The phffullpagefigure package[1]

Philippe Faist *philippe.faist@bluewin.ch*

August 15, 2016

[1] *This document corresponds to phffullpagefigure v1.0, dated 2016/08/15. It is part of the phfqitltx package suite, see https://github.com/phfaist/phfqitltx.*

phffullpagefigure—Figures which fill up a full page of a document.

1 Introduction

The package phffullpagefigure provides an implementation for figures which are to be displayed to occupy a full page.

A typical use case: suppose you have a figure in PDF format of the size of the document paper, for example, and wants to include it as a figure.

This package takes care to display the caption of the figure on the preceeding page, with a caption of the form "Figure X (on facing page): ⟨*caption*⟩."

For two-sided documents, you may specify on which (odd/even) page side you want the figure to appear. If the document is not two-sided, the figure may appear on any page.

A number of options allow you to set the exact figure contents (usually a PDF file, but it can be constructed from arbitrary LaTeX commands), the figure caption placement (top, own page, bottom), the caption and label as usual, and the

formatting of the caption if you want to replace the default "(on facing page)" or "(on next page)."

2 The Full-Page-Figure Environment

`fullpagefigure` The `\begin{fullpagefigure} ... \end{fullpagefigure}` environment starts a full-page-figure. Inside this environment, only the following commands may be used:

- one of the `\fig***` commands;
- the `\caption` command, to provide a figure caption as for regular figures;
- the `\label` command to set a label for text references to this figure, also as for regular figures.

A simple example to get started:

```
\begin{fullpagefigure}
  \figpdf{fig/my-figure} % my PDF file
  \caption{A colorful figure with letters and words.  The table
    design may remind you of going to the optician.}
  \label{fig:test}
\end{fullpagefigure}
```

The `\figpdf` command sets the PDF file to be displayed in full page (see more details below). You may contemplate the result of this code in Figure 1.

Some wizard once told me that old wise men had determined that the `fullpagefigure` environment should be placed at the beginning of a paragraph, or on its own paragraph. (TODO: I'm not too sure why this is the case or if this is still relevant.)

The contents and appearance of the figure can be adjusted by the following commands, which must be issued within the `fullpagefigure` environment.

`\figcontents` Specify the contents of the figure by calling `\figcontents{`⟨*LaTeX commands*⟩`}`. The contents can be any LaTeX commands which will generate the figure content. These commands will be called within an `afterpage` block.[1]

[1] See documentation for the afterpage package

Figure 1 (on next page): A colorful figure with letters and words. The table design may remind you of going to the optician.

T H I S

F I G U R E

F I L L S

U P

T H E

W H O L E

P A G E

If you want a figure to occupy several pages, you may use for example `\figcontents{\includepdf{pdf-1}\includepdf{pdf-2}}`, and request that the figure start on an even-side page with `\figpageside{even}`.

If the ⟨*LaTeX commands*⟩ are not `\includepdf` instructions and want to change the page geometry for the figure and you're using the geometry package, have a look at these TeX.SX answers[2] for getting the `\newgeometry` and `\restoregeometry` commands right. You might use, for example:

```
\figcontents{\newpage\thispagestyle{empty}%
    \newgeometry{margin=0.25in}% or 0in for no margins
    ⟨... LaTeX commands to draw figure contents ...⟩
    \clearpage\aftergroup\restoregeometry}.
```

`\figpdf` As a shorthand, you may use `\figpdf[`⟨*options*⟩`]{`⟨*pdf-file*⟩`}` as a shorthand for `\figcontents{\includepdf[`⟨*options*⟩`]{`⟨*pdf-file*⟩`}}`.

You should explore the options provided by `\includepdf` (from the pdfpages package[3]). For example, the figure can be resized to fill the page, pages may be selected individually from a mult-page PDF, the image may be rotated, etc.

`\caption` `\label` The `\caption[`⟨*short caption*⟩`]{`⟨*caption*⟩`}` and `\label{`⟨*identifier*⟩`}` macros may be used as for a normal figure. Be warned, though, that some dark manipulations occur here, so it may be for example that the code passed as argument to these commands is expanded only later.

`\figpageside` Use the commands `\figpageside{odd}`, `\figpageside{}`, or `\figpageside{even}` to specify on which side the figure should appear on if the document is two-sided. This command has no effect if the document is not two-sided (`twoside` class option for the `article` or `book` classes, for example). Calling `\figpageside{}`, i.e. with an empty argument, instructs `fullpagefigure` to use either side, whichever is more convenient.

`\figplacement` Specify the figure caption placement with `\figplacement{b|t|p}`. Each of `b` (bottom), `t` (top) and `p` (own page) work as for usual LaTeX floats. If you specify `p`, do NOT combine it with any other option. You may leave the argument empty (`\figplacement{}`) to use defaults.

The figure placement can also be specified as an optional argument to the environment (e.g., `\begin{fullpagefigure}[p] ... \end{fullpagefigure}`).

`\figcapmaxheight` Specify the maximum estimated height of the caption with `\figcapmaxheight{`⟨*length*⟩`}`. This is used to see whether the figure caption still fits on the current page.

TODO: This is ugly, the height of the caption should be calculated automatically... for next time.

[2] *http://tex.stackexchange.com/a/278101/32188* and *http://tex.stackexchange.com/a/40503/32188*
[3] See documentation at *https://www.ctan.org/pkg/pdfpages*

By default (if no `\figcapmaxheight` is present), the figure will never be assumed to fit in the remainder of the page.

`\fullpagefigurecaptionfmt` The figure label in the caption may be changed (e.g. "Figure X (on facing page): ...") by redefining the command `\fullpagefigurecaptionfmt`. See the default implementation for more info (section 4).

`\FlushAllFullPageFigures` If you need to make sure that all full-page-figures have been placed up to a certain point, you may issue the command `\FlushAllFullPageFigures`. (You may wish to do so before starting a new chapter.) An optional argument `\FlushAllFullPageFigures[\clearpage]` or `\FlushAllFullPageFigures[\cleardoublepage]` specifies whether to continue on any page or on an odd-side page only.

■ 3 Package Options

Only a single package option is recognized:

```
\usepackage[nopdfpages]{phffullpagefigure}
```

If this package option is given, then the pdfpages package is not loaded, and the command `\figpdf` is not made available. You may use this package option if the pdfpages package conflicts with your setup.

■ 4 Implementation

Include some general useful packages first.

```
1 \RequirePackage{etoolbox}
2 \RequirePackage{ifoddpage}
3 \RequirePackage{afterpage}
```

The placeins package provies the `\FloatBarrier` command, which we use to ensure that no other float gets in the way.

```
4 \RequirePackage{placeins}
```

`phffpf@internal@pending` Counter which stores how many full-page-figures still haven't been placed. Used for `\FlushAllFullPageFigures` as well as making sure that the full-page-figures don't interfere with one another.

```
5 \newcounter{phffpf@internal@pending}
6 \setcounter{phffpf@internal@pending}{0}
```

\phffpfFloatBarrier Redefine this if you don't want to use a \FloatBarrier. Be warned of the following points:

- \FloatBarrier introduces automatically a new paragraph. Nothing you can do about that a priori.
- If you remove \FloatBarrier, you need to either be sure that there are no floats which can mess up placement of the fullpagefigure. Alternatively, you need to provide your own mechanism that ensures that.

```
\def\phffpfFloatBarrier{\FloatBarrier}
```

4.1 The Main Environment Definition

fullpagefigure The main fullpagefigure environment.

```
\newenvironment{fullpagefigure}[1][b]{%
```

Remember that we have a float pending to be placed:

```
  \addtocounter{phffpf@internal@pending}{1}%
```

Don't allow any other floats to meddle with our calculations.

```
  \phffpfFloatBarrier%
  %[YYY]% -- debugging [where is a space being inserted?]
```

The following variables will store the relevant values of options collected in the definition of the figure with e.g. \figplacement, \figcontents, etc.

NOTE TO SELF: If you add a \phffpf@val@... storage variable, don't forget to fix that value in \phffpf@takecareofplacingfigure.

```
  \xdef\phffpf@val@pageside{\phffpf@side@}%
  \gdef\phffpf@val@captionopt{}%
  \gdef\phffpf@val@caption{}%
  \gdef\phffpf@val@label{}%
  \gdef\phffpf@val@placement{#1}%
  \gdef\phffpf@val@capmaxheight{\paperheight}%
  \gdef\phffpf@val@figcontents{}%
```

If this document is two-sided (facing odd/even pages), then by default place the float on an odd page. Otherwise, we don't care.

```
  \if@twoside%
    \xdef\phffpf@val@pageside{\phffpf@side@odd}%
  \else\fi%
```

Provide a set of commands within this figure block which allow to specify the figure contents and appearance:

```
\begingroup%
\let\figcontents\phffpf@impl@figcontents%
\let\figpageside\phffpf@impl@figpageside%
\let\caption\phffpf@impl@caption%
\let\label\phffpf@impl@label%
\let\figplacement\phffpf@impl@placement%
\let\figcapmaxheight\phffpf@impl@capmaxheight%
```

Provide `\figpdf` as a shorthand, but only if applicable (i.e., the `nopdfpages` package option was not specified and the pdfpages package was loaded):

```
\phffpf@provide@figpdf%
```

Finally, ignore any spaces following this command, as well as after the `\endenvironment` command.

```
\ignorespacesafterend%
\ignorespaces%
}
```

Now, the definitions for the end of the environment:

```
{%
```

Remove any spaces which might have been inserted.

```
\ifhmode\unskip\fi%
```

Restore `\caption`, `\label`, etc. to their original meaning:

```
\endgroup%
```

Finally we should actually take care of placing the figure.

```
\phffpf@takecareofplacingfigure%
```

Finally finally, ignore any spaces following this command. Note that because the expansion of `\endfullpagefigure` is inside the definition of LaTeX' "`\end`" and has internal commands after that, we can't just simply issue a `\ignorespaces`.

```
\phfpf@useignorespacesandallpars%
}
```

`\phffpf@useignorespacesandallpars` Utility to ignore spaces and paragraphs after the `\end{fullpagefigure}` command.[4]

[4] This solution was adapted from *http://tex.stackexchange.com/a/179034/32188* and *http://tex.stackexchange.com/a/23101/32188*.

```
\def\phfpf@useignorespacesandallpars#1\ignorespaces\fi{%
  #1\fi\phffpf@ignorespacesandallpars}
\def\phffpf@ignorespacesandallpars{%
  \begingroup%
  \catcode`\^^M=10\relax%
  \catcode`\^^J=10\relax%
  \@ifnextchar\par%
    {\endgroup\expandafter\phffpf@ignorespacesandallpars\@gobble}%
    {\endgroup}%
}
```

\fullpagefigurecaptionfmt
\fullpagefigurecaptionfmt@paren@O
\fullpagefigurecaptionfmt@paren@E
\fullpagefigurecaptionfmt@paren@x

The macro `\fullpagefigurecaptionfmt` is called to generate the text which is prepended to the figure caption. It should essentially say "Figure X (on next page): ".

The argument to `\fullpagefigurecaptionfmt` is `#1` = `O`, `E` or `x` for if the figure is on an odd page, an even page, or an unspecified page.

```
\def\fullpagefigurecaptionfmt#1{%
  \figurename\nobreakspace\thefigure\nobreakspace%
  (\csname fullpagefigurecaptionfmt@paren@#1\endcsname)%
}
\def\fullpagefigurecaptionfmt@paren@O{on facing page} % for odd page figures
\def\fullpagefigurecaptionfmt@paren@E{on next page}   % for even page figures
\def\fullpagefigurecaptionfmt@paren@x{on next page}   % for next-page figures
```

4.2 Implementation of `\fig****` Commands

These macros really just store their values for later use.

\figcontents

This macro will become `\figcontents` inside the `fullpagefigure` environment.

```
\newtoks\phffpf@tmp@toks
\long\def\phffpf@impl@figcontents#1{%
  \phffpf@tmp@toks={#1}%
  \xdef\phffpf@val@figcontents{\the\phffpf@tmp@toks}%
  \ignorespaces%
}
```

\phffpf@side@odd
\phffpf@side@even
\phffpf@side@

These hold one-character codes to signify "odd side," "even side," or "no specification."

```
\def\phffpf@side@odd{O}
\def\phffpf@side@even{E}
\def\phffpf@side@{x}
```

`\figpageside` This will become `\figpageside` inside the `fullpagefigure` environment.

```
\def\phffpf@impl@figpageside#1{%
  \ifcsname phffpf@side@#1\endcsname%
    \xdef\phffpf@val@pageside{\csname phffpf@side@#1\endcsname}%
  \else%
    \PacakgeError{phffullpagefigure}{Unknown page side designation:
      '#1'. Please use 'odd', 'even', or '' for no preference.}%
  \fi%
  \ignorespaces%
}
```

`\caption` This will become `\caption` inside the `fullpagefigure` environment.

```
\def\phffpf@NOARG{}
\def\phffpf@test@NOARG{\phffpf@NOARG}
\newcommand\phffpf@impl@caption[2][\phffpf@NOARG]{%
  \gdef\phffpf@val@captionopt{#1}%
  \gdef\phffpf@val@caption{#2}%
  \ignorespaces%
}
```

`\label` This will become `\label` inside the `fullpagefigure` environment.

```
\def\phffpf@impl@label#1{%
  \gdef\phffpf@val@label{#1}%
  \ignorespaces%
}
```

`\figplacement` This will become `\figplacement` inside the `fullpagefigure` environment.

```
\def\phffpf@impl@placement#1{%
  \gdef\phffpf@val@placement{#1}%
  \ignorespaces%
}
```

`\figcapmaxheight` This will become `\figcapmaxheight` inside the `fullpagefigure` environment.

```
\def\phffpf@impl@capmaxheight#1{%
  \gdef\phffpf@val@capmaxheight{#1}%
  \ignorespaces%
}
```

4.3 Placing the figures

Here's the gory details of how the figures are placed.

\phffpf@place@pending@figs@code

This macro will store code to be executed after the next figure has been placed. This can be used to queue other figures to be placed later.

```
\def\phffpf@place@pending@figs@code{\phffpf@place@pending@figs@code@start}
```

pf@place@pending@figs@code@start

When another figure is placed, and the \phffpf@place@pending@figs@code is updated, then the macro \phffpf@place@pending@figs@code@start contains the code which reinitializes \phffpf@place@pending@figs@code.

This reinitialization code consists in precisely making sure that a future execution of \phffpf@place@pending@igs@code@start will start by reinitializing that macro.

```
\def\phffpf@place@pending@figs@code@start{%
  \gdef\phffpf@place@pending@figs@code{\phffpf@place@pending@figs@code@start}}
```

\phffpf@impl@figcode

The code to be inserted to generate the figure.

The argument #1 is the prefix for macro names where to look up the contents of the figure and values of the figure settings. The macro names are determined as \csname #1@⟨*field-name*⟩\endcsname.

```
\gdef\phffpf@impl@figcode#1{%
```

Do we have a figure placement position request (p, t, b)? If yes, then define a macro which we will expand in front of the \begin{figure} command for the caption. If no, then that macro should be left blank (first case below):

```
  \expandafter\ifblank\expandafter{\csname #1@placement\endcsname}{%
    \edef\phffpf@tmp@figplacementarg{}%
  }{%
    \edef\phffpf@tmp@figplacementarg{[\csname #1@placement\endcsname]}%
  }
```

Invoke the figure environment, which we use to typeset the caption. Use specified placement if applicable. Set up some basic stuff in the figure: the contents, caption and label.

```
  \expandafter\figure\phffpf@tmp@figplacementarg%
  \centering%
  \begingroup%
  \def\fnum@figure{\fullpagefigurecaptionfmt{\csname #1@pageside\endcsname}}%
  \expandafter\afterpage\expandafter{\csname #1@figcontents\endcsname}%
  \expandafter\ifx\csname #1@captionopt\endcsname\phffpf@test@NOARG%
    \expandafter\caption\expandafter{\csname #1@caption\endcsname}%
  \else%
    \def\phffpf@tmp@captioncmdopt{%
      \expandafter\caption\expandafter[\csname #1@captionopt\endcsname]}%
    \expandafter\phffpf@tmp@captioncmdopt\expandafter{\csname #1@caption\endcsname}%
  \fi%
```

```
  \expandafter\notblank\expandafter{\csname #1@label\endcsname}{%
    \expandafter\label\expandafter{\csname #1@label\endcsname}%
  }{%
  }
  \endgroup%
  \endfigure%
```

Now we have placed the `figure`, so decrease our "pending-to-be-placed" counter.

```
  \addtocounter{phffpf@internal@pending}{-1}%
```

... and execute the code to place any other pending figures. (We set `\ifphffpf@flag@forcenextmaybequeuetoplacefigure` to TRUE to force the next figure in queue to be placed now.)

```
  \afterpage{%
    \phffpf@flag@forcenextmaybequeuetoplacefiguretrue%
    \phffpf@place@pending@figs@code%
  }%
}
```

Now, all options have been set etc., the fullpagefigure environment has finished, so calculate the commands to place the figure appropriately.

`\phffpf@takecareofplacingfigure` First, fix the values of the contents and settings (in case another full-page-figure comes along and messes up the `\phffpf@val@...` commands).

After the values have been fixed (in fact they are stored in the form of "restore code"), then we delegate to `\phffpf@maybequeuefigurecode`, which checks whether we can place a figure or if we should queue.

```
\def\phffpf@takecareofplacingfigure{%
```

A tricky part: make sure we save the values of `phffpf@val@`⟨*field*⟩ in a fixed way so that several figures won't overwrite each other's values.

We build a bunch of tokens which are in fact restore code for the given variables, i.e., which is a list of commands of the form `\gdef\phffpf@val@`⟨*field*⟩`{`⟨*first-level-expanded-value-of-this-field*⟩`}`. This set of tokens have the values of these variables expanded to the first level, so that it is OK if the variables `\phffpf@val@`⟨*field*⟩ are overwritten.

```
  \edef\phffpf@tmp@fixallfieldvalues{%
    \noexpand\gdef\noexpand\phffpf@val@pageside{\expandonce\phffpf@val@pageside}%
    \noexpand\gdef\noexpand\phffpf@val@captionopt{\expandonce\phffpf@val@captionopt}%
    \noexpand\gdef\noexpand\phffpf@val@caption{\expandonce\phffpf@val@caption}%
    \noexpand\gdef\noexpand\phffpf@val@label{\expandonce\phffpf@val@label}%
    \noexpand\gdef\noexpand\phffpf@val@placement{\expandonce\phffpf@val@placement}%
    \noexpand\gdef\noexpand\phffpf@val@capmaxheight{\expandonce\phffpf@val@capmaxheight}%
```

```
    \noexpand\gdef\noexpand\phffpf@val@figcontents{\expandonce\phffpf@val@figcontents}%
  }%
```

Finally, relay the call to `\phffpf@maybequeuefigurecode{`⟨*restore-code-for-figure-settings*⟩`}{`⟨*full-figure-code*⟩`}`.

```
  \edef\phffpf@tmp@figcodetwoargs{%
    {\expandonce\phffpf@tmp@fixallfieldvalues}%
    {\noexpand\phffpf@impl@figcode{phffpf@val}}%
  }%
  \expandafter\phffpf@maybequeuefigurecode\phffpf@tmp@figcodetwoargs%
}
```

`\phffpf@maybequeuefigurecode` USAGE: `\phffpf@maybequeuefigurecode{`⟨*restore-code-for-figure-settings*⟩`}{`⟨*full-figure-code*⟩`}`.

Checks if we can place the figure; if yes then place it on the right page, if no, then add it to the queue.

The arguments are: `#1` = code to restore correct `\phffpf@val@XYZ` values; `#2` = figure code. Make sure it's expanded.

```
\long\def\phffpf@maybequeuefigurecode#1#2{%
```

Possibly we have been told to place the next figure now via the flag `\ifphffpf@flag@forcenextmaybequeuetoplacefigure`. In this case, reset the flag and place the figure now (relay to `\phffpf@doplacefigure`).

```
  \ifphffpf@flag@forcenextmaybequeuetoplacefigure%
    \phffpf@flag@forcenextmaybequeuetoplacefigurefalse
    \phffpf@doplacefigure{#1}{#2}%
    %
  \else
```

See if there are other figures waiting to be placed first. If so, add ours to the queue.

```
    \ifnum\value{phffpf@internal@pending}>1\relax%
      \xdef\phffpf@place@pending@figs@code{%
          \expandonce\phffpf@place@pending@figs@code%
          \unexpanded{\phffpf@maybequeuefigurecode{#1}{#2}}%
        }%
        %\show\phffpf@place@pending@figs@code
        %[figure queued:  \texttt{\detokenize{#1}}]%  -- DEBUGGING
    \else%
```

If not, deal with placing the figure now:

```
      \phffpf@doplacefigure{#1}{#2}%
      %[figure placed: \texttt{\detokenize{#1}}]  -- DEBUGGING
    \fi%
```

```
  \fi%
}
```

Define also the flag which will force a next call to \phffpf@maybequeuefigurecode to place the next figure in the queue.

```
\newif\ifphffpf@flag@forcenextmaybequeuetoplacefigure
\phffpf@flag@forcenextmaybequeuetoplacefigurefalse
```

\phffpf@doplacefigure Place the figure now. Determine the correct number of \afterpage's to use so that the figure caption ends up on the correct page side.

The arguments to this macro are: #1 = code to restore correct phffpf@val@XYZ values, #2 = the figure code. Make sure it's expanded.

```
\long\def\phffpf@doplacefigure#1#2{%
```

Make sure the correct values of phffpf@val@XYZ are restored, because we need e.g. \phffpf@val@pageside. They may be wrong because this might be called after a figure has been queued.

```
  #1%
```

Now, determine where exactly to place the figure code. There are no other pending figures. If there is no side preference, just place the figure pretty much now.

```
  \ifx\phffpf@val@pageside\phffpf@side@%
    \let\phffpf@tmp@doplace\@firstofone%
  \else%
```

If, however, we have a side preference, then check everything more carefully. Use the helper macros \phffpf@placecode@on⟨same|other⟩parity. (The latter essentially expand to the correct number of \afterpage's.)

```
    \ifx\phffpf@val@pageside\phffpf@side@odd%
      %[CHECK DONE HERE/WANT ODD] % -- for debugging
      \checkoddpage\ifoddpage%
      %[IS ODD] % -- for debugging
        \let\phffpf@tmp@doplace\phffpf@placecode@onotherparity%
      \else%
        %[IS NOT ODD] % -- for debugging
        \let\phffpf@tmp@doplace\phffpf@placecode@onsameparity%
      \fi%
    \else%
      %[CHECK DONE HERE/WANT EVEN] % -- for debugging
      \checkoddpage\ifoddpage%
        %[IS ODD] % -- for debugging
        \let\phffpf@tmp@doplace\phffpf@placecode@onsameparity%
```

```
      \else%
        %[IS NOT ODD] % -- for debugging
        \let\phffpf@tmp@doplace\phffpf@placecode@onotherparity%
      \fi%
    \fi%
  \fi%
```

I think an `\hbox{}` might help to place the anchor which determines which page side we are currently on. Note that this starts a new paragraph and enters horizontal mode.

```
  \leavevmode\hbox{}%
```

Now, do place the figure somewhere.

```
  \phffpf@tmp@doplace{#1#2}%
}
```

`\phffpf@placecode@onsameparity` Place the figure code on the same parity (page side) as we are currently. If enough space remains on the current page, place the figure immediately. Otherwise, use two `\afterpage`'s so as the figure caption to appear in two pages.

```
\newdimen\phffpf@tmp@spaceleft
\newdimen\phffpf@tmp@compareto
\long\def\phffpf@placecode@onsameparity#1{%
```

First, see if the caption itself requires to be on its own page (and thus no height calculations are necessary, and an additional `\clearpage` is required)

```
  \def\@tmpa{p}%
  \ifx\phffpf@val@placement\@tmpa%
    \afterpage{\vspace*{0pt}\afterpage{#1\clearpage}}%
  \else%
```

Otherwise, the figure caption is there along with some text on the page.

See if there is enough place left on this page to place the figure caption; otherwise use two `\afterpage`'s.

```
    %[PLACING FIG CODE ON SAME PARITY]% -- debugging
    \phffpf@tmp@spaceleft=\textheight\relax%
    \phffpf@tmp@compareto=\phffpf@val@capmaxheight\relax%
    \advance\phffpf@tmp@spaceleft by -\pagetotal%
    %[DIM LEFT: \the\phffpf@tmp@spaceleft]%
    \ifdim\phffpf@tmp@spaceleft>\phffpf@tmp@compareto%
      %[ENOUGH DIM LEFT.] % -- debugging
      #1%\phffpf@tmp@figcode%
    \else%
      %[*NOT ENOUGH* DIM LEFT.] % -- debugging
      \afterpage{\vspace*{0pt}\afterpage{#1}}%
```

```
    \fi%
  \fi%
}
```

\phffpf@placecode@onotherparity Place the figure caption on the opposite parity as the current page. This just requires one \afterpage so as the figure code to appear on the following page.

```
\def\phffpf@placecode@onotherparity#1{%
  %[PLACING FIG CODE ON OTHER PARITY]% -- debugging
```

First, see if the caption requires to be on its own page (and thus no height calculations are necessary, and an additional \clearpage is required).

```
  \def\@tmpa{p}%
  \ifx\phffpf@val@placement\@tmpa%
    \afterpage{#1\clearpage}%
  \else%
```

The figure caption is not on its own page. Just use a simple \afterpage.

```
    \afterpage{#1}%
  \fi%
}
```

4.4 Commands to Flush All Full-Page-Figures

Here are a set of commands which can be used to ensure that all full-page figure floats have been placed.

\FlushAllFullPageFigures The name is pretty self-explanatory. The command is documented in the user doc above.

```
\newcommand\FlushAllFullPageFigures[1][\phffpf@clearpage]{%
```

As long as there are full-page-figures pending, clear pages until those figures have been placed.

```
  \ifnumcomp{\value{phffpf@internal@pending}}{>}{0}{%
    \clearpage%
    %[page cleared.]% DEBUG
    \FlushAllFullPageFigures[#1]% recurse again.
  }{%
```

All figures placed, all fine. We still need to flush one last time because at this point the figure code (ie. caption) has been placed only, and we want the text that follows to come after the figure itself. Here finally we use the clear command in #1 to continue on any page or on an odd-side page only.

```
    #1%
  }%
}
\def\phffpf@clearpage{\if@twoside\cleardoublepage\else\clearpage\fi}
```

4.5 Package Option Parsing

Note the singular form of the word “option.”

```
\def\phffpf@provide@figpdf{}
\newcommand\phffpf@impl@figpdf[2][]{%
  \figcontents{\includepdf[#1]{#2}}%
}
\def\phffpf@do@pdfpages{%
  \RequirePackage{pdfpages}%
  \def\phffpf@provide@figpdf{\let\figpdf\phffpf@impl@figpdf}%
}
%
\DeclareOption{nopdfpages}{\def\phffpf@do@pdfpages{}}
\DeclareOption*{%
  \@unknownoptionerror%
}
\ProcessOptions\relax
%
\phffpf@do@pdfpages
```

Change History

v1.0

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.