The boxhandler Package

Tools for Optimizing Captions, Presentation, and Placement of Tables and Figures

Steven B. Segletes
steven.b.segletes.civ@mail.mil

2012/10/18
v1.30

Abstract

This package facilitates the optimized presentation of LaTeX tables and figures. Not only can boxhandler conveniently lay out table and figure captions with a wide variety of stylistic appearances, but allows for figures and tables to be "wrapped" in a manner consistent with many business and government documents. For a document that might appear in different venues with different formatting, boxhandler very powerfully permits the creation of a LaTeX source document that can, with a single-line change in the source code, produce an output that has vastly different layout from the baseline configuration, both in terms of caption style, and in terms of the locations where figures, tables and lists appear (or not) in the document. Deferral routines also allow one to keep all figure and table data in a separate source file, while nonetheless producing a document with figures and tables appearing in the desired location in the document.

Contents

1 Introduction

The commands described in the boxhandler style accomplish a number of useful functions. Even without the use of a conditionally-compiled document:

1. they allow the definition of figures and tables in compact routine calls;
2. they retain wide flexibility in caption appearance & table/figure placement, through the use of simple setup calls.

And for those who wish, from a single LaTeX source file to conditionally produce, on one hand, an internal technical report for example, and on the other hand, a journal manuscript submission, this style additionally allows:

3. printing of figures & tables to be [conditionally] deferred to later in the document;
4. printing of lists (*lof*, *lot*) to be deferred later in document; and
5. the preemptive cancellation of the *toc*, *lof*, and *lot*.

2 Caption Style, Appearance and Box Placement

2.1 Commands

This routine provides several routines for creating and tailoring the appearance of captioned "boxes" (that is, figures and tables). They include:

`\bxtable[`*loc*`]{`*caption*`}{`*boxed object*`}`
`\bxfigure[`*loc*`]{`*caption*`}{`*boxed object*`}`
`\relaxCaptionWidth[`*len*`]`
`\limitCaptionWidth[`*len*`]`
`\constrainCaptionWidth[`*len*`]{`*len*`}`

`\captionStyle{`*offset type*`}{`*alignment type*`}`
`\hyperactive[`*len*`]`

`\bxtable` `\bxfigure`

The routines `\bxtable` and `\bxfigure` will actually produce the complete table or figure, including caption. In these two routines, *loc* is an optional argument. It can take on a value of: `ht`, `hb`, `t`, `b`, or `p`. This parameter refers to the same location argument that goes with the LaTeX float environments, to help LaTeX determine the placement of the item on your page. `h` is 'here', `t` is 'top', `b` is 'bottom', `p` is 'page of floats'. Omitting the first argument just uses the default placement of the table and figure environments.

In `\bxtable` and `\bxfigure`, *caption* is the argument that will eventually get passed on to the `\caption` call, after the routine properly formulates it to match the desired figure/caption style. The caption may include `\label` identifiers to be referenced by the main text.

For `\bxtable`, the *boxed object* will typically be a tabular box, though any LaTeX boxed object will satisfy the routine. Thus, the difference between a `\bxtable` and a `\bxfigure` is largely a semantic one. The true difference between `\bxtable` and `\bxfigure` in this regard is that the table caption is placed above the table, whereas the figure caption is placed below it. In addition, because these routines make use of the standard `\caption` call within the table and figure environments, calls to `\bxtable` and `\bxfigure` will have their references automatically available to the List of Tables and List of Figures, respectively.

One of the differences between the boxhandler style and LaTeX's default captioning environment is the default caption width. Whereas LaTeX defaults to a full margin-to-margin caption width, boxhandler defaults to a caption width equal to and aligned with the width of the box being captioned. However, this caption-width default within boxhandler can be changed to any arbitrary value, including full-width, if desired. The caption width, through the use of the "dead margin," can also be conveniently set to a fixed offset from the table or figure width. For example, all table captions could be set to a width 1 inch larger than their associated tables.

`\relaxCaptionWidth`

The routine `\relaxCaptionWidth` takes as its optional argument a length dimension corresponding to the minimum allowed caption width, even if an associated table or figure is of lesser box width. While this command might technically violate an organization's style guideline, it allows one to avoid the situation of trying to assign a caption to fit the width dimension of an extremely narrow table or figure. An example of this use is shown in Tables 1 and 2. In the first table, the default minimum caption width of 1 inch is retained, resulting in an unwieldy caption. The second table has been printed following an invocation of `\relaxCaptionWidth[4.0in]`. Using `\relaxCaptionWidth` with no argument resets the minimum allowed caption width to the default value of 1 inch.

If the minimum caption width is relaxed, the maximum allowed width will be bumped up, if necessary, to remain greater than or equal to the minimum allowed caption width.

Table 1. A table which provides the secret entry-word of the Halloween Ghost Club, assuming you can read the caption.

Boohoocachoo

Table 2. A table which provides the secret entryword of the Halloween Ghost Club, assuming you can read the caption.

Boohoocachoo

`\limitCaptionWidth` The routine `\limitCaptionWidth` is analogous to `\relaxCaptionWidth`. In this case, however, the maximum allowable caption width will be defined. If no argument is specified, the maximum allowed caption width is reset to the default value, which is `\textwidth`.

If the maximum caption width is constrained, the minimum allowed width will be reduced, if necessary, to remain less than or equal to the maximum allowed caption width.

`\constrainCaptionWidth` The minimum and maximum allowed caption widths may be simultaneously specified with `\constrainCaptionWidth`. The order of the two length arguments is not important. Omitting the optional argument will cause both the min and max allowable caption widths to be fixed at the specified value. In this manner, all caption widths will be set to this single value, regardless of the size of the table or figure box it describes. Note that the use of `\constrainCaptionWidth` with a single argument of `{\textwidth}` allows a full-width left-to-right margin caption style to be achieved, if desired, as in Figure 1.

`\captionStyle` To understand the the figure and table caption style command, examine the caption styles in Figures 2–5, in reference to the boxed object with which they are paired. `\captionStyle{`*offset type*`}{`*alignment type*`}` is a compact way of specifying the caption style. The first parameter specifies the *offset type* for long

captions (since "offset" has no meaning for short captions). It can take the value of `o` for "offset" captions, such as that found in Figure 2, or `n` for "nooffset" captions, such as that found in Figure 3.

The second parameter specifies the *alignment type* for short captions (since long captions are already fully aligned). It can take on the value of `l` for "left"-aligned captions, such as that in Figure 4, the value `c` for "center"-aligned captions, such as that in Figure 5, or the value of `r` for "right"-aligned captions.

The default style for this package is `\captionstyle{o}{l}`, or "offset"-style, "left"-aligned captions. This default caption style is that displayed in Figures 2 and 4, for long and short captions, respectively.

Note that caption alignment is not the same as caption justification. Regardless of alignment, any caption of size sufficient to span the full width of the caption box will be, by default, fully justified or "flushed" within that caption box. The captions in Figures 1 and 2 demonstrate this for both offset and nooffset caption types. Text justification can, however, be altered, as described later by the `\CaptionJustification` parameter.

`\hyperactive` `\hyperactive` was added in v1.22 to provide compatibility with the hyperref package, which is known to redefine many LaTeX variables, including `\caption`. It was found, when the hyperref package was active, offset captions experienced a vertical gap between the caption label and the caption text. The command `\hyperactive` should be invoked only if the hyperref package is being utilized. The optional argument to this command is the length that boxhandler will shift the caption text downward, so as to achieve alignment with the caption label. The default value is -1.55ex, which was the required corrective shift observed for some applications, when the hyperref package was active.

2.2 Additional User Parameters

In addition to the above commands, there are a variety of lengths, counters, and modes, which may be set by the user, to adjust the appearance of the caption presentation. The settings for all these parameters hold until and unless subsequently reset by the user.

```
\setlength\captionGap{len}
\setlength\TableDeadMargin{len}
\setlength\FigureDeadMargin{len}
\setcounter{abovecaptionskipterm}{integer}
\setcounter{belowcaptionskipterm}{integer}
\let\CaptionFontSize fontsize, e.g., \small
\let\TableFontSize fontsize, e.g., \small
\def\LRTablePlacement{flushleft, center, or flushright}
\def\LRFigurePlacement{flushleft, center, or flushright}
\def\CaptionJustification{blank, \raggedright,
                          \centering, or \raggedleft}
```

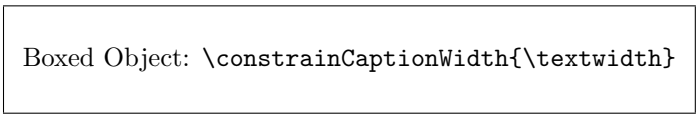

Figure 1. Here is a full-width caption that takes up the full margin-to-margin dimension, regardless of how wide the boxed object it serves is. In this case, the caption is of the "nooffset" variety.

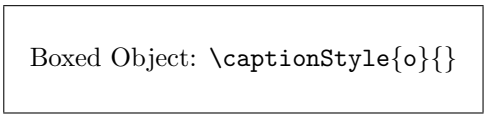

Figure 2. For "offset" captions, the ID label for the caption is offset to the left of the caption text.

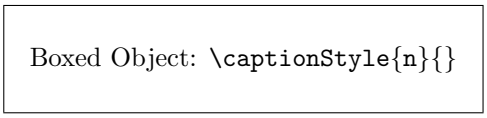

Figure 3. For "nooffset" captions, the caption's ID label is integrated into the caption text itself.

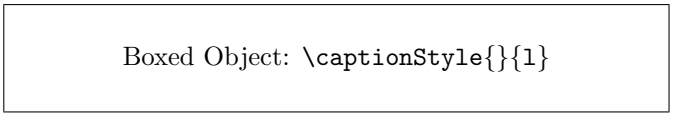

Figure 4. A short "left"-aligned caption.

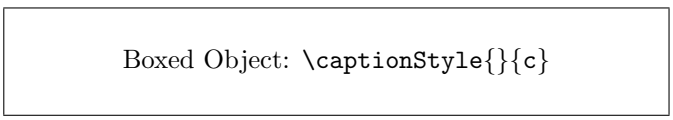

Figure 5. A short "center"-aligned caption.

\captionGap

\captionGap defines the horizontal space between the caption identifier (e.g., "Figure 1.") and the start of the caption text itself. Its default value is 1ex.

\TableDeadMargin
\FigureDeadMargin

\TableDeadMargin and \FigureDeadMargin may be set to correct the caption alignment for boxes that have a deadzone around their border. Such dead space takes up boxwidth, but shows no visible data. These parameter values should be set to the left or right-hand dead space (assumed symmetric). The default value for \TableDeadMargin, which is suitable for LaTeX generated tabular data, is 0.375em. The default for \FigureDeadMargin is 0em.

Additionally, these \...DeadMargin commands can be used to set the box-size-to-caption-size disparity to any desired non-flush value. As an example, setting the value of \FigureDeadMargin to 0.5 inch will make the figure captions always 1 inch smaller than the actual figure size. Setting it to -0.5 inch (*a negative value!*) will make the caption always 1 inch larger than the actual figure size (within the error of the actual figure dead margin width, and subject to the caption width min/max constraints).

\abovecaptionskipterm
\belowcaptionskipterm

The quantities \abovecaptionskipterm and \belowcaptionskipterm are related to LaTeX's \abovecaptionskip and \belowcaptionskip functions, defining the white- space above and below a caption. Unlike the corresponding LaTeX functions, the parameters here are integers (not lengths). The terms represent multipliers of the LaTeX length measure \p@ to be used for the above- and below-captionskip. Their default values are 10 and 7, respectively. These values affect only captions that are created as part of bxtable and bxfigure, and do not affect the \abovecaptionskip and \belowcaptionskip definitions intrinsic to your default document class.

\CaptionFontSize

\CaptionFontSize defines the default size of the caption font, for example , \large, \scriptsize, etc. The default value is \small.

\TableFontSize

\TableFontSize defines the default size of the font that appears within the tables themselves. Its default value is \small.

\LRTablePlacement
\LRFigurePlacement

\LRTablePlacement and \LRFigurePlacement define the horizontal placement of tables and figures with respect to the paper margins. The options for these two parameters include {flushleft}, {center} and {flushright}. The default is {center}. This is different from the "left", "center", and "right" alignment modes for captions, which align short captions with respect to the boxed data.

\CaptionJustification

By default, any caption that spans the entire width of the caption box will be fully justified, or "flushed" with respect to the caption box margins. However, this behavior can be reset by redefining the parameter \CaptionJustification. For a ragged-right style within the caption box, the definition should be set to \raggedright. For the brave and daring, \raggedleft and/or \centering can be explored. Use \def\CaptionJustification{} to reset subsequent captions for full flushing.

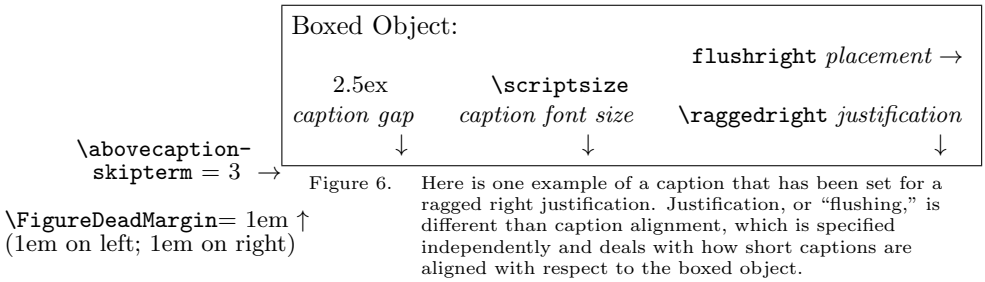

Figure 6. Here is one example of a caption that has been set for a ragged right justification. Justification, or "flushing," is different than caption alignment, which is specified independently and deals with how short captions are aligned with respect to the boxed object.

Figure 6 is provided to demonstrate some of these features including: caption justification (`\raggedright`), caption gap (2.5 ex), caption font size (`\scriptsize`), LR figure placement (`flushright`), `abovecaptionskipterm` (3), and a value for `\FigureDeadMargin` of 1em.

3 Figure and Table "Wrappers"

With boxhandler v1.20, figure and table wrappers have been added. A wrapper is here defined to mean an item that bounds a figure or table by appearing in the upper-left and lower-right corners of the figure or table. It could be an iconic image such as the company logo, a reminder such as "COMPANY PROPRIETARY", or some other such delimiter to the figure/table. The relevant commands to use them are as follows:

```
\WrapperOn[default wrapper]
\WrapperOff
\Wrapper{custom wrapper}
\def\WrapperTextStyle{text style}
```

By default, wrappers are turned off in boxhandler. They may be activated with the command `\WrapperOn`. The optional argument to `\WrapperOn`, which should be used on the initial invocation, specifies the default wrapper. Likewise, wrappers may be disabled with the command `\WrapperOff`. When wrappers are activated, every table and figure subsequently created will be wrapped using the default wrapper.

If, however, the user would like for a given figure or table to have a custom wrapper different than the default, the `\Wrapper{}` command should be used within the second mandatory argument of the call to `\bxtable` or `\bxfigure`. That is to say, the call to `\Wrapper` should be included within the argument where the actual figure or table contents are defined.

Here is an example:

```
\bxfigure[ht]{Widget details for the XMC-7936}%
  {\fbox{\hspace{1in}\rule[-.5ex]{3ex}{3ex} $\rightarrow$%
   \rule[-.2ex]{2ex}{2ex}\hspace{1in}}\Wrapper{PROPRIETARY}}%
```

Figure 7. Widget details for the XMC-7936.

PROPRIETARY

Wrapper text (both default and custom) are given a style defined by `\WrapperTextStyle`. The default style is small, boldface text, `\bf\small`.

4 Box and List Deferral/Preemption

4.1 Description of Problem

For those using conditional compilation to create various versions of the same basic document, the boxhandler package can provide great utility. The package has been designed to permit the deferral and preemption of certain aspects of the document, such as figures, tables and lists. With such a capability, one has the capacity to not only change the appearance of the alternate document version, but to fundamentally change its layout too. Table 3 provides a simple example of how a document may be set up for conditional compilation.

In this example, by setting the parameter in the first line of the document to either `\TECHRPT` or `\MANUSCRIPT`, a different collection of setup routines may be invoked for each particular document style. This works great for such parameters that affect appearance, but not placement of the LaTeX entities. For example, changing fonts, margins, indents, etc. works fine with the conditional code shown in Table 3. Even when a particular document class has unique commands not found in other classes, a converter style file may be created (such as `TR2article` in the Table 3 example) to deal rationally with invocations of class-specific features.

Where great difficulty arises is when the different styles desired of a conditional compilation utilize fundamentally different layouts of the principal document elements, such as figures, tables, and lists. For example, a technical report would have it's tables and figures interspersed throughout the document, whereas the corresponding journal manuscript submission might have tables and figures collected, one per page, at the end of the document. Whereas the report would have the List of Figures (*lof*) as part of the report's front matter, the journal manuscript might request to have the *lof* preceding the figures located at the end

Table 3. Conditionally compiled LaTeX code.

```
\def\PREPARETYPE{\TECHRPT}% choices:  \TECHRPT or \MANUSCRIPT
\newcommand\TECHRPT{% class for ARL organizational tech rpts
% CONDITIONALLY COMPILED CODE FOR TECHRPT DOCUMENTS
 \documentclass{arlticle}
}
\newcommand\MANUSCRIPT{% article-ized version of tech rpt.
% CONDITIONALLY COMPILED CODE FOR MANUSCRIPTS
 \documentclass[11pt]{article}
 \usepackage{TR2article}
% VARIOUS \MANUSCRIPT-SPECIFIC SETUP COMMANDS GO HERE
}
\PREPARETYPE
% LaTeX CODE & DOCUMENT COMMON TO BOTH STYLES FOLLOWS HEREAFTER
```

of the manuscript. A typical scientific journal manuscript submission would not include a Table of Contents (*toc*), and perhaps not a List of Tables (*lot*) either.

Arranging for such options to unfold from a single input source file is cumbersome without a special treatment. The boxhandler style provides routines to expedite and ease this cumbersome process.

The boxhandler style also conveniently allows for another useful scenario, in which figure and table data may be kept in a separate file from the document text, but nonetheless be made to print out with figures and tables appearing in the proper location in the text. The `\nextTable` and `\nextFigure` commands aid in this approach.

4.2 Deferral and Preemption Commands

For figure and table deferral, and/or list deferral/preemption, the following commands are available, all without arguments:

```
\holdTables
\holdFigures
\clearTables
\clearFigures
\killlistoftables
\killlistoffigures
\killtableofcontents
\holdlistoftables
\holdlistoffigures
\clearlistoftables
```

```
\clearlistoffigures
\nextTable[loc]
\nextFigure[loc]
```

\holdTables
\holdFigures

\holdTables and \holdFigures are used to direct that any subsequent invocations of \bxtable and \bxfigure merely store, rather than store & print the table or figure. These \hold... commands would typically be found in the conditionally compiled code for manuscripts, for example.

While the calculated width of the caption is saved at the time of call to \bxtable and \bxfigure based upon the boxhandler parameters at the time of the \bx... call, the caption format (i.e., [no]offset, center/left alignment, caption justification) is not set until the figure/table is later printed as part of a \clear... command (the logic here is that the caption style will be consistent through the course of the document, thus alleviating the need to store this data for each figure and table).

The one potential pitfall with the use of these commands is the situation when a citation is made within the caption of a deferred table or figure. In this case, the citation is numbered based on when the table or figure is finally displayed, not when it was created. Thus, if \holdTables and/or \holdFigures is used to prevent the printing of tables and/or figures until the end of the document, then any unique citation defined in a table or figure caption will receive a higher citation number than even a citation appearing in the last line of document text.

There is, however, an easy solution to this predicament. That is, when a figure or table is employed which contains a citation, the main text of the document should invoke \nocite{*key_list*} to the same reference. This occurance of \nocite should be placed in the main document (*i.e.*, outside of bxtable or bxfigure) at the point where the table or figure is actually being invoked. In this manner, the citation gets added to the *.aux* file at the proper spot, regardless of when the table or figure is actually printed out.

\clearTables
\clearFigures

\clearTables and \clearFigures directs that any tables or figures that have been stored, but not yet printed, be output at this point. As mentioned above, the offset, alignment, and justification for captions is set at the time of printing, not at the time of table or figure creation. The format for presenting tables and figures to be cleared (i.e., one table/figure per page, vertically centered) can be changed by renewing the commands \theClearedTable and/or \theClearedFigure. See the next section on Advanced Use for details.

Note that if tables and figures have not been subject to a \hold... command, the \clear... commands have no effect, since they affect only those tables and/or figures which have not already been printed. Thus, these commands would typically appear in the common document text at the appropriate point where tables and figures, if previously held, should be printed. Note that, though commands have not been explicitly provided to kill the printing of figures and tables, this can be effectively accomplished by putting figures and/or tables on hold, and then ending the document without having issued a corresponding \clear... command.

\killlistoftables
\killlistoffigures
\killtableofcontents

\killlistoftables, \killlistoffigures and \killtableofcontents direct that any subsequent calls to print the respective list be ignored and that the list be discarded. This action cannot later be undone with a \hold... command. These commands would typically appear in the conditionally compiled region of the document.

\holdlistoftables
\holdlistoffigures

\holdlistoftables and \holdlistoffigures direct that calls to print the particular list cited be deferred until a later invocation of the corresponding \clear... command. Note that a call to \clearTables and \clearFigures will also clear the corresponding list first, if it has been subject to a \hold... command (but not a \kill... command). These \hold... commands would typically appear in the conditionally compiled region of the document.

\clearlistoftables
\clearlistoffigures

\clearlistoftables and \clearlistoffigures clear the cited list (i.e., those lists currently "on hold"). No list will be printed if: the \listoftables or \listoffigures command hadn't earlier been invoked; if it had already been printed out because it hadn't been subject to a hold; or if the list had previously been killed. Note: calls to \clearTables or \clearFigures automatically causes a call to \clearlistoftables or \clearlistoffigures, respectively. Therefore, these particular calls are only needed explicitly in your LaTeX document if it is desired to clear the list well in advance of the associated tables or figures.

With the deferral and preemption commands now described, we show how they might be used to complete the multi-mode LaTeX document stencil that was first laid out in Table 3. To see this, refer to Table 4. The document is created with \bxtable and \bxfigure calls dispersed throughout text, and nominally asks for the *toc*, *lof*, and *lot* to be printed at the beginning of the document. When \PREPARETYPE is defined as {\TECHRPT}, this is exactly how the document unfolds.

However, by merely defining \PREPARETYPE as {\MANUSCRIPT}, the *toc* and *lot* will be killed, preempting subsequent invocations. Printing of the *lof* will be deferred. As bxtables and bxfigures are invoked, they will be created and stored, but not printed. At the very end of the document, all the tables will first be cleared, one per page, vertically centered. The call to \clearFigures will then clear the *lof* on a new page, and finally clear all the figures in a similar manner.

With the change of a single line of code, a vastly different document layout has been achieved!

\nextTable
\nextFigure

The deferral commands described to this point have been cast in terms of tools to aid in printing out multiple vastly different versions of a document from a single source file. With boxhandler v1.10, the commands \nextTable and \nextFigure have been introduced. These commands, while somewhat similar in function to \clearTables and \clearFigures, are useful in a completely different regard. In the case of \nextTable and \nextFigure, only a single table or figure is cleared. Unlike \clearTables and \clearFigures, the table or figure is not printed on a page by itself, but inline the document. The optional argument specifies the location on the page (ht, hb, t, b or p) where the table or figure should appear, consistent with LaTeX convention.

Table 4. Conditionally compiled LaTeX code with `boxhandler` package deferral and preemption directives.

```
\def\PREPARETYPE{\TECHRPT}% choices:  \TECHRPT or \MANUSCRIPT
\newcommand\TECHRPT{% class for ARL organizational tech rpts
% CONDITIONALLY COMPILED CODE FOR TECHRPT DOCUMENTS
 \documentclass{arlticle}
 \usepackage{boxhandler}
}
\newcommand\MANUSCRIPT{% article-ized version of tech rpt.
% CONDITIONALLY COMPILED CODE FOR MANUSCRIPTS
 \documentclass[11pt]{article}
 \usepackage{TR2article}
 \usepackage{boxhandler}
 \killtableofcontents
 \killlistoftables
 \holdlistoffigures
 \holdTables
 \holdFigures
% VARIOUS \MANUSCRIPT-SPECIFIC SETUP COMMANDS GO HERE
}
\PREPARETYPE
% LaTeX CODE & DOCUMENT COMMON TO BOTH STYLES FOLLOWS HEREAFTER
\begin{document}
\maketitle
\tableofcontents
\listoffigures
\listoftables
...
% DOCUMENT CONTAINING \bxtable AND \bxfigure CALLS GOES HERE.
...
\clearTables
\clearFigures
\end{document}
```

The unique utility of the `\nextTable` and `\nextFigure` commands is in allowing one to define all the document's figures and tables "up front," at the beginning of the document, perhaps even in a seperate file that is `\input` into the document. Then, when a table or figure is referred to in the text, all that need be included in the main document is an occurance of `\nextTable` or `\nextFigure`.

In this manner, all the tables and figures can be printed out in the document at their appropriate location, while logically, the document can be organized during preparation to keep the figure and table matter separate from the text matter of the document.

If one wishes to incorporate conditional compiliation (as in Table 4) while simultaneously keeping table and figure definition in a separate file `\input` at the beginning of the document, this can be done, too. The tables and figures would be added after the `\begin{document}` command and invocation would be accomplished by way of `\nextTable` and `\nextFigure` as described here. However, in the `\MANUSCRIPT` preamble (using the nomenclature of Table 4), `\nextTable` and `\nextFigure` would be nullified as:

```
\renewcommand\nextTable[1][]{}
```

and

```
\renewcommand\nextFigure[1][]{}.
```

In this manner, for the `\MANUSCRIPT` report type, tables and figures would be made to not appear until the final invocations of `\clearTables` and `\clearFigures`.

5 Advanced Use (DANGER!)

Additionally, there are certain lower level, yet accessible, counters, macros and lengths, which are intended for advanced use only. It is possible to get into difficulty if not thinking through their use thoroughly. Pitfalls will be laid out, where known. These accessible hooks into the inner workings of the boxhandler package include:

Lengths:
`\DeadMargin, \CaptionBoxWidth`
Counters:
`TableIndex, FigureIndex, TableClearedIndex, FigureClearedIndex, promptTablesFlag, promptFiguresFlag`
Macros:
`\StoreTable{`*caption*`}{`*boxed object*`}{`*wrapper*`}{`*wrapper status*`}`
`\StoreFigure{`*caption*`}{`*boxed object*`}{`*wrapper*`}{`*wrapper status*`}`
`\SaveCBox{`*new cmd*`}{`*boxed object*`}`
`\ReciteTable[`*loc*`]{`*caption*`}{`*cmd*`}{`*wdth*`}{`*wrapper*`}{`*wrapper status*`}`
`\ReciteFigure[`*loc*`]{`*caption*`}{`*cmd*`}{`*wdth*`}{`*wrapper*`}{`*wrapper status*`}`
`\theClearedTable[`*loc*`]{`*caption*`}{`*cmd*`}{`*wdth*`}{`*wrapper*`}{`*wrapper status*`}`
`\theClearedFigure[`*loc*`]{`*caption*`}{`*cmd*`}{`*wdth*`}{`*wrapper*`}{`*wrapper status*`}`

`TableIndex` `FigureIndex` `TableClearedIndex` `FigureClearedIndex`

To keep track of tables and figures as they are created, the counters `TableIndex` and `FigureIndex` are used. To keep track of how many tables and figures have already been printed, `TableClearedIndex`, and `FigureClearedIndex` are used. The appropriate index is incremented whenever a table or figure is created or cleared. These index counters are also used as part of the naming convention employed by the `\StoreTable` and `\StoreFigure` commands, to be subsequently described.

`promptTablesFlag` `promptFiguresFlag`

The counters `promptTablesFlag` and `promptFiguresFlag` are used as binary switches to determine whether calls to `\bxtable` and `\bxfigure` result in the

prompt display (1) or deferred display (0) of the table or figure. The `\holdTables` and `\holdFigures` commands change these switches to their '0' state. Any significant use of these switches to achieve the printing of latter tables/figures prior to the clearing of earlier ones will require the rewrite, by the user, of the `\clearTables` and `\clearFigures` commands, since these `\clear...` macros were written using a first-in-first-out (FIFO) methodology.

`\StoreTable` `\StoreFigure`

The macros `\StoreTable` and `\StoreFigure` use the same form of caption and data arguments as `\bxtable` and `\bxfigure`, with two additional wrapper variables added. In fact, the `\bx...` commands call upon the `\Store...` macros. The difference is that the `\StoreTable` and `\StoreFigure` macros will save the boxed object without printing it, regardless of whether a `\hold...` command has been issued. The saved information will consist of five pointers necessary to recreate the table or figure. These pointers will be named according to boxhandler's internal naming convention.

At this point, it is worth noting the naming convention of the pointers used by boxhandler to store the variables for figures and tables. The saved information resulting from a `\StoreTable` or `\StoreFigure` command will consist of a pointer to a saved box, a pointer to the caption text, a pointer to the calculated width of the caption box (based on the state of the boxhandler parameters at the time of the function call), and pointers to the wrapper and a flag indicating whether wrappers are active for this figure or table.

The counters `TableIndex` and `FigureIndex` are used to create a unique part of these pointer names, in the form of `\roman{`*indexname*`}`. Saved box pointers have the prefix `tbl-` or `fig-`, saved caption pointers have the prefix `tblcap-` or `figcap-` and caption-width pointers have the prefix `tblcapwdth-` and `figcapwdth-`. The wrapper pointer has the prefix `tblwrap-` and the pointer to flag indicating the status of wrapper activity is `tblwrapstatus-`. Thus, for example, the fourth invocation of `\bxtable` or `\StoreTable` will create an sbox named `\tbliv` that is used to store the boxed (e.g., tabular) data, a pointer `\tblcapiv` that is used to store the caption text, a length pointer `\tblcapwdthiv` that stores the value of the calculated caption width, a wrapper pointer `\tblwrapiv`, which stores the wrapper, and a pointer `\tblwrapstatusiv`, which stores a 'T' or 'F' to indicate whether wrappers are currently active or not. When avoiding creative programming, the pointer index (e.g., 'iv' in this example) will correspond to the actual Table or Figure number (i.e., '4') appearing in the caption ID label. However, it is wise to remember,

1. when creating tables or figures outside of the boxhandler style;
2. when bypassing the `\Store...` commands and going straight to the lower level `\SaveCBox` command; or
3. when using the `\Recite...` commands to print multiple occurances of a box;

that the internal numbering of boxhandler tables and figures will likely be out of syncronization with the Table and Figure counters.

\SaveCBox

The low-level routine which saves a "captioned box" is called \SaveCBox. As its arguments, it follows the form of the LaTeX \sbox command: it takes a new command name and the boxed object as its parameters. This is the routine called by the \Store... commands with a command argument something like \tbliv or \figiii, for example. You are, however, free to pass your own command names to this routine, so as not to conflict with the names autogenerated by the \Store... commands. Keep in mind that calls to \SaveCBox will not, by themselves, increment TableIndex or FigureIndex, and will not be tracked by the \clear... commands.

\DeadMargin
\CaptionBoxWidth

In addition to saving the boxed object in the specified command bin, the width of the associated caption is calculated, based upon the prevailing boxhandler parameters at the time of call. One such parameter that is used to calculate the caption width, is the dead margin. \SaveCBox, being a low-level routine, is used for both tables and figures. As such, the appropriate variable to set, in order to define the dead margin is the generic length \DeadMargin, regardless of whether the call is to save a figure or a table box. The calculated caption width is stored in the length variable \CaptionBoxWidth. This length variable is ephemeral, being updated with each new figure or table generated, and so it is up to the user to save the value of \CaptionBoxWidth somewhere else if it is to be used to later define a recited box's caption width. Likewise, while \SaveCBox does not even deal with the box's caption text per se, it is the user's responsibility to save the actual caption somewhere, for later recitation.

\ReciteTable
\ReciteFigure

Reciting stored tables or figures is accomplished by way of \ReciteTable and \ReciteFigure. As the [optional] first of their four arguments, they take the location directive for placement on the page (e.g., ht, hb, t, b, or p). Both the caption and the caption width arguments may be specified directly, or indirectly by way of a stored string and length variable, respectively. The command argument to these macros must be the bin for the saved box object that constitutes the actual table or figure data.

If one desires, the \Recite... commands may be used repeatedly to print a given table or figure multiple times. However, any \label associated with the [repeatedly recited] caption will be reassigned to the most recent invocation of the \Recite... call. Thus, references to the table or figure number by way of a \ref call are likely to produce an undesired result, if a given table or figure is recited multiple times.

\theClearedTable
\theClearedFigure

Finally, two very useful routines to be familiar with are the \theClearedTable and \theClearedFigure macros. These are the routines which are repeatedly called by \clearTables and \clearFigures, respectively, to print out all tables and figures which have been "on hold." The manner in which boxhandler clears them is one per page, vertically centered. It is easy to envision applications in which the method of clearing would take on a different appearance than this. To change the appearance by which tables and/or figures are cleared, these commands

need to be redefined by the user by way of a \renewcommand. For reference, the \theClearedTable command is defined by boxhandler as:

```
\newcommand\theClearedTable[6][ht]{
  \vspace*{\fill}
  \ReciteTable[#1]{#2}{#3}{#4}{#5}{#6}
  \vspace*{\fill}
  \clearpage
}
```

As one can see, \theClearedTable is a call to \ReciteTable that is surrounded by the code necessary to place the table recitation at the desired location upon the page. \theClearedFigure is defined analogously. Modification of this layout should be straightforward for the user. For example, if it were desired to have two tables per page during the clearing process, one might use the following renewal:

```
\newcounter{toggle} \setcounter{toggle}{0}
\renewcommand\theClearedTable[6][ht]{
  \addtocounter{toggle}{1}
  \ifnum \arabic{toggle} = 2 \setcounter{toggle}{0} \fi
  \ifnum \arabic{toggle} = 1
    \ReciteTable[t]{#2}{#3}{#4}{#5}{#6}
  \else
    \ReciteTable[b]{#2}{#3}{#4}{#5}{#6}
    \clearpage
  \fi
}
```

6 Examples

Examples of a number of calls provided in this style are given below, in no particular order:

```
\bxtable[t]
  {This is a test of nonwrapping caption\label{tb:mytable}}
  {
  \begin{tabular}{|c|c|c|}
    \hline
    column 1 data & 2nd column data & and the third column data\\
    \hline
  \end{tabular}
  }
```

```
\usepackage{graphicx}
\bxfigure[ht]
  {Here is an example of a particularly long caption that will
          test wrapping}
  {\includegraphics[width=3.64in,height=4.30in]{Atest1.eps}}

\relaxCaptionWidth[2in]

\captionStyle{}{c}

\setlength\captionGap{3ex}

\setcounter{abovecaptionskipterm}{10}
\setcounter{belowcaptionskipterm}{0}

\TableFontSize\normalsize

\LRFigurePlacement{flushleft}

\killtableofcontents

\holdFigures

\clearFigures

\newsavebox{\mybox}
\SaveCBox{\mybox}{\framebox(200,100){Ack!}}

\ReciteFigure[ht]{\figcapii}{\figii}{\figcapwdthii}%
                 {\figwrapii}{\figwrapstatusii}
```

7 Vestigials

`\arltable` `\arlfigure`

In order to retain backward compatibility with the predecessor to the boxhandler package, the vestigial commands `\arltable` and `\arlfigure` are defined in this package, and are equivalent to `\bxtable` and `\bxfigure`, respectively.

As an historical note, the genesis of the boxhandler package was a requirement to comply with my organization's caption style ("offset" style, "left" aligned) specified for its technical reports. Various piecemeal approaches were out there to handle it, requiring case-specific decisions on the part of the author, depending on the specifics of the figure/table box and the caption. boxhandler was intended to simplify and generalize the approach.

The figure/table deferral features of boxhandler were borne of my own laziness. I just didn't relish keeping two different versions of the same document up to date... one for my organization, and the other as a manuscript for possible journal submission. The `\bx...` commands provided an easier "hook" through which to achieve the holding and clearing of boxes than did any attempt to muck with the underlying `Figure` and `Table` environments of LaTeX.

8 Acknowledgments/Salutations

That's it for a description of boxhandler! I would like to thank a number of colleagues for their consultive assistance. Several were instrumental in getting me over early bumps in my LaTeX learning curve, including Mike Scheidler, Stephen Schraml, Brian Krzewinski, and Paul Tanenbaum, all of the USARL. I am particularly grateful to Fred Brundick (also of ARL) for sharing his extensive knowledge of LaTeX with me. He clued me in to the sticky quirks of offset captions, and shared his methods for dealing with the different cases. He also pointed me in the right direction for the systematic saving of LaTeX box objects, which contributed to the section of this package dealing with object deferral.

I hope this package provides you some utility. The only thing left is the code listing itself.

boxhandler.sty

9 Code Listing

I'll try to lay out herein the workings of the boxhandler style package. I apologize if the code fails in some way to conform to LaTeX programming conventions. I am but an enthusiastic novice.

```
1 ⟨*package⟩
```

pbox

This package makes use of the ifthen and pbox style packages to aid in the boxing of captions.

```
2 \RequirePackage{ifthen}
3 \usepackage{pbox}
```

TableIndex
FigureIndex
TableClearedIndex
FigureClearedIndex

We start by defining and initializing the counters that keep track of how many tables and figures have been created and how many have been cleared (i.e., printed out).

```
4 \newcounter{TableIndex}          \setcounter{TableIndex}{0}
5 \newcounter{FigureIndex}         \setcounter{FigureIndex}{0}
6 \newcounter{TableClearedIndex}   \setcounter{TableClearedIndex}{0}
7 \newcounter{FigureClearedIndex}  \setcounter{FigureClearedIndex}{0}
```

\old@makecaption
\oldabovecaptionskip
\oldbelowcaptionskip

Here, we save the prevailing definitions of \@makecaption, \abovecaptionskip and \belowcaptionskip, so that they can be altered before and restored after every invocation of \bxtable and \bxfigure.

```
8 \let\old@makecaption \@makecaption% SAVE PREVAILING \@makecaption STYLE
9 \newlength\oldabovecaptionskip
10 \setlength\oldabovecaptionskip \abovecaptionskip
11 \newlength\oldbelowcaptionskip
12 \setlength\oldbelowcaptionskip \belowcaptionskip
```

\captionGap

Initialize \captionGap to 1ex, which sets the horizontal space between the caption label and the caption box for offset-style captions.

```
13 %% \captionGap CAN BE INCREASED TO PLACE MORE SPACE BETWEEN THE CAPTION
14 %% LABEL AND THE ACTUAL CAPTION TEXT.
15 \newlength\captionGap          \setlength\captionGap{1ex}
```

\TableDeadMargin
\FigureDeadMargin

The default values for the dead margin around tables and figures is initialized here. The value of 0.375em for tables corresponds to what the tabular environment seems to produce. No presupposition is made on what the dead margin is for the figure environment, however.

```
16 %% \TableDeadMargin AND \FigureDeadMargin REFER TO THE TABLE & FIGURE
17 %% MARGIN OF A BOX THAT COUNTS TOWARDS ITS SIZE, BUT IN WHICH NO
18 %% ACTIVE DATA FALLS; DEADMARGIN ASSUMED ON BOTH LEFT & RIGHT SIDE.
19 \newlength\TableDeadMargin     \setlength\TableDeadMargin{0.375em}
20 \newlength\FigureDeadMargin    \setlength\FigureDeadMargin{0em}
```

abovecaptionskipterm
belowcaptionskipterm

The integer parameters abovecaptionskipterm and belowcaptionskipterm are initialized here. They will be used to guide the temporarily alteration of \abovecaptionskip and \belowcaptionskip during invocations of \bxtable and \bxfigure.

```
%% INITIALIZE \above- AND \belowcaptionskip VALUES; CAN BE RESET
%% USED TO SET GAPS ABOVE & BELOW CAPTIONS
\newcounter{abovecaptionskipterm}  \setcounter{abovecaptionskipterm}{10}
\newcounter{belowcaptionskipterm}  \setcounter{belowcaptionskipterm}{7}
```

\@minCaptionBoxWidth
\@minCaptionBoxWidthDefault
\@maxCaptionBoxWidth
\@maxCaptionBoxWidthDefault

The minimum and maximum allowed width defaults for the caption box are defined here. The variables holding the current values of min/max caption width are also allocated here. Their values will be set and altered through the use of the macros \relaxCaptionWidth, \limitCaptionWidth and \constrainCaptionWidth, to be described later.

```
%% RESET \@minCaptionBoxWidth TO Default VALUE WITH \relaxCaptionWidth
%% SET \@minCaptionBoxWidth TO USER VALUE WITH \relaxCaptionWidth[<len>]
\newlength\@minCaptionBoxWidth
\newlength\@minCaptionBoxWidthDefault
\setlength\@minCaptionBoxWidthDefault{1in}
%% RESET \@maxCaptionBoxWidth TO Default WITH \limitCaptionWidth
%% SET \@maxCaptionBoxWidth WITH \limitCaptionWidth[<len>]
\newlength\@maxCaptionBoxWidth
\newlength\@maxCaptionBoxWidthDefault
\setlength\@maxCaptionBoxWidthDefault{\textwidth}
```

promptTablesFlag
promptFiguresFlag
\holdTables
\holdFigures

promptTablesFlag and promptFiguresFlag are binary switches to define whether calls to \bxtable and \bxfigure are cleared (i.e., printed) promptly or merely stored for later clearing. The change to put them "on hold" is actuated by the \holdTables and \holdFigures commands, respectively. Because the associated \clear... commands were written with FIFO logic, no mechanism is provided to reset the flags to "prompt", once set to "on hold." Even so, tables and figures can be cleared in sub-batches, by issuing a series of \clear... commands at intermediate points in the document.

```
%% DEFINE promptTablesFlag & PROVIDE ROUTINE TO CHANGE IT FROM
%% "PROMPT"(1) TO "ON HOLD"(0)
\newcounter{promptTablesFlag}    \setcounter{promptTablesFlag}{1}
\newcommand\holdTables{\setcounter{promptTablesFlag}{0}}
%% SAME FOR promptFiguresFlag
\newcounter{promptFiguresFlag}   \setcounter{promptFiguresFlag}{1}
\newcommand\holdFigures{\setcounter{promptFiguresFlag}{0}}
```

\CaptionFontSize
\TableFontSize

Default size of font in both captions and tables is, by default, \small. However, these variables allow those defaults to be reset to the desired font size.

```
%% DEFAULT CaptionFontSize & TableFontSize AS \small; CAN BE RESET
\let \CaptionFontSize \small
\let \TableFontSize \small
```

\LRTablePlacement
\LRFigurePlacement

These variables set the placement of the table or figure either flushed to the left or right margin, or else centered between the margins (the default). They can be reset by the user.

```
%% DEFAULT TABLE & FIGURE LR ALIGNMENT TO center;
%% CAN RESET TO flushleft/-right
\def \LRTablePlacement {center}
\def \LRFigurePlacement {center}
```

\CaptionJustification

The flushing of the actual text within the caption box may be accomplished by setting \CaptionJustification. When it is defined as {} (the default), the caption is fully justified (i.e., flushed) within the caption box. It may also be set to {\raggedleft}, {\raggedright} or {\centering}.

```
%% WITHIN-CAPTION JUSTIFICATION CAN BE SET
%% OPTIONS: {}, {\raggedleft}, {\raggedright}, or {\centering}
\def\CaptionJustification{} % <---DEFAULT IS FULL JUSTIFICATION

```

\DeadMargin
\CaptionBoxWidth
\@DataBoxWidth
\@DataBoxOffset
\@CaptionBoxOffset
\@captionIDwidth
\@captionWidth
\@DataBoxSurplus

New working variables are defined here. All are @-protected from access except for \DeadMargin and \CaptionBoxWidth, which have been made accessible for so-called "Advanced Use." \DeadMargin is the low-level variable where the dead margin of the current figure or table box is specified by the advanced user. \@DataBoxWidth is the calculated width of the data-box portion of the current table or figure. \CaptionBoxWidth is the calculated width of the caption box for the current figure, taking into account the dead margin as well as the min/max allowed caption widths. The "advanced user" can save this datum for future figure/table recitation. \@DataBoxOffset is the calculated spacer length that must be added to both sides of the data box to bring it to the size of the caption box. It will be zeroed, if the data box width exceeds the caption box width. \@CaptionBoxOffset is the calculated spacer length that must be added to both sides of the caption box to bring it to the size of the data box. It will be zeroed, if the caption box width exceeds the data box width. \@captionIDwidth is used for offset-style captions, and is the width of the caption ID plus the caption gap (e.g., the width of "Figure 4. "). \@captionWidth is, for offset-style captions, simply \CaptionBoxWidth minus \@captionIDwidth; this value equals the width of the \parbox which is used for the caption text in offset-style captions. Finally, \@DataBoxSurplus is the excess of data box width over the caption box width. It can be positive or negative, depending on whether the data box or caption box is larger.

```
%% WORKING VARIABLES
\newlength\DeadMargin
\newlength\@DataBoxWidth
\newlength\CaptionBoxWidth
\newlength\@DataBoxOffset
\newlength\@CaptionBoxOffset
\newlength\@captionIDwidth
```

```
\newlength\@captionWidth
\newlength\@DataBoxSurplus

```

\WrapperOn
\WrapperOff
\Wrapper
\WrapperTextStyle

Wrappers are identifying text (or icons) that bound figures and tables in the upper-left and lower-right corners. Initially disabled, `\WrapperOn` turns wrappers on. The optional argument of `\WrapperOn`, which should be used on the first invocation, specifies the default wrapper. Wrappers can be turned off with `\WrapperOff`. The default wrapper can be changed with a subsequent invocation to `\WrapperOn`.

However, the wrapper for any given figure or table may be individually specified (without changing the default wrapper) by way of the argument to `\Wrapper`. Both the default wrapper as well as one passed as an argument to `\Wrapper` are presented with the style given in `\WrapperTextStyle`, which defaults to small, bold font, `\bf\small`. The command `\Wrapper`, if used, should be placed within the second mandatory argument of `\bxtable` and `\bxfigure`, if a wrapper other than the default is desired.

```
%% FIGURE & TABLE WRAPPER INITIALIZATIONS
\def\wrapper{F}
\def\WrapperTextStyle{\bf\small}
\def\WrapperTextDefault{DEFAULT WRAPPER}
\global\def\WrapperText{\noexpand\WrapperTextStyle\WrapperTextDefault}
\newcommand\WrapperOn[1][]{%
  \def\wrapper{T}%
  \ifthenelse{\equal{#1}{}}%
  {}{\def\WrapperTextDefault{\noexpand#1}}%
  \global\def%
     \WrapperText{\noexpand\WrapperTextStyle\WrapperTextDefault}%
}
\newcommand\WrapperOff{\def\wrapper{F}}
\newcommand\Wrapper[1]{\global\def%
  \WrapperText{\noexpand\WrapperTextStyle\noexpand#1}}

```

\bxtable

The routine `\bxtable` will store the specified table. If `promptTablesFlag` equals unity, the table will also be immediately cleared using the specified *loc* parameter. If `\roman{TableIndex}` equalled, for example, `vii`, then the table data would be stored in the box `\tblvii`, the caption text would be stored in the pointer `\tblcapvii`, and the calculated caption width would be stored in the pointer `\tblcapwdthvii`.

```
\newcommand\bxtable[3][t]{%
  \StoreTable[#1]{#2}{#3}{\WrapperText}{\wrapper}%
  \ifnum\arabic{promptTablesFlag}=1%
    \addtocounter{TableClearedIndex}{1}%
    \def\TableBoxLabel{tbl\roman{TableIndex}}%
    \def\TableCaptionLabel{tblcap\roman{TableIndex}}%
```

```
    \def\TblCaptionWidthLabel{tblcapwdth\roman{TableIndex}}%
    \def\TableWrapper{tblwrap\roman{TableIndex}}%
    \def\WrapperStatus{tblwrapstatus\roman{TableIndex}}%
    \ReciteTable[#1]{\csname\TableCaptionLabel\endcsname}%
                    {\csname\TableBoxLabel\endcsname}%
                    {\csname\TblCaptionWidthLabel\endcsname}%
                    {\csname\TableWrapper\endcsname}%
                    {\csname\WrapperStatus\endcsname}%
  \fi
}

```

`\StoreTable` `\StoreTable` calculates the names of the pointers where the table-data, -caption, and -width will be stored, and calls upon the low-level `\SaveCbox` routine to actually save the tabular data and compute the caption width. It finally stores the caption text itself and the calculated caption width. Note that the optional first argument, *loc*, is a dummy argument that is not used here.

```
\newcommand\StoreTable[5][]{%
  \addtocounter{TableIndex}{1}%
  \setlength\DeadMargin\TableDeadMargin%
  \def\TableBoxLabel{tbl\roman{TableIndex}}%
  \def\TableCaptionLabel{tblcap\roman{TableIndex}}%
  \def\TblCaptionWidthLabel{tblcapwdth\roman{TableIndex}}%
  \def\TableWrapper{tblwrap\roman{TableIndex}}%
  \def\WrapperStatus{tblwrapstatus\roman{TableIndex}}%
  \expandafter\SaveCBox\csname\TableBoxLabel\endcsname{\TableFontSize#3}%
  \expandafter\def\csname\TableCaptionLabel\endcsname{#2}%
  \expandafter\newlength\csname\TblCaptionWidthLabel\endcsname%
  \expandafter\setlength\csname\TblCaptionWidthLabel\endcsname%
                                                \CaptionBoxWidth%
  \expandafter\edef\csname\TableWrapper\endcsname{#4}%
  \expandafter\edef\csname\WrapperStatus\endcsname{#5}%
%% After storing table, reset wrapper to default value
  \global\def%
   \WrapperText{\noexpand\WrapperTextStyle\WrapperTextDefault}%
}

```

`\ReciteTable` The `\ReciteTable` routine recites a previously stored table, using the pointers that are provided as arguments. First, the `\Table` environment is invoked, and the left/right table-placement environment is opened. The routine then defines new definitions for `\@makecaption`, `\abovecaptionskip` and `\belowcaptionskip`. It then uses the provided pointers to set the data-box and caption-box widths, and calculates the offsets. It recites the box caption (using the caption-box offset, if needed) and then it recites the tabular-data box (using the data-box offset, if needed). The original definitions for `\@makecaption`, `\abovecaptionskip` and `\belowcaptionskip` are restored, the table placement environment is concluded

and the table itself is ended.

```
\newcommand\ReciteTable[6][ht]{%
  \begin{table}[#1]%
    \begin{\LRTablePlacement}%
      \let\@makecaption\new@makecaption%
      \setlength\abovecaptionskip{\arabic{abovecaptionskipterm}\p@}%
      \setlength\belowcaptionskip{\arabic{belowcaptionskipterm}\p@}%
        \set@DataBoxWidth{#3}%
        \setlength\CaptionBoxWidth{#4}%
        \set@BoxOffsets%
        \if T#6%
          \rule{\@DataBoxOffset}{0in}%
          \makebox[\@DataBoxWidth][l]{#5}%
          \rule{\@DataBoxOffset}{0in}\\%
        \fi
        \rule{\@CaptionBoxOffset}{0em}%
        \parbox{\CaptionBoxWidth}{\bx@caption{#2}}%
        \rule{\@CaptionBoxOffset}{0em}%
        \par%
        \rule{\@DataBoxOffset}{0in}%
        \usebox{#3}%
        \rule{\@DataBoxOffset}{0in}\\%
        \if T#6%
          \rule{\@DataBoxOffset}{0in}\\%
          \makebox[\@DataBoxWidth][r]{#5}%
        \rule{\@DataBoxOffset}{0in}%
        \fi
      \let\@makecaption\old@makecaption%
      \setlength\abovecaptionskip \oldabovecaptionskip%
      \setlength\belowcaptionskip \oldbelowcaptionskip%
    \end{\LRTablePlacement}%
  \end{table}%
}

```

`\nextTable` This routine will clear a single table, if there are any that have not yet been printed as a result of a previously invoked command. It assumes FIFO logic. The optional argument is the location on the page where the table is to be printed, in accordance with standard LaTeX logic. If there are no uncleared tables left to format, then the command has no effect.

```
\newcommand\nextTable[1][ht]{%
  \ifnum\arabic{TableClearedIndex}<\arabic{TableIndex}{%
    \addtocounter{TableClearedIndex}{1}%
%%  \TableBoxLabel       : tbli,    tblii,    tbliii,    tbliv,    etc.
%%  \TableCaptionLabel   : tblcapi, tblcapii, tblcapiii, tblcapiv, etc.
%%  \TblCaptionWidthLabel: tblcapwdthi, tblcapwdthii, tblcapwdthiii,etc.
    \def\TableBoxLabel{tbl\roman{TableClearedIndex}}%
    \def\TableCaptionLabel{tblcap\roman{TableClearedIndex}}%
```

```
    \def\TblCaptionWidthLabel{tblcapwdth\roman{TableClearedIndex}}%
    \def\TableWrapper{tblwrap\roman{TableClearedIndex}}%
    \def\WrapperStatus{tblwrapstatus\roman{TableClearedIndex}}%
    \ReciteTable[#1]{\csname\TableCaptionLabel\endcsname}%
                    {\csname\TableBoxLabel\endcsname}%
                    {\csname\TblCaptionWidthLabel\endcsname}%
                    {\csname\TableWrapper\endcsname}%
                    {\csname\WrapperStatus\endcsname}%
  }\fi
}

```

`\clearTables` This routine will clear all stored tables that have not yet been printed. It assumes a FIFO logic. It starts by clearing the page and clearing the List of Tables (i.e., prints the *lot* **if** the List of Tables had been put on hold, and subsequently invoked while "on hold"). If the *lot* had previously been killed, then `\clearlistoftables` will have no effect. Once the *lot* has been cleared (or not), a loop is set up in which the names of stored-table pointers are reconstructed, and successively passed to `\theClearedTable` which defines the format for clearing and actually calls for the table to be recited.

```
\newcommand\clearTables{%
  \clearpage%
  \clearlistoftables%
  \clearpage%
%%DO UNTIL ALL HELD TABLES ARE CLEARED
  \whiledo{\arabic{TableClearedIndex}<\arabic{TableIndex}}{%
    \addtocounter{TableClearedIndex}{1}%
%%  \TableBoxLabel       : tbli,    tblii,    tbliii,    tbliv,    etc.
%%  \TableCaptionLabel   : tblcapi, tblcapii, tblcapiii, tblcapiv, etc.
%%  \TblCaptionWidthLabel: tblcapwdthi, tblcapwdthii, tblcapwdthiii,etc.
    \def\TableBoxLabel{tbl\roman{TableClearedIndex}}%
    \def\TableCaptionLabel{tblcap\roman{TableClearedIndex}}%
    \def\TblCaptionWidthLabel{tblcapwdth\roman{TableClearedIndex}}%
    \def\TableWrapper{tblwrap\roman{TableClearedIndex}}%
    \def\WrapperStatus{tblwrapstatus\roman{TableClearedIndex}}%
    \theClearedTable{\csname\TableCaptionLabel\endcsname}%
                    {\csname\TableBoxLabel\endcsname}%
                    {\csname\TblCaptionWidthLabel\endcsname}%
                    {\csname\TableWrapper\endcsname}%
                    {\csname\WrapperStatus\endcsname}%
  }%
}

```

`\theClearedTable` Quite simply, this routine prints out each table to be cleared, one per page, vertically centered. It can be renewed by the user if a different clearing format is desired.

```
%% \theClearedTable CAN BE RENEWED IF DIFFERENT OUTPUT FORMAT IS DESIRED
\newcommand\theClearedTable[6][ht]{%
%%  CLEAR THIS TABLE ON A PAGE BY ITSELF, CENTERED VERTICALLY
  \vspace*{\fill}%
  \ReciteTable[#1]{#2}{#3}{#4}{#5}{#6}%
  \vspace*{\fill}%
  \clearpage%
}

```

\bxfigure This routine is analogous to \bxtable in every way. For figures, the pointers which save the figure use a \fig- prefix, instead of a \tbl- prefix.

```
\newcommand\bxfigure[3][t]{%
  \StoreFigure[#1]{#2}{#3}{\WrapperText}{\wrapper}%
  \ifnum\arabic{promptFiguresFlag}=1%
    \addtocounter{FigureClearedIndex}{1}%
    \def\FigureBoxLabel{fig\roman{FigureIndex}}%
    \def\FigureCaptionLabel{figcap\roman{FigureIndex}}%
    \def\FigCaptionWidthLabel{figcapwdth\roman{FigureIndex}}%
    \def\FigureWrapper{figwrap\roman{FigureIndex}}%
    \def\WrapperStatus{figwrapstatus\roman{FigureIndex}}%
    \ReciteFigure[#1]{\csname\FigureCaptionLabel\endcsname}%
                         {\csname\FigureBoxLabel\endcsname}%
                         {\csname\FigCaptionWidthLabel\endcsname}%
                         {\csname\FigureWrapper\endcsname}%
                         {\csname\WrapperStatus\endcsname}%
  \fi
}

```

\StoreFigure This routine is analogous to \StoreTable in every way.

```
\newcommand\StoreFigure[5][]{%
  \addtocounter{FigureIndex}{1}%
  \setlength\DeadMargin\FigureDeadMargin%
  \def\FigureBoxLabel{fig\roman{FigureIndex}}%
  \def\FigureCaptionLabel{figcap\roman{FigureIndex}}%
  \def\FigCaptionWidthLabel{figcapwdth\roman{FigureIndex}}%
  \def\FigureWrapper{figwrap\roman{FigureIndex}}%
  \def\WrapperStatus{figwrapstatus\roman{FigureIndex}}%
  \expandafter\SaveCBox\csname\FigureBoxLabel\endcsname{#3}%
  \expandafter\def\csname\FigureCaptionLabel\endcsname{#2}%
  \expandafter\newlength\csname\FigCaptionWidthLabel\endcsname%
  \expandafter\setlength\csname\FigCaptionWidthLabel\endcsname%
                                                \CaptionBoxWidth%
  \expandafter\edef\csname\FigureWrapper\endcsname{#4}%
  \expandafter\edef\csname\WrapperStatus\endcsname{#5}%
%% After storing figure, reset wrapper to default value
  \global\def%
```

```
    \WrapperText{\noexpand\WrapperTextStyle\WrapperTextDefault}%
}
```

\ReciteFigure This routine is analogous to \ReciteTable in every way, except one. In the case of \ReciteFigure, the figure-data box is output **before** the caption, not after.

```
\newcommand\ReciteFigure[6][ht]{%
  \begin{figure}[#1]%
    \begin{\LRFigurePlacement}%
      \let\@makecaption\new@makecaption%
      \setlength\abovecaptionskip{\arabic{abovecaptionskipterm}\p@}%
      \setlength\belowcaptionskip{\arabic{belowcaptionskipterm}\p@}%
        \set@DataBoxWidth{#3}%
        \setlength\CaptionBoxWidth{#4}%
        \set@BoxOffsets%
        \if T#6%
          \rule{\@DataBoxOffset}{0in}%
          \makebox[\@DataBoxWidth][l]{#5}%
          \rule{\@DataBoxOffset}{0in}\\%
        \fi
        \rule{\@DataBoxOffset}{0in}%
        \usebox{#3}%
        \rule{\@DataBoxOffset}{0in}%
        \par%
        \rule{\@CaptionBoxOffset}{0em}%
        \parbox{\CaptionBoxWidth}{\bx@caption{#2}}%
        \rule{\@CaptionBoxOffset}{0em}%
        \if T#6%
          \rule{\@DataBoxOffset}{0in}\\%
          \makebox[\@DataBoxWidth][r]{#5}%
          \rule{\@DataBoxOffset}{0in}%
        \fi
      \let\@makecaption\old@makecaption%
      \setlength\abovecaptionskip\oldabovecaptionskip%
      \setlength\belowcaptionskip\oldbelowcaptionskip%
    \end{\LRFigurePlacement}%
  \end{figure}%
}
```

\nextFigure This routine is analogous to \nextTable in every way.

```
\newcommand\nextFigure[1][ht]{%
  \ifnum\arabic{FigureClearedIndex}<\arabic{FigureIndex}{%
    \addtocounter{FigureClearedIndex}{1}%
%%  \FigureBoxLabel:     : figi,    figii,    figiii,    figiv,    etc.
%%  \FigureCaptionLabel  : figcapi, figcapii, figcapiii, figcapiv, etc.
%%  \FigCaptionWidthLabel: figcapwdthi, figcapwdthii, figcapwdthiii,etc.
    \def\FigureBoxLabel{fig\roman{FigureClearedIndex}}%
```

```
    \def\FigureCaptionLabel{figcap\roman{FigureClearedIndex}}%
    \def\FigCaptionWidthLabel{figcapwdth\roman{FigureClearedIndex}}%
    \def\FigureWrapper{figwrap\roman{FigureClearedIndex}}%
    \def\WrapperStatus{figwrapstatus\roman{FigureClearedIndex}}%
    \ReciteFigure[#1]{\csname\FigureCaptionLabel\endcsname}%
                     {\csname\FigureBoxLabel\endcsname}%
                     {\csname\FigCaptionWidthLabel\endcsname}%
                     {\csname\FigureWrapper\endcsname}%
                     {\csname\WrapperStatus\endcsname}%
  }\fi
}

```

`\clearFigures` This routine is analogous to `\clearTables` in every way.

```
\newcommand\clearFigures{%
  \clearpage%
  \clearlistoffigures%
  \clearpage%
%%DO UNTIL ALL HELD FIGURES ARE CLEARED
  \whiledo{\arabic{FigureClearedIndex}<\arabic{FigureIndex}}{%
    \addtocounter{FigureClearedIndex}{1}%
%%  \FigureBoxLabel      : figi,    figii,    figiii,    figiv,    etc.
%%  \FigureCaptionLabel  : figcapi, figcapii, figcapiii, figcapiv, etc.
%%  \FigCaptionWidthLabel: figcapwdthi, figcapwdthii, figcapwdthiii,etc.
    \def\FigureBoxLabel{fig\roman{FigureClearedIndex}}%
    \def\FigureCaptionLabel{figcap\roman{FigureClearedIndex}}%
    \def\FigCaptionWidthLabel{figcapwdth\roman{FigureClearedIndex}}%
    \def\FigureWrapper{figwrap\roman{FigureClearedIndex}}%
    \def\WrapperStatus{figwrapstatus\roman{FigureClearedIndex}}%
    \theClearedFigure{\csname\FigureCaptionLabel\endcsname}%
                     {\csname\FigureBoxLabel\endcsname}%
                     {\csname\FigCaptionWidthLabel\endcsname}%
                     {\csname\FigureWrapper\endcsname}%
                     {\csname\WrapperStatus\endcsname}%
  }%
}

```

`\theClearedFigure` This routine is analogous to `\theClearedTable` in every way... one figure per page, vertically centered.

```
%% \theClearedFigure CAN BE RENEWED IF DIFFERENT OUTPUT FORMAT DESIRED
\newcommand\theClearedFigure[6][ht]{%
%%  CLEAR THIS FIGURE ON A PAGE BY ITSELF, CENTERED VERTICALLY
  \vspace*{\fill}%
  \ReciteFigure[#1]{#2}{#3}{#4}{#5}{#6}%
  \vspace*{\fill}%
  \clearpage%
}
```

\relaxCaptionWidth This routine sets the minimum permitted caption width. When called with no argument, it resets the min caption width to its 1 inch default value. If necessary, the maximum allowed caption width will be bumped up, so as to remain greater than or equal the minimum allowed caption width.

```
\newcommand\relaxCaptionWidth[1][\@minCaptionBoxWidthDefault]{%
  \setlength\@minCaptionBoxWidth{#1}%
  \ifdim \@minCaptionBoxWidth > \@maxCaptionBoxWidth%
    \setlength\@maxCaptionBoxWidth\@minCaptionBoxWidth%
  \fi
}
\relaxCaptionWidth% SET INITIAL \@minCaptionBoxWidth TO DEFAULT VALUE
```

\limitCaptionWidth This routine sets the maximum permitted caption width. When called with no argument, it resets the max caption width to its default value of \textwidth. If necessary, the minimum allowed caption width will be reduced, so as to remain less than or equal the maximum allowed caption width.

```
\newcommand\limitCaptionWidth[1][\@maxCaptionBoxWidthDefault]{%
  \setlength\@maxCaptionBoxWidth{#1}%
  \ifdim \@maxCaptionBoxWidth < \@minCaptionBoxWidth%
    \setlength\@minCaptionBoxWidth\@maxCaptionBoxWidth%
  \fi
}
\limitCaptionWidth% SET INITIAL \@maxCaptionBoxWidth TO DEFAULT VALUE
```

\constrainCaptionWidth Straightforward code to set both min- and max-allowed caption widths. Only twist: if only one argument given, both min- and max-caption widths set to that value.

```
\newcommand\constrainCaptionWidth[2][-1in]{%
  \ifdim #1 < 0in%
    \setlength\@minCaptionBoxWidth{#2}%
    \setlength\@maxCaptionBoxWidth{#2}%
  \else
    \ifdim #1 < #2%
      \setlength\@minCaptionBoxWidth{#1}%
      \setlength\@maxCaptionBoxWidth{#2}%
    \else
      \setlength\@minCaptionBoxWidth{#2}%
      \setlength\@maxCaptionBoxWidth{#1}%
    \fi
  \fi
}
```

`\SaveCBox` Low-level routine to save box data in an `sbox`. Also, calculates data box width and associated caption box width.

```
\newcommand\SaveCBox[2]{%
  \newsavebox{#1}%
  \sbox{#1}{#2}%
  \set@BoxWidths{#1}%
}

```

`\set@BoxWidths` Call successive routines to define `\@DataBoxWidth` and `\CaptionBoxWidth`.

```
\newcommand\set@BoxWidths[1]{% of DataBox & CaptionBox (-2\DeadMargin)%
  \set@DataBoxWidth{#1}%
  \set@CaptionBoxWidth%
}

```

`\set@DataBoxWidth` Calculate and set data-box width.

```
\newcommand\set@DataBoxWidth[1]{%
  \setlength {\@DataBoxWidth}{\widthof{\usebox{#1}}}%
}

```

`\set@CaptionBoxWidth` Calculate and set caption-box width, subject to constraints of dead margin as well as caption-box min/max allowable widths.

```
\newcommand\set@CaptionBoxWidth{%
  \setlength\CaptionBoxWidth\@DataBoxWidth%
  \addtolength{\CaptionBoxWidth}{-2\DeadMargin}%
  \ifdim \CaptionBoxWidth < \@minCaptionBoxWidth%
    \setlength\CaptionBoxWidth\@minCaptionBoxWidth%
  \fi
  \ifdim \CaptionBoxWidth > \@maxCaptionBoxWidth%
    \setlength\CaptionBoxWidth\@maxCaptionBoxWidth%
  \fi
}

```

`\set@BoxOffsets` Calculate `\DataBoxSurplus` which holds the excess width of the data box with respect to the associated caption box. Use it to set `\@DataBoxOffset` and `\@CaptionBoxOffset`.

```
\newcommand\set@BoxOffsets{%
  \setlength\@DataBoxSurplus{\@DataBoxWidth}%
  \addtolength\@DataBoxSurplus{-\CaptionBoxWidth}%
  \ifdim \@DataBoxSurplus > 0in%
    \setlength\@CaptionBoxOffset{0.5\@DataBoxSurplus}%
    \setlength\@DataBoxOffset{0in}%
  \else
    \setlength\@CaptionBoxOffset{0in}%
    \setlength\@DataBoxOffset{-0.5\@DataBoxSurplus}%
  \fi
}

```

`\offset@caption` `\nooffset@caption` Define the code for placing offset- and nooffset-captions in the caption box.

```
\long\def\offset@caption#1#2{%
  \setlength\@captionIDwidth{\widthofpbox{\CaptionFontSize{#1.}}}%
  \addtolength\@captionIDwidth\captionGap%
  \setlength\@captionWidth\CaptionBoxWidth%
  \addtolength\@captionWidth{-\@captionIDwidth}%
  \CaptionFontSize{#1.}\hfill\parbox[t]{\@captionWidth}%
                  {\CaptionJustification\CaptionFontSize{#2.}}%
}

\long\def\nooffset@caption#1#2{%
  \CaptionJustification\CaptionFontSize #1.\rule{\captionGap}{0in}#2.%
}

```

`\shortleft@caption` `\shortcenter@caption` `\shortright@caption` Define the code for placing short-left, -center, and -right captions in the caption box.

```
\long\def\shortleft@caption#1#2{%
  \raggedright\CaptionFontSize #1.\rule{\captionGap}{0in}#2.%
}

\long\def\shortcenter@caption#1#2{%
  \centering\CaptionFontSize #1.\rule{\captionGap}{0in}#2.%
}

\long\def\shortright@caption#1#2{%
  \raggedleft\CaptionFontSize #1.\rule{\captionGap}{0in}#2.%
}

```

`\new@makecaption` Define the new `@makecaption` code, which is defined in terms of long- and short-caption definitions, that can be changed on the fly.

```
\long\def\new@makecaption#1#2{%
  \vskip\abovecaptionskip%
  \sbox\@tempboxa{\CaptionFontSize #1.\rule{\captionGap}{0in}#2.}%
  \ifdim \wd\@tempboxa >\hsize%
    \long@caption{#1}{#2}%
  \else
    \short@caption{#1}{#2}%
  \fi
  \vskip\belowcaptionskip%
}

```

`\captionStyle` `\long@caption` `\short@caption` Define the user routine `\captionStyle`, which allows the user to redefine the captions styles for long and short captions, respectively.

```
\newcommand\captionStyle[2]{%
  \if o#1\let\long@caption\offset@caption\fi
  \if n#1\let\long@caption\nooffset@caption\fi
  \if l#2\let\short@caption\shortleft@caption\fi
  \if c#2\let\short@caption\shortcenter@caption\fi
  \if r#2\let\short@caption\shortright@caption\fi
}

```

Define the default value for caption style, which is offset-style, left-aligned.

```
%% SET DEFAULT CAPTION STYLE: CAPTION ID OFFSET FOR LONG CAPTIONS,
%%                            SHORT CAPTIONS LEFT ALIGNED
\captionStyle{o}{l}

```

`\killtableofcontents` Kills subsequent calls for the Table of Contents by renewing the command as null.

```
\newcommand\killtableofcontents{%
  \renewcommand\tableofcontents{}%
}

```

`lofInvocations` `lofprints` `\oldlistoffigures` Set up for *lof* handling, by preparing to count invocations of *lof*, the number of times the *lof* is printed, and by saving prevailing definition of `\listoffigures`.

```
%%LIST OF FIGURES HANDLING:
\newcounter{lofInvocations}   \setcounter{lofInvocations}{0}
\newcounter{lofPrints}        \setcounter{lofPrints}{0}
\let\oldlistoffigures\listoffigures

```

\killlistoffigures Kills subsequent calls for List of Figures by renewing the command (and redefining the saved command) as null.

```
\newcommand\killlistoffigures{%
  \def\oldlistoffigures {}%
  \renewcommand\listoffigures{}%
}
```

\holdlistoffigures To put the *lof* "on hold," we merely redefine `\listoffigures` to increment the `lofInvocations` counter.

```
\newcommand\holdlistoffigures{%
  \renewcommand\listoffigures{\addtocounter{lofInvocations}{1}}%
}
```

\clearlistoffigures This routine will clear (i.e., print) the List of Figures the number of times it was invoked while "on hold" (most likely 0 or 1 time). It does this by incrementing `lofPrints` until it reaches a value of `lofInvocations`.

```
\newcommand\clearlistoffigures{%
  \whiledo{\arabic{lofPrints} < \arabic{lofInvocations}}{%
    \addtocounter{lofPrints}{1}%
    \oldlistoffigures%
  }%
}
```

lotInvocations
lotPrints
\oldlistoftables
\killlistoftables
\holdlistoftables
\clearlistoftables
List of Tables handling is wholly analogous to List of Figures handling just described.

```
%%LIST OF TABLES HANDLING:
\newcounter{lotInvocations}   \setcounter{lotInvocations}{0}
\newcounter{lotPrints}        \setcounter{lotPrints}{0}
\let\oldlistoftables\listoftables

\newcommand\killlistoftables{%
  \def\oldlistoftables {}%
  \renewcommand\listoftables{}%
}

\newcommand\holdlistoftables{%
  \renewcommand\listoftables{\addtocounter{lotInvocations}{1}}%
}

\newcommand\clearlistoftables{%
  \whiledo{\arabic{lotPrints} < \arabic{lotInvocations}}{%
    \addtocounter{lotPrints}{1}%
    \oldlistoftables%
```

```
  }%
}
```

\hyperactive Prepare corrections if the hyperref package is being used. Set default caption treatment in boxhandler to \caption. Define alternate caption treatment as \hyper@cap. If \hyperactive is invoked, redefine caption treatment as \hyper@cap. The optional argument to \hyperactive is the downward-shift to be applied to the caption, relative to the caption label.

```
%% \hyperactive PROVIDES A CORRECTIVE CAPTION SHIFT WHEN USING THE
%% hyperref PACKAGE; OPTIONAL ARGUMENT IS SHIFT LENGTH
\let\bx@caption\caption
\newlength\hyper@shift
\newcommand\hyper@cap[1]{\caption{\vspace*{\hyper@shift}#1}}%
\newcommand\hyperactive[1][-1.55ex]{%
  \setlength\hyper@shift{#1}\let\bx@caption\hyper@cap}
```

arltable arlfigure To retain backward compatibility to the simpler predecessor of the boxhandler package, the vestigial commands \arltable and \arlfigure are provided. Their definitions are directly linked to \bxtable and \bxfigure.

```
%% TO RETAIN BACKWARD COMPATIBILITY WITH THE PREDECESSOR TO boxhandler,
%% THE FOLLOWING ASSIGNMENTS ARE MADE.
\let\arltable\bxtable
\let\arlfigure\bxfigure
```

We are done now.

⟨/package⟩