWST@xentrystretch taken by a table entry. Initialise to 10% increase.

```
101 \newcommand{\xentrystretch}[1]{\def\PWST@xentrystretch{#1}}
102 \xentrystretch{0.1}
103
```

\ST@headht The register \ST@headht holds the height of the first head of a supertabular.
\ST@tailht The register \ST@tailht holds the height of the tail.

```
104 \newdimen\ST@headht
105 \newdimen\ST@tailht
```

\ST@pagesofar	Register \ST@pagesofar stores the estimate of the amount of the page already filled up.

```
106 \newdimen\ST@pagesofar
```

\ST@pboxht	The measured (total) height of a parbox argument.

```
107 \newdimen\ST@pboxht
```

\ST@lineht \ST@stretchht \ST@prevht	The estimated height of a normal line is stored in \ST@lineht. The register \ST@stretchht is used to store the difference between the normal line height and the line height when \arraystretch has a non-standard value. This is used in the case when p-box entries are added to the tabular. \ST@prevht stores the height of the previous line to use it as an estimate for the height of the next line. This is needed for a better estimate of when to break the tabular.

```
108 \newdimen\ST@lineht
109 \newdimen\ST@stretchht
110 \newdimen\ST@prevht
```

\ST@toadd	When a tabular row is ended with \\[...] we need to temporarily store the optional argument in \ST@toadd.

```
111 \newdimen\ST@toadd
```

\ST@dimen	A private scratch dimension register.

```
112 \newdimen\ST@dimen
```

\ST@pbox	A box register to store the contents of a parbox.

```
113 \newbox\ST@pbox
114
```

\ST@tabularcr \ST@xtabularcr \ST@argtabularcr	These are redefinitions of \@tabularcr and \@xtabularcr. This is needed to include \ST@cr in the definition of \@xtabularcr.
	All redefined macros have names that are similar to the original names, except with a leading 'ST'.

```
115 \def\ST@tabularcr{%
116   {\ifnum0='}\fi
117   \@ifstar{\ST@xtabularcr}{\ST@xtabularcr}}
118 \def\ST@xtabularcr{%
119   \@ifnextchar[%
120     {\ST@argtabularcr}%
121     {\ifnum0='{\fi}\cr\ST@cr}}
122 \def\ST@argtabularcr[#1]{%
123   \ifnum0='{\fi}%
124   \ifdim #1>\z@
125     \unskip\ST@xargarraycr{#1}%
126   \else
127     \ST@yargarraycr{#1}%
128   \fi}
```

\ST@xargarraycr
\ST@yargarraycr In this case we need to copy the value of the optional argument of \\ in our private register \ST@toadd.

```
129 \def\ST@xargarraycr#1{%
130   \@tempdima #1\advance\@tempdima \dp \@arstrutbox
131   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr
132   \noalign{\global\ST@toadd=#1}\ST@cr}
```

Here we need to insert \ST@cr

```
133 \def\ST@yargarraycr#1{%
134   \cr\noalign{\vskip #1\global\ST@toadd=#1}\ST@cr}
135
```

\ST@startpbox The macros that deal with parbox columns need to be redefined, because we need to know the size of the parbox.

```
136 \def\ST@startpbox#1{%
```

To achieve our goal we need to save the text in box.

```
137 %%%%  \setbox\ST@pbox\vtop\bgroup\hsize#1\@arrayparboxrestore}
138   \setbox\ST@pbox\vtop\bgroup\setlength\hsize{#1}\@arrayparboxrestore}
```

\ST@astartpbox supertabular version of \@astartpbox.

```
139 \def\ST@astartpbox#1{%
140   \bgroup\hsize#1%
141 %%%%  \setbox\ST@pbox\vtop\bgroup\hsize#1\@arrayparboxrestore}
142   \setbox\ST@pbox\vtop\bgroup\setlength\hsize{#1}\@arrayparboxrestore}
```

\ST@endpbox
\ST@aendpbox supertabular versions of \@endpbox and \@aendpbox.

```
143 \def\ST@endpbox{%
144   \@finalstrut\@arstrutbox\par\egroup
145   \ST@dimen=\ht\ST@pbox
146   \advance\ST@dimen by \dp\ST@pbox
147   \ifnum\ST@pboxht<\ST@dimen
148     \global\ST@pboxht=\ST@dimen
149   \fi
150   \ST@dimen=\z@
151   \box\ST@pbox\hfil}
152 \def\ST@aendpbox{%
153   \@finalstrut\@arstrutbox\par\egroup
154   \ST@dimen=\ht\ST@pbox
155   \advance\ST@dimen by \dp\ST@pbox
156   \ifnum\ST@pboxht<\ST@dimen
157     \global\ST@pboxht=\ST@dimen
158   \fi
159   \ST@dimen=\z@
160   \unvbox\ST@pbox\egroup\hfil}
161
```

\estimate@lineht Estimates the height of normal line taking \arraystretch into account. Also computes the difference between a 'normal' line and a stretched one.

```
162 \def\estimate@lineht{%
163   \ST@lineht=\arraystretch \baselineskip
164   \global\advance\ST@lineht by 1\p@
165   \ST@stretchht\ST@lineht\advance\ST@stretchht-\baselineskip
166   \ifdim\ST@stretchht<\z@\ST@stretchht\z@\fi
167   \ST@trace\tw@{Basic line height: \the\ST@lineht\MessageBreak%
168            Arrayed line height: \the\ST@stretchht}%
169   \global\advance\ST@lineht \PWST@xentrystretch\ST@lineht
170   \ST@trace\tw@{Stretched line height: \the\ST@lineht}}
171
```

\@calfirstpageht Estimates the space left on the current page and decides whether the tabular can be started on this page or on a new page. Aspects of the original code are modified for the extension.

```
172 \def\@calfirstpageht{%
173   \ST@trace\tw@{Calculating height of xtabular on first page}%
```

The TeX register \pagetotal contains the height of the page sofar, the LaTeX register \@colroom contains the height of the column.

```
174   \global\ST@pagesofar\pagetotal
175   \global\ST@pageleft\@colroom
176   \ST@trace\tw@{Height of previous text = \the\pagetotal; \MessageBreak
177            Height of column = \the\ST@pageleft}%
```

When we are in twocolumn mode TeX may still be collecting material for the first column although there seems to be no space left. In this case we have to check against two times \ST@pageleft.

```
178   \if@twocolumn
179     \ST@trace\tw@{two column mode}%
180     \if@firstcolumn
181       \ST@trace\tw@{First column}%
182       \ifnum\ST@pagesofar > \ST@pageleft
183         \global\ST@pageleft=2\ST@pageleft
184         \ifnum\ST@pagesofar > \ST@pageleft
185           \newpage\@calnextpageht
186           \ST@trace\tw@{starting new page}%
187         \else
```

In this case we're in the second column, so we have to compensate for the material in the first column.

```
188           \ST@trace\tw@{Second column}%
189           \global\advance\ST@pageleft -\ST@pagesofar
190           \global\advance\ST@pageleft -\@colroom
191         \fi
```

When \ST@pagesofar is smaller than \ST@pageleft TeX is still collecting material for the first column, so we can start a new tabular environment like we do on a single column page.

```
192     \else
193       \global\advance\ST@pageleft by -\ST@pagesofar
```

```
194        \global\ST@pagesofar\z@
195      \fi
196    \else
```

When we end up here, TEX has already decided it had enough material for the first column and is building the second column.

```
197        \ST@trace\tw@{Second column}%
198        \ifnum\ST@pagesofar > \ST@pageleft
199          \ST@trace\tw@{starting new page}%
200          \newpage\@calnextpageht
201        \else
202          \global\advance\ST@pageleft by -\ST@pagesofar
203          \global\ST@pagesofar\z@
204        \fi
205      \fi
206    \else
```

In one column mode there is a simple decision.

```
207        \ST@trace\tw@{one column mode}%
208        \ifnum\ST@pagesofar > \ST@pageleft
209          \ST@trace\tw@{starting new page}%
210        \newpage\@calnextpageht
```

When we are not starting a new page subtract the size of the material already on it from the available space.

```
211      \else
212        \global\advance\ST@pageleft by -\ST@pagesofar
213        \global\ST@pagesofar\z@
214      \fi
215    \fi
216    \ST@trace\tw@{Available height: \the\ST@pageleft}%
```

Now we need to know the height of the head of the table. In order to measure this we typeset it in a normal `tabular` environment.

```
217    \ifx\@@tablehead\@empty
218      \ST@headht=\z@
219    \else
220      \setbox\@tempboxa=\vbox{\@arrayparboxrestore
221        \ST@restore
222        \expandafter\tabular\expandafter{\ST@tableformat}%
223        \@@tablehead\endtabular}%
224      \ST@headht=\ht\@tempboxa\advance\ST@headht\dp\@tempboxa
225    \fi
226    \ST@trace\tw@{Height of head: \the\ST@headht}%
```

To decide when to start a new page, we need to know the vertical size of the tail of the table.

```
227    \ifx\@tabletail\@empty
228      \ST@tailht=\z@
229    \else
230      \setbox\@tempboxa=\vbox{\@arrayparboxrestore
```

```
231     \ST@restore
232     \expandafter\tabular\expandafter{\ST@tableformat}%
233       \@tabletail\endtabular}%
234     \ST@tailht=\ht\@tempboxa\advance\ST@tailht\dp\@tempboxa
235   \fi
```

We add the average height of a line to this because when we decide to continue the tabular we need to have enough space left for one line and the tail.

```
236   \advance\ST@tailht by \ST@lineht
237   \ST@trace\tw@{Height of tail: \the\ST@tailht}%
```

```
238   \ST@trace\tw@{Maximum space for xtabular: \the\ST@pageleft}%
```

Now we decide whether we can continue on the current page or whether we need to start a new page. We assume that the minimum height of a tabular is the height of the head and tail and one line of data. If that doesn't fit, start a new page.

```
239   \@tempdima\ST@headht
240   \advance\@tempdima\ST@lineht
241   \advance\@tempdima\ST@tailht
```

Extension: I also add the height of the caption to the required space. The amount to be added depends on whether it is a top or bottom caption. Allowance is also made for skips around the caption.

```
242   \if@topcaption
243     \setbox\@tempboxa=\vbox{\PWST@thecaption}%
244     \PWSTcapht=\ht\@tempboxa\advance\PWSTcapht\dp\@tempboxa
245     \advance\PWSTcapht by 20\p@
246   \else
247     \PWSTcapht = 10\p@
248   \fi
249   \ST@trace\tw@{Caption height: \the\PWSTcapht}%
250   \advance\@tempdima\PWSTcapht
```

Continue with the original code.

```
251   \ST@trace\tw@{Minimum height of xtabular: \the\@tempdima}%
252   \ifnum\@tempdima>\ST@pageleft
253     \ST@trace\tw@{starting new page}%
```

Extension: The next line in the original code is `\newpage\@calnextpageht`. I need to start a new page, making allowance for the space required by the caption.

```
254     \newpage
255     \global\ST@pageleft\@colroom
256     \global\advance\ST@pageleft by -\PWSTcapht
257     \global\ST@pagesofar=\z@
```

Finish up with the original code.

```
258   \fi}  % end \@calfirstpageheight
259
```

\@calnextpageht This calculates the maximum height of the tabular on all subsequent pages of the supertabular environment.

```
260 \def\@calnextpageht{%
261   \ST@trace\tw@{Calculating height of xtabular on next page}%
262   \global\ST@pageleft\@colroom
263   \global\ST@pagesofar=\z@
264   \ST@trace\tw@{Maximum space for xtabular: \the\ST@pageleft}}
265
```

\PWSTcalchtlines *Extension:* A macro to calculate the space required by an empty table and the number of lines in an empty table.

The appropriate heads and tails are typeset in a temporary box so we can measure them.

```
266 \newcommand{\PWSTcalchtlines}{%
```

Measure the lasttail.

```
267   \setbox\@tempboxa=\vbox{\@arrayparboxrestore
268     \ST@restore
269     \expandafter\tabular\expandafter{\ST@tableformat}%
270     \@table@last@tail\endtabular}%
271   \PWST@ht=\ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
272   \global\PWST@lastht = \PWST@ht
```

And repeat for the lasthead.

```
273   \setbox\@tempboxa=\vbox{\@arrayparboxrestore
274     \ST@restore
275     \expandafter\tabular\expandafter{\ST@tableformat}%
276     \@table@last@head\endtabular}%
277   \PWST@ht = \ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
278   \global\advance\PWST@lastht by \PWST@ht
279   \ST@trace\tw@{Height of empty xtabular on last page = \the\PWST@lastht}%
```

Now repeat pretty well all of the above for a general table (i.e., one that is not on the first page nor the designated last page).

First the tail.

```
280   \setbox\@tempboxa=\vbox{\@arrayparboxrestore
281     \ST@restore
282     \expandafter\tabular\expandafter{\ST@tableformat}%
283     \@tabletail\endtabular}%
284   \PWST@ht=\ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
285   \global\PWST@generalht = \PWST@ht
```

And on to the general head.

```
286   \setbox\@tempboxa=\vbox{\@arrayparboxrestore
287     \ST@restore
288     \expandafter\tabular\expandafter{\ST@tableformat}%
289     \@tablehead\endtabular}%
290   \PWST@ht = \ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
291   \global\advance\PWST@generalht by \PWST@ht}
292
```

\PWSTcalnextpageht *Extension:* From some experiments that I ran it appeared as though the supertabular package ignored the possibility that the space required for the table header

and tail on pages after the first one might be different. If the subsequent head/tail combination were longer (i.e., took more vertical space) then the table could overflow the page. This is an attempt to fix this problem by calculating the actual minimum space required after the first page.

The calculations are similar to, but simpler, than those for `\@calfirstpageht`.

```
293 \newcommand{\PWSTcalnextpageht}{%
294   \ifnum\PWSTcurpage = \PWSTpenultimate
295     \ST@trace\tw@{Calculating height of xtabular on last page}%
```

We are on the penultimate page, so get the height of the last head/tail.

```
296       \PWST@ht=\PWST@lastht
```

Otherwise I need the general page data.

```
297   \else
298     \ST@trace\tw@{Calculating height of xtabular on next general page}%
299     \PWST@ht=\PWST@generalht
300   \fi
```

Having dealt with the two cases, I can now calculate the minimum space for a supertabular on the following page.

```
301   \global\ST@pageleft\@colroom
302   \global\advance\ST@pageleft -\PWST@ht
303   \global\ST@pagesofar=\z@
304   \ST@trace\tw@{Available space for xtabular: \the\ST@pageleft}}
305
```

`\x@supertabular` The various supertabular environments share a lot of code. Thus, to avoid needless repetition, the shared code is defined in this macro.

This macro has been modified as part of the supertabular extension.

```
306 \def\x@supertabular{%
```

First save the original definition of `\tabular` and then make it point to `\inner@tabular`. This is done to enable supertabular cells to contain a `tabular` environment without getting unexpected results when the `supertabular` would be split across this inner `tabular` environment.

```
307   \let\org@tabular\tabular
308   \let\tabular\inner@tabular
```

The same has to be done for the `tabular*` environment. The coding is more verbose.

```
309   \expandafter\let
310     \csname org@tabular*\expandafter\endcsname
311     \csname tabular*\endcsname
312   \expandafter\let\csname tabular*\expandafter\endcsname
313     \csname inner@tabular*\endcsname
```

Extension: The original code printed out the top caption at this point. If there is too little space on the first page of the table, the tabular data is printed on the following page. If this is the case (and its not known yet whether it is), then the caption should also be printed on the following page.

```
314 %%%%  \if@topcaption \@process@tablecaption \fi
```

Back to the original code. Save the original definition of \\.

315   \global\let\@oldcr=\\

Save the current value of \baselineskip, as we need it in the calculation of the average height of a line.

316   \def\baslineskp{\baselineskip}%

We have to check whether array.sty was loaded, because some of the internal macros have different names.

317   \ifx\undefined\@classix

Save old \@tabularcr and insert the definition of \@stabularcr.

318   \let\org@tabularcr\@tabularcr
319   \let\@tabularcr\ST@tabularcr

Activate the new parbox algorithm.

320   \let\org@startpbox=\@startpbox
321   \let\org@endpbox=\@endpbox
322   \let\@@startpbox=\ST@startpbox
323   \let\@@endpbox=\ST@endpbox
324   \else

When array.sty was loaded things are a bit different.

325   \let\org@tabularcr\@arraycr
326   \let\@arraycr\ST@tabularcr
327   \let\org@startpbox=\@startpbox
328   \let\org@endpbox=\@endpbox
329   \let\@startpbox=\ST@astartpbox
330   \let\@endpbox=\ST@aendpbox
331   \fi

Check if the head of the table should be different for the first and subsequent pages.

332   \ifx\@table@first@head\undefined
333   \let\@@tablehead=\@tablehead
334   \else
335   \let\@@tablehead=\@table@first@head
336   \fi

The first part of a supertabular may be moved to the next page if it doesn't fit on the current page. Subsequent parts can not be moved; therefore we will have to switch the definition of \ST@skippart around.

337   \let\ST@skippage\ST@skipfirstpart

Now we can estimate the average line height and the height of the first page of the supertabular.

338   \estimate@lineht
339   \@calfirstpageht

Extension: Call the macro to initialize the extension code for this table.

340   \PWSTinit

Extension: At this point I know, and have adjusted for, the page on which the first part of the table will be printed. It should now be safe to print the top caption, if any. Unfortunately, in spite of everthing, the TeX page breaking mechanism might still think that there is too little space left.

```
341    \if@topcaption \@process@tablecaption \fi
342    \noindent
```

Extension: Finally, subtract the space required by the header and the tail (as these don't update the available space when output).

```
343    \global\advance\ST@pageleft -\ST@headht%
344    \ST@trace\tw@{Available space after accounting for header: \the\ST@pageleft}%
345    \global\advance\ST@pageleft -\ST@tailht%
346    \ST@trace\tw@{Available space after accounting for tail: \the\ST@pageleft}}
347
```

\PWSTinit *Extension:* This routine initialises the extension data.

```
348 \newcommand{\PWSTinit}{%
```

At the end of processing each supertabular (see later) the number of pages consumed by the supertabular is written to the `.aux` file. At the start of a supertabular, after incrementing the number of supertabulars processed, the prior number of pages are read from the file. These are stored in *PWSTlastpage*.

```
349    \global\advance\c@PWSTtable\@ne
350    \global\expandafter\let\expandafter\PWSTtempc
351      \csname PWST@\romannumeral\c@PWSTtable\endcsname
```

I have to take account of the fact that there might be no entry in the `.aux` file, and hence the lastpage number might not be set.

```
352    \ifx\PWSTtempc\relax
353      \ST@trace\tw@{Table \the\c@PWSTtable: Processing for the first time}%
354      \PWSTlastpage=\@m% = 1000
355    \else
356      \PWSTlastpage=\PWSTtempc
357    \fi
358    \ST@trace\tw@{Table \the\c@PWSTtable: last page set to \the\PWSTlastpage}%
```

Set the current page counter to unity.

```
359    \PWSTcurpage=\@ne
```

Perform the calculations for the empty table data.

```
360    \PWSTcalchtlines
```

Initialise the line counter and set `firstcall` to TRUE.

```
361    \global\PWSTlines=\z@
362    \global\firstcalltrue
```

If we have the `iso` class, then I have to flag that we are in a 'float'.

```
363      \infloattrue}
364
```

\xtabular We start by looking for an optional argument, which will be duly ignored as it seems to make no sense to try to align a multipage table in the middle...

Extension: Use xtabular instead of supertabular, and similarly for the others, so this will not be mentioned explicitly again.

```
365 \def\xtabular{%
366   \@ifnextchar[{\@supertabular}%]
367                {\@supertabular[]}}
```

\@supertabular We can now save the preamble of the tabular in a macro.

```
368 \def\@supertabular[#1]#2{%
369   \def\ST@tableformat{#2}%
370   \ST@trace\tw@{Starting a new xtabular}%
```

Then remember that this is not a supertabular* environment.

```
371   \global\ST@starfalse
```

Don't use minipages.

```
372   \global\ST@mpfalse
```

Most of the following code is shared between the supertabbular and supertabular* environments. So to avoid duplication it is stored in a macro.

```
373   \x@supertabular
```

Finally start a normal tabular environment.

```
374   \expandafter\org@tabular\expandafter{\ST@tableformat}%
375   \@@tablehead}
376
```

\xtabular* We start by looking for the optional argument of the tabular environment.

```
377 \@namedef{xtabular*}#1{%
378   \@ifnextchar[{\@nameuse{@supertabular*}{#1}}%]
379                {\@nameuse{@supertabular*}{#1}[]}%]
380   }
```

We start by saving the intended width and the preamble of the tabular*.

```
381 \@namedef{@supertabular*}#1[#2]#3{%
382   \ST@trace\tw@{Starting a new xtabular*}%
383   \def\ST@tableformat{#3}%
384   \ST@wd=#1\relax
385   \global\ST@startrue
386   \global\ST@mpfalse
```

Now we can call the common code for both environments.

```
387   \x@supertabular
```

And we can start a normal tabular* environment.

```
388   \expandafter\csname org@tabular*\expandafter\endcsname
389   \expandafter{\expandafter\ST@wd\expandafter}%
390   \expandafter{\ST@tableformat}%
391   \@@tablehead}
392
```

\mpxtabular   This version of the supertabular environment puts each tabular into a minipage, thus making footnotes possible. We start by looking for an optional argument, which will be ignored as it makes no sense to try and align a multipage table in the middle...

```
393 \def\mpxtabular{%
394   \@ifnextchar[{\@mpsupertabular}%]
395                {\@mpsupertabular[]}}
```

We can now save the preamble in a macro.

```
396 \def\@mpsupertabular[#1]#2{%
397   \def\ST@tableformat{#2}%
398   \ST@trace\tw@{Starting a new mpxtabular}%
```

Remember that this is not a `mpsupertabular*` environment and also note we have to close the minipage later.

```
399   \global\ST@starfalse
400   \global\ST@mptrue
```

Since we are about to start a minipage of `\columnwidth` the horizontal alignment will not work. We have to remember the values and then restore them inside the minipage.

```
401   \ST@rightskip \rightskip
402   \ST@leftskip \leftskip
403   \ST@parfillskip \parfillskip
```

Call the code that is common to all the environments.

```
404   \x@supertabular
```

Finally, start a normal `tabular`

```
405   \minipage{\columnwidth}%
406   \parfillskip\ST@parfillskip
407   \rightskip \ST@rightskip
408   \leftskip \ST@leftskip
409   \noindent\expandafter\org@tabular\expandafter{\ST@tableformat}%
410   \@@tablehead}
411
```

\mpxtabular*   We start by looking for the optional argument of the tabular environment.

```
412 \@namedef{mpxtabular*}#1{%
413   \@ifnextchar[{\@nameuse{@mpsupertabular*}{#1}}%
414                {\@nameuse{@mpsupertabular*}{#1}[]}%]
415 }
```

Now we can save the intended width and the preamble of the `tabular*`.

```
416 \@namedef{@mpsupertabular*}#1[#2]#3{%
417   \ST@trace\tw@{Starting a new mpxtabular*}%
418   \def\ST@tableformat{#3}%
419   \ST@wd=#1\relax
420   \global\ST@startrue
421   \global\ST@mptrue
422   \ST@rightskip \rightskip
```

```
423    \ST@leftskip \leftskip
424    \ST@parfillskip \parfillskip
```

Now is the time to call the common code for both environments.

```
425    \x@supertabular
```

And we can start a normal `tabular*` environment.

```
426    \minipage{\columnwidth}%
427    \parfillskip\ST@parfillskip
428    \rightskip \ST@rightskip
429    \leftskip \ST@leftskip
430    \noindent\expandafter\csname org@tabular*\expandafter\endcsname
431    \expandafter{\expandafter\ST@wd\expandafter}%
432    \expandafter{\ST@tableformat}%
433    \@@tablehead}%
434
```

\endxtabular   These close the `xtabular` and `xtabular*` environments.
\endxtabular*   For the extension, this macro has been modified to write out to the `.aux` file
the number of pages used for the supertabular.

```
435  \def\endxtabular{%
436    \ifx\@table@last@tail\undefined
437      \@tabletail
438    \else
439      \@table@last@tail
440    \fi
441    \csname endtabular\ifST@star*\fi\endcsname
```

While studying the original code to determine where additions were needed for
the extension, I realized that the last part of the `\end...` code was common to all
the environments. I have broken it out into a seperate routine which also includes
the modification needed for the extension.

```
442    \x@endsupertabular
```

And back to the original code.

```
443    \ST@trace\tw@{Ended a xtabular\ifST@star*\fi}}
```

The definition of the ending of the `xtabular*` environment is simple:

```
444 \expandafter\let\csname endxtabular*\endcsname\endxtabular
```

\x@endsupertabular   This macro contains the code that is common to all the `\end...` commands. It
includes the modification required for the extension.

```
445 \newcommand{\x@endsupertabular}{%
```

Restore the original definition of `\@tabularcr`

```
446    \ST@restore
```

Check if we have to insert a caption and restore to default behaviour of putting
captions at the top.

```
447    \if@topcaption
448    \else
```

24

```
449        \@process@tablecaption
450        \global\@topcaptiontrue
451     \fi
```

Restore the meaning of \\ to the one it had before the start of this environment. Also re-initialize some control-sequences

```
452     \global\let\\=\@oldcr
453     \global\let\@table@first@head\undefined
454 %%%  \global\let\@table@last@tail\undefined
455     \global\let\@process@tablecaption\relax
```

Extension: For the extension, write the number of the last page to the `.aux` file. Also, if we are in the `iso` class, reset the 'float' flag.

```
456     \PWSToplastpagenum
457        \infloatfalse}
458
```

\PWSToplastpagenum   *Extension:* This routine is responsible for writing the number of the last page of the supertabular to the `.aux` file.

What gets written is `\PWST@vi{4}`, assuming that the value of *c@PWSTtable* is 6 and the value of *PWSTcurpage* is 4.

```
459 \newcommand{\PWSToplastpagenum}{%
```

There are a number of cases to consider. The first decision is whether the current page is the previously calculated last page.

```
460    \ifnum\PWSTcurpage=\PWSTlastpage
```

The current table ends on the calculated last page. There are four cases to consider:

1. The table has not overflowed (`firstcall` is TRUE) and the table is not empty — this page is still the last page.

2. The table has not overflowed (`firstcall` is TRUE) and the table is empty — this page is after the actual last page, so decrease the page number.

3. The table has overflowed (`firstcall` is FALSE) and the overflow is large enough to generate a non-empty table on the next page — increment the page number.

4. The table has overflowed (`firstcall` is FALSE) and the overflow is small enough to generate an empty table on the next page — this page is still the last page.

```
461     \iffirstcall  % on last, no overflow
462 %%       \ifnum\PWSTlines < \PWSTlasthead % this table is empty
463       \ifnum\PWSTlines < \@ne           % this table is empty
464         \global\advance\PWSTcurpage \m@ne
465       \fi
466     \else % overflow
467 %%       \ifnum\PWSTlines > \tw@ % enough for another page
```

```
468        \ifnum\PWSTlines > \z@     % enough for another page
469          \global\advance\PWSTcurpage \@ne
470        \fi
471      \fi
472    \else
```

The table has ended on a page that is not the calculated last page. If the table is empty, then decrement the page number, else this is the last page.

```
473 %%      \ifnum\PWSTlines < \PWSThead % empty table
474    \ifnum\PWSTlines < \@ne          % empty table
475      \global\advance\PWSTcurpage \m@ne
476    \fi
477  \fi
```

Finally, write out the 'new' last page number.

```
478    \if@filesw\immediate\write\@auxout%
479      {\gdef\string\PWST@\romannumeral\c@PWSTtable{\the\PWSTcurpage}}%
480      \ST@trace\tw@{Table \the\c@PWSTtable:\MessageBreak
481                  wrote \the\PWSTcurpage\space as the last page}%
482    \fi}
483
```

\endmpxtabular These close the mpxtabular and mpxtabular* environments.
\endmpxtabular*
```
484 \def\endmpxtabular{%
485    \ifx\@table@last@tail\undefined
486      \@tabletail
487    \else
488      \@table@last@tail
489    \fi
490    \csname endtabular\ifST@star*\fi\endcsname
491    \endminipage
```

Now call the common code for all \end....

```
492    \x@endsupertabular
```

Finish per the original code.

```
493    \ST@trace\tw@{Ended an mpxtabular\ifST@star*\fi}}
```

The definition of the ending of the mpxtabular* environment is simple:

```
494 \expandafter\let\csname endmpxtabular*\endcsname\endmpxtabular
495
```

\ST@restore This macro restores the original definitions of the macros that handle parbox entries and the 'end of row' macros.

```
496 \def\ST@restore{%
497    \ifx\undefined\@classix
498      \let\@tabularcr\org@tabularcr
499    \else
500      \let\@arraycr\org@tabularcr
501    \fi
502    \let\@startpbox\org@startpbox
```

```
503    \let\@endpbox\org@endpbox}
504
```

\inner@tabular
\inner@tabular* In order to facilitate complete `tabular` environments to be in a cell of a `supertabular` we need to adapt the definition of the original environments. For the inner `tabular` a number of definitions have to be restored.

```
505 \def\inner@tabular{%
506    \ST@restore
507    \let\\=\@oldcr
508    \noindent
509    \org@tabular}
510 \@namedef{inner@tabular*}{%
511    \ST@restore
512    \let\\=\@oldcr
513    \noindent
514    \csname org@tabular*\endcsname}
515
```

\ST@cr This macro is called by each `\\` inside the tabular environment. It updates the estimate of the amount of space left on the current page and starts a new page if necessary.

```
516 \def\ST@cr{%
517    \noalign{%
518    \ST@trace\thr@@{Parbox height: \the\ST@pboxht\MessageBreak
519                    Line height: \the\ST@lineht}%
520    \ifnum\ST@pboxht<\ST@lineht
```

If there is a non-empty line, but an empty parbox, then `\ST@pboxht` might be non-zero, but too small thereby breaking the algorithm. Therefore we estimate the height of the line to be `\ST@lineht` in this case, and store it in `\ST@prevht`.

```
521       \global\advance\ST@pageleft -\ST@lineht
522       \global\ST@prevht\ST@lineht
523    \else
```

When the parbox is not empty we take its height into account plus a little extra.

```
524       \global\advance\ST@pboxht \PWST@xentrystretch\ST@pboxht
525       \global\advance\ST@pboxht \ST@stretchht
526       \ST@trace\thr@@{Added par box with height \the\ST@pboxht}%
527       \global\advance\ST@pageleft -\ST@pboxht
528       \global\ST@prevht\ST@pboxht
529       \global\ST@pboxht\z@
530    \fi
```

`\ST@toadd` is the value of the optional argument of `\\`.

```
531       \global\advance\ST@pageleft -\ST@toadd
532       \global\ST@toadd=\z@
533       \ST@trace\thr@@{Space left for xtabular: \the\ST@pageleft}%
```

Extension: Increment the line number at this point.

```
534       \global\advance\PWSTlines \@ne
```

```
535     \ST@trace\thr@@{Line counter incremented by one to: \the\PWSTlines}%
536   }% end of noalign
```

In general, when the `\ST@pageleft` has become negative, the last row was so high that the supertabular doesn't fit on the current page. In this case we skip the current page and start at the top of the next one; otherwise TEX will move this part of the table to a new page anyway, probably with a message about an overfull `\vbox`.

Extension: For the extension I do some special handling if we are on the last page. Essentially the idea is not to start a new page, but to continue on the current page, noting any overflow.

```
537   \ifnum\PWSTcurpage=\PWSTlastpage
538     \PWST@lastpagecr
539   \else
```

Execute the original code.

```
540     \ifnum\ST@pageleft<\z@
541       \ST@skippage
542     \else
```

When there is not enough space left on the current page, we start a new page. To compute the amount of space needed we use the height of the previous line (`\ST@prevht`) as an estimate of the height of the next line. If we are processing an `mpsupertabular` we also need to take the height of the footnotes into account.

```
543       \noalign{\global\@tempdima\ST@tailht
544         \global\advance\@tempdima\ST@prevht
545       \ifST@mp
546         \ifvoid\@mpfootins\else
547           \global\advance\@tempdima\ht\@mpfootins
548           \global\advance\@tempdima 3pt
549         \fi
550       \fi}% end noalign
551       \ifnum\ST@pageleft<\@tempdima
552         \ST@newpage
553       \else
```

This line is necessary because the tablehead has to be inserted *after* the `\if\else\fi`-clause. For this purpose `\ST@next` is used. In the middle of tableprocessing it should be an *empty* macro (*not* `\relax`).

```
554         \noalign{\global\let\ST@next\@empty}%
555       \fi
556     \fi
```

Extension: Close off the `\iflastpage`;

```
557   \fi
```

and finish per the original code.

```
558   \ST@next}
559
```

\PWST@lastpagecr   *Extension:* This routine handles newlines on the last page of a supertabular. The idea is that when we are on the last page the table continues to be processed until the end without calling for a newpage even if the table will be too long. I do need to record whether or not the table has 'overflowed' the allowable space on the page. The code is very similar to the last part of the code for `\ST@cr`.

```
560 \newcommand{\PWST@lastpagecr}{%
561   \noalign{%
562     \ifnum\ST@pageleft<\z@
```

The table has overflowed, so record the fact.

```
563       \PWST@setfirstcall
564     \fi
```

Now continue along the lines of `\ST@cr`.

```
565     \global\@tempdima\ST@tailht
566     \global\advance\@tempdima\ST@prevht
567       \ifST@mp
568         \ifvoid\@mpfootins\else
569           \global\advance\@tempdima\ht\@mpfootins
570           \global\advance\@tempdima 3pt
571         \fi
572       \fi
573       \ifnum\ST@pageleft<\@tempdima
```

Again, the table has overflowed.

```
574         \PWST@setfirstcall
575       \fi
```

Finish like `\ST@cr`.

```
576       \global\let\ST@next\@empty
577     }}
578
```

\PWST@setfirstcall   *Extension:* This routine records that a table on the last page has overflowed by setting `firstcall` to FALSE. If it is the first such overflow it also zeroes the line counter.

```
579 \newcommand{\PWST@setfirstcall}{%
580   \iffirstcall
581     \global\firstcallfalse
582     \global\PWSTlines=\z@
583     \ST@trace\thr@@{Overflow on last page. Line counter set to \the\PWSTlines}%
584   \fi}
585
```

\ST@skipfirstpart   This macro skips the current page and moves the entire supertabular that has been built so far to the next page.

```
586 \def\ST@skipfirstpart{%
587   \noalign{%
588     \ST@trace\tw@{Tabular too high, moving to next page}%
```

In order for this to work properly we need to adapt the value of `\ST@pageleft`. When this macro is called it has a negative value. We should add the height of the next page to that (`\@colroom`). From the result the 'normal' height of the supertabular should be subtracted (`\@colroom - \pagetotal`). This could be coded as follows:

```
\ST@dimen\@colroom
\advance\ST@dimen-\pagetotal
\global\advance\ST@pageleft\@colroom
\global\advance\ST@pageleft-\ST@dimen
```

However, note that `\@colroom` is added *and* subtracted. Thus the code can be simplified to:

```
589    \global\advance\ST@pageleft\pagetotal
```

Then we can set `\ST@pagesofar` to zero and start the new page.

```
590    \global\ST@pagesofar\z@
591    \newpage
```

Finally we make sure that this macro can only be executed once for each supertabular by changing the definition of `\ST@skippage`.

```
592    \global\let\ST@skippage\ST@newpage
593  }}
594
```

`\ST@newpage` This macro performs the actions necessary to start a new page.

This macro is also modified for the extension to supertabluar.

```
595 \def\ST@newpage{%
596   \noalign{\ST@trace\tw@{Starting new page, writing tail}}%
```

Output `\tabletail`, close the tabular environment, close a minipage if necessary, output all material and start a fresh new page.

```
597   \@tabletail
598   \ifST@star
599     \csname endtabular*\endcsname
600   \else
601     \endtabular
602   \fi
603   \ifST@mp
604     \endminipage
605   \fi
```

Then we make sure that `\ST@skippage` can no longer be executed for this supertabular by changing its definition.

```
606   \global\let\ST@skippage\ST@newpage
```

On with the output.

Extension: The original code had the next line as `\newpage\@calnextpageht`. However, if the general header has a vertical height that differs from the first

header, then the table on the continuation pages may run short or, more discon-
certing, long. The extension, I think, cures that by using a different algorithm to
calculate the height on the next page.

```
607    \newpage\PWSTcalnextpageht
608    \ST@trace\tw@{writing head}%
```

Extension: The original code just let \ST@next to \@tablehead. The extension
has to handle the special case of of the heading on the last page.

```
609    \PWSTsethead
```

Now we are back to the original supertabular code.

```
610    \ifST@mp
611      \noindent\minipage{\columnwidth}%
612      \parfillskip\ST@parfillskip
613      \rightskip \ST@rightskip
614      \leftskip \ST@leftskip
615    \fi
616    \noindent
617    \ifST@star
618      \expandafter\csname org@tabular*\expandafter\endcsname
619      \expandafter{\expandafter\ST@wd\expandafter}%
620      \expandafter{\ST@tableformat}%
621    \else
622      \expandafter\org@tabular\expandafter{\ST@tableformat}%
623    \fi}
624
```

\PWSTsethead *Extension:* This is more extension code for use within \ST@newpage. It provides
the proper table head for the page about to be processed.

```
625 \newcommand{\PWSTsethead}{%
```

First the line counter is zeroed.

```
626    \global\PWSTlines=\z@
627    \ST@trace\thr@@{Newpage, line counter set to: \the\PWSTlines}%
```

The current page counter is incremented and it is checked against the old page
counter to see if this is the last page of this supertabular.

```
628    \global\advance\PWSTcurpage\@ne
629    \ST@trace\tw@{Table \the\c@PWSTtable:\MessageBreak
630               current page = \the\PWSTcurpage,\MessageBreak
631               last page = \the\PWSTlastpage}%
632    \ifnum\PWSTcurpage=\PWSTlastpage
633      \ST@trace\tw@{Newpage is the last page}%
```

We are on the last page. If there are more than one pages and the last table heading
has been specified, then the heading is set to \@table@last@head, otherwise it is
set to \@tablehead.

```
634      \ifnum\PWSTcurpage>\@ne
635        \ifx\@table@last@head\relax
636          \let\ST@next\@tablehead
637          \ST@trace\tw@{Set heading to tablehead}%
```

```
638      \else
639        \let\ST@next\@table@last@head
640        \ST@trace\tw@{Set heading to tablelasthead}%
641      \fi
642    \fi
643  \else
```

We are not on the last page, so just set the heading to `\@tablehead`.

```
644    \let\ST@next\@tablehead
645    \ST@trace\tw@{Set heading to tablehead}%
646  \fi}
647
```

The end of this package

648 ⟨/xtab⟩

References

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Publishing Company, 1994.

[Wil96]   Peter R. Wilson. *LaTeX for standards: The LaTeX package files user manual.* NIST Report NISTIR, June 1996.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.