Page Layout with Crop Marks

Zdeněk Wagner
http://icebearsoft.euweb.cz

Package date: 2020/02/28

Abstract

This package was developed as a typographers toolbox offering the most important features for everyday work. First it allows setting the paper size as well as the page layout. The next important feature is the ability of printing crop marks both with TEX + dvips or (x)dvipdfm(x) and with pdfTEX. Finally it is possible to reflect pages both horizontally and vertically, select black overprint and set various PDF/X features.

Contents

1 Introduction

Users will probably ask: "Why another page layout package? Is not GEOMETRY all what we need?" The answer to this question is not easy. The GEOMETRY package provides a lot of features, maybe even too many features. However, some features are missing. Packages for crop marks can also be found but correct positioning of the crop marks requires cooperation with the page layout package. It is therefore natural to integrate both these functions into a single package.

The package is useful also for preparation of book covers. It places the marks showing the position of a spine and optionally flaps. If you want to see whether the spine and the front and back cover contents are properly aligned, you can display the frames. Moreover, it is often necessary to create a proof of the cover while the book is not yet finished and the thickness of the spine is not known and can be only approximated. You can thus leave the paper width empty and the package will calculate it from the page width, the flap width and the spine thickness. Whenever the final value of the spine thickness is known, you change it in the list of options and the whole layout will adapt automatically.

The files usually fail to meet PDF/X conformance due to silly reasons. It is always mandatory to set the title, creation date and modification date in any PDF file. It can be done by the HYPERREF package. However, HYPERREF is a complex package that may cause clashes with other packages. We do not want to kill ants by cannon balls, therefore we do simple things by simple means. Options for partial PDF/X conformance were added but may be switched off if a user wishes to use HYPERREF.

The package is a result of a long-term development inspired by author's everyday needs. It is a collection of macros that were manually copied from document to document, later placed into private packages and finally it matured into a package prepared for general use.

Being a result of development it may happen that you may be in the same situation in which I was myself. The book was finished and the last task was to add the crop marks. Removing layout definition from a document and replacing it with a different package may cause errors and one more proof-reading is time consuming and expensive. This package therefore allows the page layout logic to be switched off and just add the crop marks provided the paper dimensions are correctly supplied. The details will be explained later when describing the package options.

2 Installation

The package consists of a single file, `zwpagelayout.sty`. Put it to a directory where LaTeX expects packages, preferably `texmf-dist/tex/latex/zwpagelayout`. All remaining files belong to the documentation, put them to a directory where documentation (including documentation sources) is expected by TEXDOC or a similar program, preferably `texmf-dist/doc/zwpagelayout`.

3 Package dependence

As written in the introduction, the goal was to implement as much within this single package in order to reduce the risk of clashes. Yet a few packages may be loaded. The package needs to know what engine is being used. For this purpose the IFTEX package[1] is used. If the package is not found, it is assumed that the engines are not available. No error is reported. The color support requires the COLOR package. It is loaded only if the color support is requested. The algorithm for deciding when the package is needed will be described in detail in section 12.

Option setting is implemented via the KVOPTIONS package which in turn loads the KEYVAL package. These packages should be available in any nowaday's TeX distribution.

4 Usage

The package is loaded simply by:

```
\usepackage[⟨options⟩]{zwpagelayout}
```

Remember that the options cannot be given later, all of them must be present either here or as options to the `\documentclass` command. The package will set some

[1] Up to version 1.4c both IFXETEX and IFPDF were used. Nowadays both these packages are deprecated and load IFTEX. The package would fail with a critical error if they were used.

dimensions based upon `\normalsize`. The package that modifies font sizes must therefore be loaded before ZWPAGELAYOUT.

5 Option values

Some options accept a value. If the option is not specified, it may have a **default** value. For ease of use an option may be given without a value. It will then have a **standard** value. The default and standard values will be given in the option description.

The option value may sometimes contain spaces and commas. However, commas are used to separate options, and during parsing the spaces are ignored. The spaces must therefore be preceded by a backslash and an option value containing commas must be enclosed in braces, for example:

```
\usepackage[a4,subject=Some\ subject,
            keywords={keyword\ 1,\ keyword\ 2}]{zwpagelayout}
```

6 Driver selection

The driver is usually selected automatically. The package makes use of the IFXETEX and IFPDF packages to detect either XeTeX or pdftex. If none of them is detected, dvips is assumed.

It may sometimes be necessary to set the driver manually by giving the DRIVER option. This option recognizes drivers `unknown`, `pdftex`, `xetex`, and `dvips`. In addition `other` is an alias to `unknown`, `dvipdfm` and `xdvipdfmx` are aliases to `xetex`. DRIVER

The package shows the driver in the log file. If the driver is set to `unknown`, all driver specific features will be disabled.

7 Paper size and orientation

The first task of the package is to define paper size and orientation. Remember that the package is intended for use in real life where we cannot be limitted to standard papers sizes. However, the standard paper sizes are used quite often and we must be able to use them too. The options were tested with pdfTeX, DVIPS as well as (X)DVIPDFM(X).

7.1 Orientation

The package usually accepts the dimentsions as they are, i. e. width and height of the page. The orientation cannot be changed. If you wish to have the possibility to change the orientation, you have to use this option. It is used automatically with all standard paper sizes (see section 7.2.2). You will rarely specify it yourself. ALLOWWIDTHHEIGHTSWITCHING

These options sets the paper orientation, PORTRAIT is the default. They can be used only if ALLOWWIDTHHEIGHTSWITCHING was specified, otherwise they are ignored. PORTRAIT LANDSCAPE

7.2 Paper size

The package option defines the trimmed paper size. If the crop marks are added, the real paper size will be added. Dimensions `\paperwidth` and `\paperheight` will thus

contain the values that will subsequently be sent to the driver. If you for any reason wish to know the trimmed dimensions, you can calculate them as follows:

$$\begin{aligned}\text{trimmed width} &= \texttt{\textbackslash paperwidth} - 2\texttt{\textbackslash hoffset}\\ \text{trimmed height} &= \texttt{\textbackslash paperheight} - 2\texttt{\textbackslash voffset}\end{aligned}$$

7.2.1 Generic paper size

PAPERSIZE

Option PAPERSIZE is used to define arbitrary paper size. You define it as a pair of two dimensions, the width and the height. The numbers are separated by a comma, therefore the specification must be enclosed within braces, e. g.

```
\usepackage[papersize={200mm,100mm}]{zwpagelayout}
```

The above written command sets the paper width to 200 mm and the paper height to 100 mm, i. e. the paper orientation will be landscape. Now consider the following:

```
\usepackage[papersize={200mm,100mm},
    AllowWidthHeightSwitching]{zwpagelayout}
```

Since the width/height switching is allowed and the default orientation is portrait (option PORTRAIT is the default), the width will be 100 mm and the height will be 200 mm.

You can achieve the same result as the first command by the lengthy code:

```
\usepackage[papersize={200mm,100mm},
    AllowWidthHeightSwitching,Landscape]{zwpagelayout}
```

You see that the command may become confusing. You should better not use ALLOWWIDTHHEIGHTSWITCHING together with PAPERSIZE.

If the PAPERSIZE option is not used, it defaults to the A4 size, i. e. to paper dimensions 210 mm × 297 mm.

If the book cover is being typeset, we want to calculate at least the width. The detailed explanation will be given in section 9. Here we put just as an example:

```
\usepackage[papersize={,200mm},spine=15mm,textwidth=14cm,
            margins]{zwpagelayout}
```

It is even possible to calculate both dimensions using e. g.

```
\usepackage[papersize,spine=15mm,textwidth=14cm,textheight=20cm,
            margins]{zwpagelayout}
```

7.2.2 Standard paper sizes

A0 . . . C10
EXECUTIVE
LEGAL
LETTER

Options A0. . . A10, B0. . . B10, C0. . . C10 are used to select paper size according to the A, B or C series where the dimensions are rounded to integers in milimeters. For instance, the A6 size is 105 mm × 148 mm. The package also supports paper sizes EXECUTIVE (7.25 in × 10.5 in), LEGAL (8.5 in × 14 in), and LETTER (8.5 in × 11 in). If one of these options is used, ALLOWWIDTHHEIGHTSWITCHING and PORTRAIT are assumed. The page orientation may be switched by option LANDSCAPE (see section 7.1).

7.3 Page bounding boxes

The drivers usually set MediaBox to the physical size of the page. If cropmarks are requested (see section 11), the package sets also BleedBox and TrimBox, ArtBox is explicitely forbidden by PDF/X. CropBox is intentionally unset since it causes cropped display in Adobe Reader. Since page size setting is delayed, MediaBox contains the whole page including the area for the cropmarks. MediaBox is calculated by the driver, the other boxes are calculated by TEX. Their dimensions are therefore not exact because TEX does not use floating point arithmetic. The difference is negligible.

Option noBboxes disables setting the above mentioned boxes. Currently there is a known bug causing fatal error in Adobe Distiller if these boxes are set. noBboxes

Macro `\noBboxes` performs exactly the same action as the noBboxes option. The only advantage is that it can be used later in the preamble of the document. It cannot be used after `\begin{document}` because the boxes are already set at that time. `\noBboxes`

8 Page layout options

These options define the layout of the odd page. The layout of the even page is assumed to be a mirror image. The values of `\oddsidemargin` and `\evensidemargin` are calculated from the values of the options.

8.1 Margins

This set of options us used to define margin sizes. Remember that all margins are measured from the top left corner of the paper, not from 1in border which is the DVI origin!

8.1.1 Option margins, standard 0 mm

This option sets all margins (top, left, right, bottom) to the specified size. You will not be able to change some of the margin to another size. If you need different size for any margin, you have to set all of them separately and not use this option. Notice that almost everything may be calculated automatically, thus it is not necessary to specify all values. margins

8.1.2 Option topmargin, default 1 in

This option sets the value of `\topmargin`. topmargin

8.1.3 Option botmargin, standard topmargin

This option specifies the size of the margin below the running footer. botmargin

8.1.4 Option leftmargin, default −1 in

This option sets the value of the margin at the left side of an odd page (and the margin at the right side of an even page). Negative value means that the value was not given. leftmargin

8.1.5 Option RIGHTMARGIN, default −1 in

This is the size of the right side of an odd page. Similarly the negative value means that the option was not specified. RIGHTMARGIN

8.2 Text dimensions

The following options are used to specify the height and width of the text.

8.2.1 Option TEXTWIDTH, default −1 in

This option specifies the width of the text without marginal notes. Its meaning is different for book covers. The option specifies the width of the cover without the extra spine (XSPINE), flap (FLAP) and trimming (XTRIM or TRIM). TEXTWIDTH

8.2.2 Option TEXTHEIGHT, default −1 in

This option specifies the total height of the text including the header and footer. TEXTHEIGHT

8.2.3 Option HEADHEIGHT, default 0 mm

This option specifies the height of the running head. HEADHEIGHT

8.2.4 Option HEADSEP, default 0 mm

This is the value of `\headsep`. HEADSEP

8.2.5 Option FOOTSKIP, default 0 mm

This option specifies the value of `\footskip`. FOOTSKIP

8.2.6 Option TOPSKIP, default `\topskip`

This option enables changing the value of `\topskip`. It is rarely needed. However, the value of `\topskip` is usually less than `\baselineskip` which may cause problems in some page layout schemes. Since we have already written that font sizes must be defined before ZWPAGELAYOUT is loaded, you can safely specify `topskip=\baselineskip`. TOPSKIP

8.2.7 Option STRICTHEIGHT, default *false*, standard *true*

The value of the LaTeX dimension `\textheight` is calculated from the values of the options and later adjusted so that together with `\topskip` it represents an integer number of lines. This may not be desirable in some cases. This option instructs the package not to perform any adjustment and accept the calculated value. STRICTHEIGHT

8.2.8 Option ADJUSTFOOTSKIP, default *true*, standard *true*

If option STRICTHEIGHT is not given, the calculated value of `\textheight` is decreased to the nearest integer multiple of `\baselineskip`. The difference is then added to `\footskip` in order to preserve to total height. If ADJUSTFOOTSKIP is set to *false*, the difference is added to `\headsep` instead. ADJUSTFOOTSKIP

The difference betweem these two options is shown in figure 1. Adjustment of `\footskip` was achieved by:

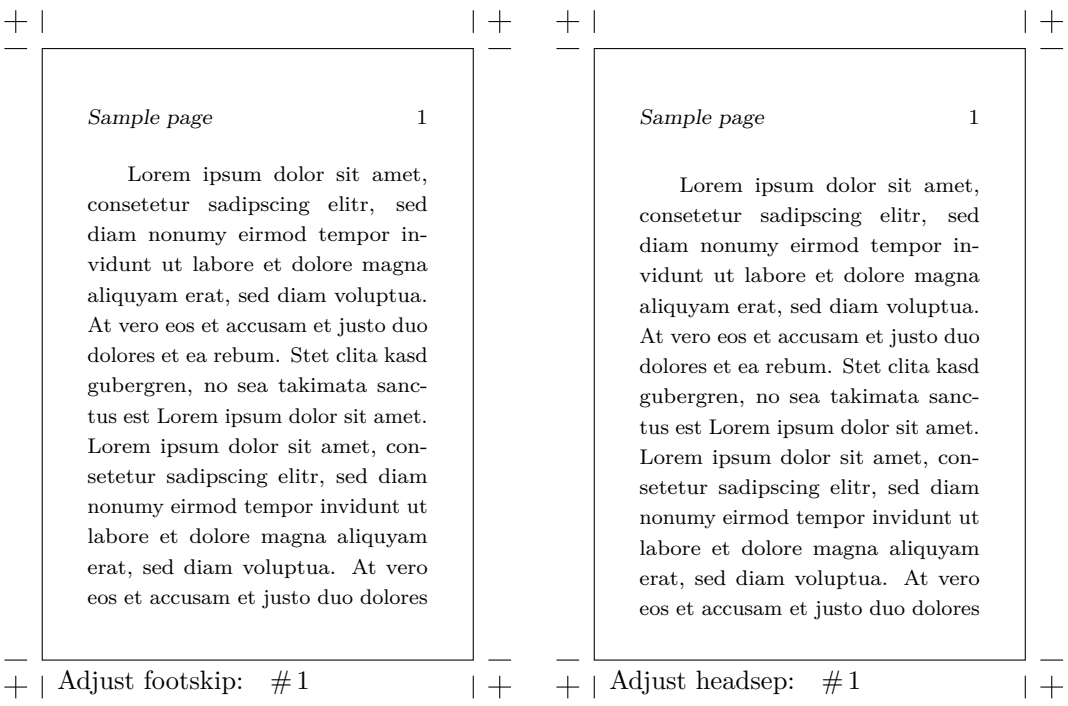

Figure 1: Compensation of `\textheight` by changing `\footskip` (on the left) and `\headsep` (on the right)

```
\usepackage[c8,margins=6mm,headheight=4mm,headsep=4mm,
   croplength=3mm,cropgap=2mm,
   cropmarks,cropframe,croptitle=Adjust\ footskip]{zwpagelayout}
```

The sample on te right side of the figure was created by:

```
\usepackage[c8,margins=6mm,headheight=4mm,headsep=4mm,adjustheadsep,
   croplength=3mm,cropgap=2mm,
   cropmarks,cropframe,croptitle=Adjust\ headsep]{zwpagelayout}
```

8.2.9 Option ADJUSTHEADSEP, default *true*, standard *true*

This is a complementary option to ADJUSTFOOTSKIP. ADJUSTHEADSEP

9 Calculation of missing dimensions

The package can either calculate paper size from the page layout dimensions or calculate missing page layout dimensions if the paper size and some dimensions are known. It operates separately for the height and width. You can e. g. define the paper height and calculate the text height from it and the margins but specify all width layout dimensions and calculate the paper width.

When designing simple pages it is better to define the paper size and calculate some page layout dimensions. However, for bok covers it is recommended to calculate at least the paper width from the layout dimensions of the front cover, the spine and the flap width.

9.1 Calculation of the paper size

Remember that paper size can be calculated only if all page layout dimensions for the corresponding orientation (height, width) are specified. There is no diagnostics for warning you if some important options are missing, the resukt will just be wrong or the package may even report an error. All dimensions are considered *without* the area for the crop marks.

9.1.1 Paper height

Calculation of the paper height is very simple. The formula is:

`\paperheight` = TOPMARGIN + TEXTHEIGHT + BOTMARGIN + 2 YTRIM

Remember that TEXTHEIGHT includes also `\headheight`, `\headsep`, and `\footskip`. It is not necessary to specify YTRIM, the package checks the existence of this option and removes it from the formula if not given.

9.1.2 Paper width

Algorithm for calculating the paper width differs for simple pages and book covers. In the simple case the paper width is calculated as follows:

`\paperwidth` = LEFTMARGIN + TEXTWIDTH + RIGHTMARGIN

The value of `\textwidth` will be equal to TEXTWIDTH. The same value will be stored in macro `\UserWidth`, see section 15.2.

If the book cover is designed, the TEXTWIDTH option refers to the width of the text at the front cover but LEFTMARGIN and RIGHTMARGIN are used to set `\oddsidemargin` and `\evensidemargin`. It is therefore better to set these options to zero, or alternatively to the same value as FLAP or XTRIM if the corresponding area will be left empty. The value of `\textwidth` will then be calculated from the final paper width by:

`\textwidth` = `\paperwidth` − LEFTMARGIN − RIGHTMARGIN

If the FLAP option is used without the SPINE option in order to emulate the front cover and spine with an empty back cover, the paper width is calculated as:

`\paperwidth` = LEFTMARGIN + FLAP + TEXTWIDTH + XTRIM + RIGHTMARGIN

The value of XTRIM may be omitted. The value of XSPINE will be silently ignored. It is not allowed to have just the spine, the front cover and the flap while leaqving the back cover empty.

The case of a book cover with the spine given is a bit more complicated:

width = XSPINE + TEXTWIDTH + FLAP + XTRIM
`\paperwidth` = LEFTMARGIN + SPINE + 2 *width* + RIGHTMARGIN

Options XSPINE, FLAP, and XTRIM need not be specified if these elements are not needed.

9.2 Calculation of page layout dimensions

Algorithm for calculating the page layout dimensions is intended for simple pages, not for book covers. Options XTRIM, YTRIM, SPINE, XSPINE, and FLAP are silently ignored but will be taken into account when producing the crop marks. You can still make use of this algorithm if you understand what you are doing and if you wish to do extra calculations yourself.

The dimensions may be overdetermined. In such a case the algorithm disregards one of the dimensions and calculates it.

9.2.1 Vertical dimensions

The algorithm first looks whether TEXTHEIGHT was given. If not it is assumed that the user wishes to have it calculated from the paper height and the margins. If the BOTMARGIN option was not set, it will be set to the same valu as TOPMARGIN.

In the second step the package looks whether BOTMARGIN is defined, either by the user or from the previous step. If so, the text height is calculated, otherwise the bottom margin is calculated from the top margin and the text height. As a matter of fact, the bottom margin is never used by TEX.

Finally the value of `\textwidth` is reduced by `\headheight`, `\headsep`, and `\footskip`. If STRICTHEIGHT is `\false`, the values are later adjusted according to options ADJUSTFOOTSKIP and ADJUSTHEADSEP so that `\textheight` – `\topskip` is an integer multiple of `\baselineskip`.

9.2.2 Horizontal dimensions

The sum of the horizontal dimensions must be equal to the paper width according to the relation:

LEFTMARGIN + TEXTWIDTH + RIGHTMARGIN = `\paperwidth`

If all three dimensions are specified, TEXTWIDTH is disregarded and calculated from the other dimensions. If any two dimensions are given, the missing one is calculated from the above formula. If only one dimension is given, it is assumed that both margins have the same size and the above formula is applied. It is even possible to omit all dimensions. In such a case the margins are assumed to have the same size as TOPMARGIN and the text width can thus be calculated.

10 Page reflection

REFLECT-HORIZONTALLY
REFLECTVERTICALLY

We sometimes need to print the whole document as a mirror image. Although there are external tools that can provide such a job taking PDF or PS as input, it is useful to do everything in a single step. The package provides options REFLECTHORIZONTALLY for horizontal reflection and REFLECTVERTICALLY for vertical reflection, respectively. If you specify both, some drivers may interpret both of them and print rotated 180°, some interpret only one of them.

The PostScript to PDF converters optionally rotate pages according to the text direction. They may be confused by reflected text and add undesired rotation. If the horizontally reflected text is roteted 180°, it looks as if it were reflected vertically instead.

Remember that these options are intended for printing only. The hypertext links made by the HYPERREF package will be wrong. If you wish to rotate parts of texts and preserve hyperlinks, use ROTATING instead.

A word of warning has to be said. In `pdftex` reflection is implemented by redefininng `\shipout`. We add PDF literal code to the beginning of each page. For `dvips` we add code to the `bop-hook`. If you need your own code in the `bop-hook`, you have to store the old definition and execute it. For `xetex` we add code to `bop` and `eop`. It seems that only one `bop` and one `eop` can be used. That is why the page is reflected vertically if both options are used. As a result you are not allowed to use your own `bop` and `eop` code together with these options.

11 Crop marks

The crop marks must appear outside the print area. The package assumes that `\paperheight` and `\paperwidth` contain the page size after trimming. These dimensions will then be increased and the page will be shifted by setting `\hoffset` and `\voffset`. If you wish to print the crop marks using ZWPAGELAYOUT, you must not change values of `\hoffset` and `\voffset`.

The values of `\hoffset` and `\voffset` will be set to the sum of CROPLENGTH and CROPGAP, see the following text.

11.1 Basic crop marks options

Options described in this section define basic behaviour of the crop marks. The options can be used in all crop mark styles.

11.1.1 Option ONLYCROPMARKS, default *false*, standard *true*

ONLYCROPMARKS

It may happen that a document is already finished and proof-read and just the crop marks have to be added. By specifying option ONLYCROPMARKS you instruct the package to ignore all page layout options and interpret the crop marks options only. You just have to ensure that `\paperheight` and `\paperwidth` are set to the dimensions of the trimmed page before the ZWPAGELAYOUT package is loaded and `\hoffset` and `\voffset` are left at their default values.

11.1.2 Option CROPMARKS, default *false*, standard *true*

CROPMARKS

This option asks for creation of the crop marks.

11.1.3 Option NOCROPMARKS, default *true*, standard *true*

NOCROPMARKS

This is a complementary option to CROPMARKS.

11.1.4 Option CROPLENGTH, default 5 mm

CROPLENGTH

This option specifies the length of the crop mark.

11.1.5 Option CROPGAP, default 5 mm

CROPGAP

This is the space that must be left blank between the crop marks and the trimmed page. This area is also known as bleed.

11.1.6 Option CROPFRAME, default *false*, standard *true*

When designing a book cover we often wish to verify whether all elements are properly aligned. Option CROPFRAME can be used to print the frames. It is active only if CROPMARKS was specified. CROPFRAME

11.1.7 Option NOCROPFRAME, default *true*, standard *true*

This is a complementary option to CROPFRAME. NOCROPFRAME

11.1.8 Option CROPSTYLE

This option selects the style of the crop marks. Two styles are defined, *default* and *leaflet*. CROPSTYLE

11.1.9 Option CROPTITLE

This option defines the text that should be printed on each page. It may e. g. be the title of the document. Remember that all spaces are gobbled when parsing the options. The spaces must therefore be specified as `\␣` or the text must be enclosed in curly braces. CROPTITLE

11.1.10 Option CROPSEPARATOR

This is the separator between the title and the page number. Its default value is `:\quad`. CROPSEPARATOR

11.1.11 Option PAGENUMBERFIRST, default *false*, standard *true*

Each page contains the title followed by the separator and the page number. If you set this option to *true*, the order will be reversed. PAGENUMBERFIRST

11.1.12 Option PAGENUMBERLAST, default *true*, standard *true*

This is a complementary option to PAGENUMBERFIRST. PAGENUMBERLAST

11.1.13 Option USEPAGENUMBERS, default *true*, standard *true*

This option requests printing the page numbers. USEPAGENUMBERS

11.1.14 Option NOPAGENUMBERS, default *false*, standard *true*

This is a complementary option to USEPAGENUMBERS. NOPAGENUMBERS

11.1.15 Option NOBLEEDCLIP, default *false*, standard *true*

The pages are clipped to the bleed box so that graphics does not collide with the crop marks. Since the package must work also with DVIPS, the PDF operators are not used and the crop marks area outside the bleed box is just overpainted with white rectangles. The COLOR package is thus required and will be loaded automatically. If you set NOBLEEDCLIP to *true*, the pages will not be clipped. It may be useful if you know that nothing extends to the crop marks area and the COLOR package may clash with other packages required by your document. NOBLEEDCLIP

11.2 Options for the *default* style

These options can be used if CROPSTYLE is set to *default* or omitted.

11.2.1 Option SPINE

This option is used for designing book covers. It specifies the width of the book spine. You can even request a zero width but negative values will have disastrous results. SPINE

11.2.2 Option XSPINE

This option defines the extra space adjacent to the spine so that the book can be safely opened. This space should not be used for printing logos etc., because they will soon be damaged by frequent use of the book. The option is ignored unless SPINE is used. XSPINE

11.2.3 Option FLAP

This is another option for book covers. It specifies the width of the flap. If SPINE is given, the book cover is supposed to have flaps of equal sizes on both sides. Sometimes you do not have flaps and the back cover remains empty, thus you do not like to prepare the design of it, you just wish to design the spine and the front cover. This can be acheved by specifying FLAP as the spine thickness and omitting SPINE. FLAP

11.2.4 Options TRIM, XTRIM, YTRIM

The paper or canvas used for printing the book covers is folded to the inner side yet part of it will be visible. Such part is specified by XTRIM in horizontal direction and YTRIM in vertical direction. You can use TRIM to set both of them to the same value. TRIM XTRIM YTRIM

11.3 Options for the *leaflet* style

The following options can be used if CROPSTYLE is defined as *leaflet*. As the value says, the style is useful for leaflet production because in addtion to the crop marks additional marks for folding. Not all options can be used with all fold types. It will be described in greater detail in the following text.

The leaflets are usually printed on both sides and sometimes one of the leaves needs width correction as will be explained later. The macros will apply the correction the one side on odd pages and on the other side on even pages. This can only happen if the TWOSIDE class option is given or a class is used where two-side printing is the default. Usually the leaflet will consist of two pages only but no check is made.

The CROPFRAME option (see 11.1.6) can also be used for previewing the leaflet layout.

11.3.1 Option LEAFCOUNT, default 4

This option defines the number of leaves to be used in the Z-type. It is silently ignored with other fold types. Internally other fold types overwrite it with the value they need in the macros. LEAFCOUNT

11.3.2 Option FOLDCORR, default 0mm

As default all leaves have equal width. However, from technical or stylistic reasons the letmost or righmost leaf requires a different width. This can be achieved by this FOLDCORR

option. The value specifies an additional horizontal skip that is applied when printing the crop/fold marks and the optional frames.

The width of the leaves is equal because it is achieved by `\hfil`. If you specify `foldcorr=-2mm`, the corrected leaf will be 2 mm narower that the others. Since there is *1fil* in the width specification of the leaves, you can achieve nice tricks be specifying `foldcorr=5cm plus -1fil`. In such a case the *1fil* in the corrected leaf will vanish and its width will be 5 cm.

Width corrections are not allowed in the *Z* and *4* types and the option value is ignored. The details of the FOLDCORR will be given in the next section.

11.3.3 Option FOLD, default 2

This option selects a fold type used with the leaflet. Supported options are *2*, *3left*, *3right*, *4*, *Z*. Their exact meaning is described below. FOLD

2 This option is used for singly folded leaflet. Usually the leaves will have different widths. The width of the left leaf is modified by the FOLDCORR option.

3left This option defines a leaflet with three leaves folded inside. For technical reasons the leaf folded inside must be slightly narrower. Typically this is the leftmost leaf. The leaf can be made narrower by applying FOLDCORR.

3right This is essentially the same as the previous option with the only difference that the FOLDCORR correction is applied to the rightmost leaf.

Z This option is used for Z-folded leaflets. The number of leaves is defined by the LEAFCOUNT option and with correction is not allowed.

4 This type of leaflet looks as the Z-type but without a fold in the middle. It can be viewed as a middle leaf of double width and the outer leaves are folded so that they touch each other. The middle crop mark is therefore missing because there is no fold there but a line will be drawn if the CROPFRAME option is requested.

As a final remark it is important to write that the width correction is applied as explained above on the odd pages. The correction of the same size is applied to the opposite side of the leaflet on even pages.

12 Color support

The package offers basic color support, namely it prints the names of separations. The color support is implemented via a few options. Color printing is performed using the COLOR package that is loaded automatically. The package does not use predefined color names.

The package recognizes the COLOR package by existence of the `\definecolor` macro. If this macro is not defined, the COLOR package will be added if:

- option COLOR is *true*
- option NOBLEEDCLIP is *false* and CROPMARKS is *true*
- option REDEFINEBLACK is *true*
- option REDEFINETOCMYK is *true*
- option OVERPRINT is *true*

If none of the above applies, the color support is not needed and the package will not be loaded.

If the COLOR package is being loaded, no options are given to it. Especially the driver selected by the DRIVER option (see section 6) is not set. If you must specify any option for the COLOR package, you have to load it yourself before loading ZWPAGELAYOUT.

12.1 Color support for cropmarks

Option COLOR asks for the color support. Without setting this option to *true* all other color options are silently ignored. COLOR

Option COLORMODEL defines the color model used. Its default value is `cmyk`. You will rarely need to change it. COLORMODEL

Option CROPCOLOR specifies the color to be used for the crop marks. Remember that the crop marks must be visible on all separations, thus it might not be sufficient to print them in black. As default their color is mixed of 100 % of all components of the current color model. Since the default model is `cmyk`, the default value of this option is `{1,1,1,1}`. Notice that the syntax conforms to the requirements of the COLOR package. CROPCOLOR

Option COLORS assigns names to the color components of the current model. Specification of each color must be enclosed in curly brackets. The color name is followed by a colon and comma separated values conforming to the syntax of the COLOR package. It will be clear from the following examples. COLORS

Suppose that for some strange reason you prepare the document in the RGB space. Your specification will then be:

```
\usepackage[cropmarks,color,colormodel=rgb,cropcolor={1,1,1},
    colors={{RED:1,0,0},{GREEN:0,1,0},{BLUE:0,0,1}},
    croptitle=RGB\ example]{zwpagelayout}
```

Now we show a more realistic example. The document should be printed in custom colors. Since the true specification of custom colors requires much work and is rarely worth the trouble, we make use of the CMYK model. We will use cyan instead of Pantone 298 (blue), magenta instead of Pantone 213 (red), black will remain black and the yellow separation will be unused. The crop marks should not produce anything on the yellow separation and we should provide proper color names. The specification will therefore look as:

```
\usepackage[cropmarks,color,cropcolor={1,1,0,1},
    colors={{Pantone\ 298\ (blue):1,0,0,0},
            {Pantone\ 213\ (red):0,1,0,0},{Black:0,0,0,1}},
    croptitle=Example\ with\ custom\ colors]{zwpagelayout}
```

Notice that the COLORMODEL option was not specified. Its default value was used. The CROPCOLOR option left zero for the yellow separation.

If a color is light as e. g. the process yellow, it may be better to display its name in white on a colored background. This is achieved by preceding the color name

with an asterisk. This is now the default behaviour for the yellow color. The default definition is:

```
colors={{CYAN:1,0,0,0},{MAGENTA:0,1,0,0},{*YELLOW:0,0,1,0},{BLACK:0,0,0,1}}
```

12.2 CMYK colors

cmykblack
cmykread
cmykgreen
cmykblue

The COLOR package defines *black* using the GRAY model and colors *red*, *green*, *blue* by the RGB model. The GRAY model rarely causes any problem but the RGB model is not acceptable for printing. The ZWPAGELAYOUT package therefore defines the corresponding colors using the CMYK model. Different names are selected so that there is no clash in case you really need these colors in RGB.

12.3 Overprinting support, default *false*, standard *true*

OVERPRINT

The overprinting support must be requested by the OVERPRINT option. If the option remains *false*, the overprinting commands will be defined but will do nothing. Remember that you can overprint any color, not just black. You thus should not enable overprinting globally. The mode is therefore set to *knockout* within the package.

\OverprintXeTeXExtGState

Overprint is implemented in all supported drivers. However, there is a minor problem with the (x)dvipdfm(x) family of drivers. The definition of the graphic state must be present in the resources of each page where overprinting is used. The (x)dvipdfm(x) drivers do not do it automatically, it has to be done by the `\OverprintXeTeXExtGState` macro. Since the cropmarks switch overprint off, they require the definition of the graphic state and the macro is always invoked from the running head. This requirement is thus a minor problem. The user usually does not care whether overprint is enabled for preview and proof-reading. The final document will have cropmarks and thus overprint will be enabled.

\SetOverprint
\SetKnockout

These macros change the mode to *overprint* or *knockout*, respectively. They act as declarative macros, similarly as e. g. `\itshape`. You have to use them within a group. The macros are intended to be used for changing the mode for a longer part of text. Due to the way how environments are handled in LaTeX, it is also possible to write:

```
\begin{SetOverprint}
Some long text to be overprinted...
\end{SetOverprint}
```

\textoverprint
\textknockout

If a short part of text should be printed in a different mode, one-argument macros `\textoverprint` and `\textknockout` can be used. They act similarly as the `\textit` macro.

REDEFINEBLACK

Overprinting works only if both the background and forground colors are in the CMYK model. However, the *black* color, which is most often overprinted, is defined in the GRAY model as default. This package defines the `cmykblack` color that may be used for this purpose. In addition if REDEFINEBLACK is set to *true*, the standard *black* color will be redefined in CMYK.

REDEFINETOCMYK

Although *red*, *green*, *blue* will rarely be used for overprinting (maybe just as

a background), option REDEFINETOCMYK requests redefinition of these colors to CMYK. The *black* color will be redefined as well.

Important note: the (x)dvipdfm(x) drivers may switch to the gray colour model after `\textcolor` even if redefinition of black or even all colours to CMYK was requested. If black overprint does not work, insert explicit `\color{cmykblack}`. This trick is not needed with pdfTEX or dvips.

If the colors are redefined to CMYK, the original definitions are not available. Although you redefine them due to a printing process where the RGB colors are undesirable, you can sometimes need them. For this purpose *grblack*, *rgbred*, *rgbgreen*, *rgbblue* colors are always available.

`grblack` `rgbred` `rgbgreen` `rgbblue`

13 PDF information

The package writes the basic information that is required by PDF/X even if PDF/X conformace is not requested. The basic pieces of information are `CreationDate` and `ModDate`. Implementation is driver dependent and some information is supplied by the driver itself. Thus `pdftex` inserts both, (x)dvipdfm(x) (XETEX) adds `CreationDate` only, dvips adds none. The missing information is supplied by this package based on `\date` and `\time`. The next subsections will explain how other fields can be defined and how they can be disabled if needed.

13.1 PDF title

The TITLE option is used to set the title field. The default is to take the contents of the CROPTITLE option even if the cropmarks are switched off. If CROPTITLE is empty, `\jobname` is used. If you specify this option without a value, it has a special meaning of suppressing creation of PDF information field. Since this is not mnemonic, there is a NOPDFINFO option with the same effect, see subsection 13.5.

TITLE

13.2 PDF author

The AUTHOR option sets the author of the document. The field is not required by PDF/X, therefore it has no default value.

AUTHOR

13.3 PDF subject

The SUBJECT option defines the subject of the document.

SUBJECT

13.4 PDF keywords

The KEYWORDS option is used to set the list of keywords. Usually the keywords will be given as a comma separated list. They must therefore be enclosed in braces.

KEYWORDS

13.5 Option NOPDFINFO

The above mentioned fields may also be set in the PDF file by the HYPERREF package. If HYPERREF is used, it may not be desirable if ZWPAGELAYOUT wrote the PDF information. If you specify NOPDFINFO in the list of options, setting all above mentined information (including `ModDate`) will be disabled. This option is not boolean, it just erases the contents of the TITLE option, see subsection 13.1.

NOPDFINFO

Since options are procesed in the order in which they are declared in the package, NOPDFINFO will always erase the PDF title even if it is specified before TITLE.

It has recently been found that these packages do not conflict, it is safe to specify some pieces of information by ZWPAGELAYOUT and other pieces of information via HYPERREF. Anyway, this option will be preserved for the case that it might be needed in the future.

14 PDF/X-1a compliance

The package partially implements the PDF/X-1a standard. Remember that implementation is driver dependent and not everything can be achieved with all drivers. The following sections will give you more detail. Unlike the PDFX package we try to do as much in `xetex` and set the bounding boxes according to the real page size.

14.1 Option PDFMINORVERSION

PDFMINORVERSION

This option allows you to set the version of the PDF file. It works with `pdftex` only. There is no `\special` for sending this information to (x)dvipdfm(x), it can only be set on the command line. Similarly PostScript to PDF converters accept such setting from the command line or a configuration file although there is a PS command for setting it.

`\SetPDFminorversion`

The same effect can also be achieved by `\SetPDFminorversion` from a preamble. The macro accepts a single argument and is implemented for `pdftex` only. It does nothing for other drivers.

14.2 Option PDFX

PDFX

This option ask the package to create PDF/X-1a file. The first requirement is to set PDF version to 1.3, thus it sets the minor version to 3 and then disables macro `\SetPDFminorversion`. Other options have reasonable default values but they can be changed as described in the following subsections.

PDF/X-1a compliance is not handled for `dvips` and some features are not available for `xetex`.

14.3 Options OUTPUTCONDITION and OUTPUTCONDITIONIDENTIFIER, default Euroscale Coated v2

OUTPUTCONDITION
OUTPUTCONDITION-IDENTIFIER

These options specify the ICC profile. I am not sure how to obtain the correct names. The default values correspond to the ICC profiles used in Europe but you can easily set another value. From my personal experience I prefer *eucmyk50* when converting my images from AdobeRGB to CMYK.

14.4 Option ICCFILE

ICCFILE

This option specifies the name of the file containing the ICC profile. There is no default definition and no profile is embedded. The profiles usually accompany commercial products as printers and scanners or can be downloaded from the web. If you have a CMYK profile that you wish to embed, you can specify its name with the optICCfile option. The profile can only be embedded by `pdftex`.

14.5 Font embedding

PDF/X requires all fonts to be embedded. The ZWPAGELAYOUT package cannot ensure it. You have to verify your configuration files and make sure that fonts are embedded.

14.6 Page bounding boxes

It is mandatory to set BleedBox and TrimBox in addittion to MediaBox. Setting these boxes is explained in section 7.3. ArtBox is explicitely forbidden by PDF/x, therefore it is not set.

14.7 PDF information

Mandatory fields are title, CreationDate and ModDate. All these fields are set automatically unless they are disabled as described in section 13.

14.8 MPT metadata

Emdedding MPT metadata is currently not implemented. It will be implemented in the future but only for `pdftex`.

14.9 Color

PDF/X-1a allows only CMYK colors and spot colors. Since ZWPAGELAYOUT does not handle colors itself but makes use of the COLOR package, it cannot verify that only allowed colors are used.

15 Useful macros

The package offers two types of useful macros. The macros from the first group help in designing the document. However, as mentioned in the Introduction, the package may be deployed in already existing document just for adding the crop marks. In order to reduce the risk of conflicts with other packages these macros are unavailable if the ONLYCROPMARKS option is used. The second group contains macros that are primarily used in the crop mark generation. They are always visible.

15.1 Userspace macros

As written above, macros of this group are available only if the ONLYCROPMARKS option was not used. It is not planned to make any interface for providing them even with the ONLYCROPMARKS option. If the document is already finished, you do not need them. If you write a new document, it is preferable to use the whole package for defining the page layout. The macros will than be available.

`\topskip` `\Vcorr`

TEX should use integer arithmetic but some implementations, e. g. emTEX, violate this rule. This makes processing the text faster but may have bad results. If you combine text with boxes and fixed size glues the height of which is an integer multiple of `\baselineskip`, the round-off errors may add a few scaled points, the page overflows and as a result is made one line shorter and underfull vbox is reported. For this reason tiny shrink is added to `\topskip`, its size is 11.00153pt minus 0.00763pt. Yet it may not be sufficient in some cases. We therefore provide vertical skip macro `\Vcorr` the size of which is shrinkable zero. The meaning of `\Vcorr` is `macro:->\vskip 10sp minus 50sp`.

We often need a vertical skip the size of which is a multiple of \baselineskip. Macro \vb serves this very purpose. It accepts an optional argument in square brackets which denotes the multiple of \baselineskip. The default value is 1. The command also contains compensation shrinkage. Some packages activate several characters, even those which could be used in numerical and dimension specifications. Hyphens and dots are temporarily deactivated within the optional argument of the \vb macro. You thus can comfortably specify negative as well as fractional dimensions.

\vb

LaTeX provides \cleardoublepage for moving the next text to the beginning of the odd page. However, you have no control over the page style of the inserted empty even page. This feature is enabled in the \NewOddPage macro. Its syntax is:

\NewOddPage

```
\NewOddPage*[⟨style⟩]
```

Optional parameter ⟨style⟩ defines the style of the empty even page that should be fed to the \thispagestyle command. The default is empty. The starred version displays a warning in the log file if an empty page was inserted.

This macro sets the contents of the message that should be displayed as a warning if an empty page was inserted as a result of \NewOddPage*. The macro acts as \gdef. This means that the definition is global and you can therefore have only one message unless you redefine it. The body of the message may contain macros. They will be expanded when the message is being written. You can use \MessageBreak in order to split the text into several lines.

\SetOddPageMessage

Generally speaking, you should not start a chapter on an even page but there are cases when it is desirable. Imagine the situation when each chapter starts with a full page illustration on the left and with its title page on the right. In such cases you need to start at the even page but LaTeX does not contain any direct tool for doing it. This package provides the \NewEvenPage with the same syntax as \NewOddPage.

\NewEvenPage

Similarly this macro serves for setting the message text that appears if an empty page was inserted as a result of the \NewEvenPage* macro.

\SetEvenPageMessage

15.2 Crop mark macros

TeX offers zero-width horizontal boxes with contents overlapping to the right (\rlap) or to the left (\llap). The zwpagelayout package frequently needs a zero-width box with the contents centered overlapping evenly to both sides. The package thus provides macro \clap and makes it available for the user.

\clap

It was written in the Introduction that the package allows to prepare the book cover before the exact dimensions are known. Later we just adapt the values of the package options and everything is recalculated automatically. As a matter of fact, it could not happen if we did not refer to selected dimensions in the document symbolically. Thus the values of some options are available in macros: \CropFlap = flap, \CropSpine = spine, \CropXSpine = spine, \CropXtrim = xtrim, \CropYtrim = ytrim, \UserWidth = textwidth, \UserLeftMargin = leftmargin, \UserRightMargin = rightmargin.

\CropFlap
\CropSpine
\CropXSpine
\CropXtrim
\CropYtrim
\UserWidth
\UserLeftMargin
\UserRightMargin

The use of these macros can be shown by an example. The book cover with flaps displayed in Figure 2 was prepared by the following code:

```
\input utf8-t1 % conversion from UTF-8 to T1 by encTeX
\documentclass{book}
\usepackage[papersize={,80mm},topmargin=5mm,
```

```
    botmargin,strictheight,
    leftmargin=0mm,flap=40mm,textwidth=50mm,spine=8mm,
    cropmarks,cropframe,croptitle=Cover]{zwpagelayout}
\usepackage[T1]{fontenc}
\usepackage{lmodern,rotating}
\pagestyle{empty}
\parindent 0mm
\def\thePageNumber{Designed by Zdeněk Wagner}
\begin{document}
\small
\hbox to \textwidth{%
  \vbox to \textheight{\hsize \CropFlap \centering
        Back flap\vfill}\hss
  \vbox to \textheight{\hsize \UserWidth \centering
        Back cover\vfill \leavevmode ISBN+EAN}\hss
  \vbox to \textheight{\hsize \CropSpine \centering
        \vfill
        \begin{sideways}Spine\end{sideways}\vfill}\hss
  \vbox to \textheight{\hsize \UserWidth \centering
        Front cover\vfill \LaTeX}\hss
  \vbox to \textheight{\hsize \CropFlap \centering
        Front flap\vfill}}
\end{document}
```

Remember that these macros contain only the dimensions that were specified in the `\usepackage` command, not those that were calculated by the algorithm described in section 9.2. If a shortcut option was used, all real dimensions are defined, i. e. the MARGINS option fills all four margin options, TOPMARGIN, LEFTMARGIN, RIGHTMARGIN, BOTMARGIN, and the TRIM option fills both XTRIM and YTRIM. You can take advantage of it to make your document more general. You can e. g. define the book cover in two variants, with or without flaps, using this trick:

```
\ifcat$\CropFlap$
  % no flaps
\else
  \vbox to \textheight{\hsize\CropFlap
    Flap text
  }\hss
\fi
```

Macro `\thePageNumber` is used to print the page number together with the crop mark text. We have already shownd that it may be suppressed by the NOPAGENUMBERS option. Another possibility is to redefine it. We can e. g. define the book title with the CROPTITLE option and use the text “Cover” instead of the page number when typesetting the cover. Usage of this macro is also demonstrated in Figure 2. `\thePageNumber`

15.3 Driver switching macros

These macros provides conditional branching according to the currently selected driver no matter whether it is chosen automatically or by specifying the DRIVER

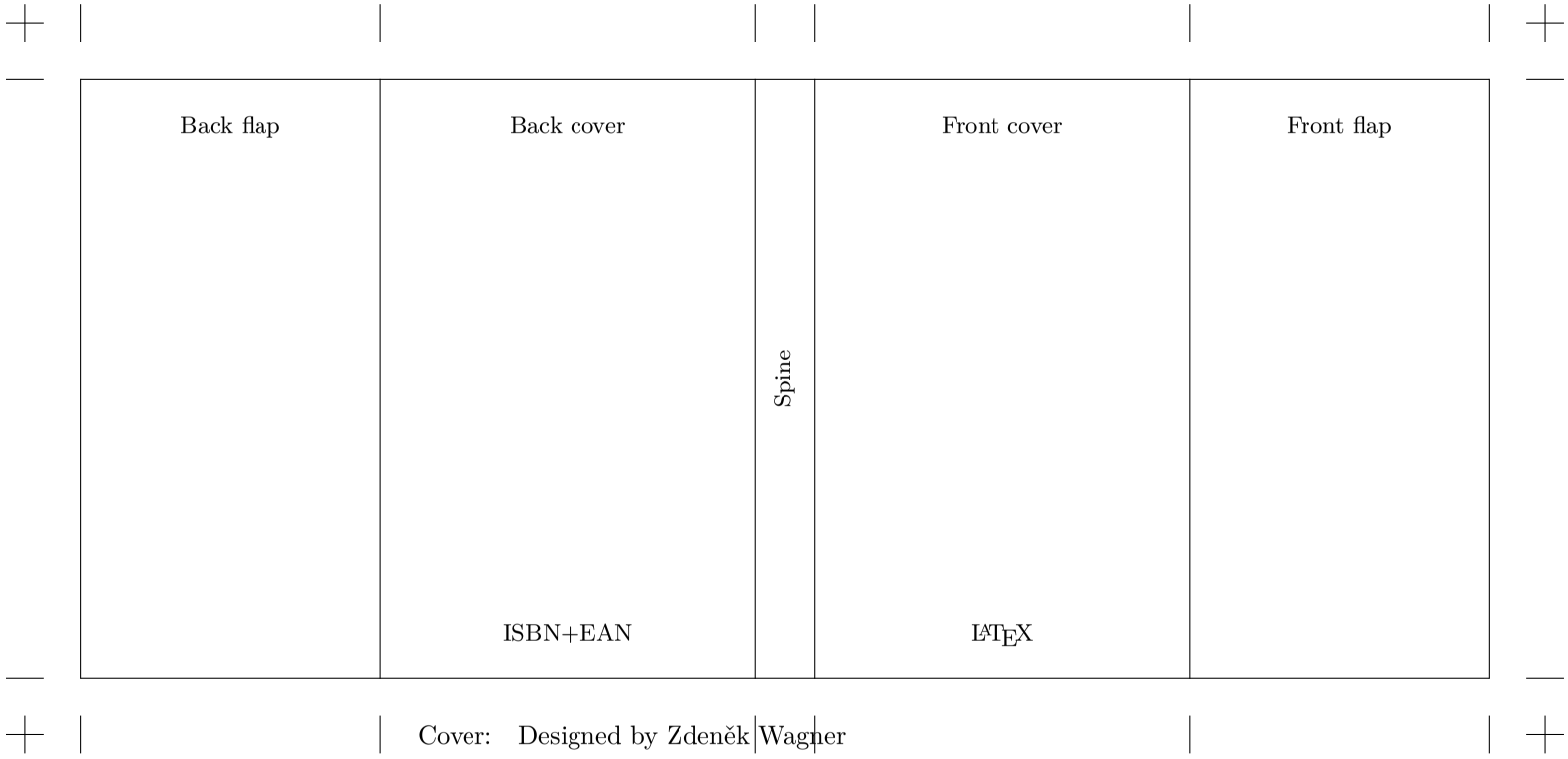

Figure 2: Sample of a book cover with flaps

option described in section 6. The macros do not change page layout, they are therefore available even if the ONLYCROPMARKS option was given.

Macro `\ifcaseZWdriver` expands to an open `\ifcase` primitive. You can thus branch the code according to the driver. The numeric values are 0 for `unknown`, 1 for `pdftex`, 2 for `xetex`, 3 for `dvips`. The branching may look as follows: `\ifcaseZWdriver`

```
\ifcaseZWdriver
  Code for unknown
\or
  Code for pdftex
\or
  Code for xetex
\or
  Code for dvips
\else
  Error, it may not happen!
\fi
```

Since `\ifcaseZWdriver` is a macro, it cannot be used inside another conditional statement. If it falls to the *false* branch, it will not be expanded and TEX will complain that it sees misplaced `\else` etc. You would have to enter all corresponding contidional as `\csname` constructions or define the condition in a separate macro that will be invoked from the condition.

Important note: It is not guaranteed that the numerical assignment will be kept unchanched in future versions.

The `\ZWifdriver` allows conditional execution of a code for a specified driver. It requires two parameters. The first parameter is the driver name, the second parameter is the code to be executed. The macro recognizes all driver names and aliases as given in section 6. It can be used e. g. in cases when you have to perform some action for a specific driver only. Suppose that you do not want to set the page bounding boxes (see 7.3) if `dvips` is used but want to set them if the same document is processed by any other driver. You can thus put the following command to the preamble of the document: `\ZWifdriver`

```
\ZWifdriver{dvips}{\noBboxes}
```

Note: This macro does not depend on the numerical assignment given above. It will therefore work the same way in the future versions.

16 Customizing crop styles

This section is intended for real TEXperts. The package tries to provide a lot of options that enable to customize the default crop style. Try to live with them because here you are touching the very guts of the package. If you squeeze them badly, your document will suffer from strange problems the source of which will be difficult to trace. However, if you like tough challenges and your stomach is strong enough, then read on, but remember, you have been warned.

You ask for a different crop style by the CROPSTYLE option as described in section 11.1.8 on page 13. Assume that you load the package with:

```
\usepackage[cropmarks,cropstyle=special,...]{zwpagelayout}
```

You will then have to define macro `\cropstyle@special`. The package first patches the footer in all page styles. A zero-width, zero-height, zero-depth box is added to the beginning of the footer. The actual point is set to the lower left corner of the paper as defined by `\paperheight` and `\paperwidth`. If NOBLEEDCLIP was not used, the area outside the bleed box is erased. The current color is set according to CROPCOLOR option and the font is selected. Afterwards the `\cropstyle@special` macro is called. `\cropstyle@special`

Some crop styles may require initialization. If it is the case of your style, you have to define macro `\cropstyle@special@setup`. This macro will be executed in the `\AtBeginDocument` hook. Execution of the macro in that hook allows you to defin the crop style in a package that may be loaded after ZWPAGELAYOUT. It is not an error if the setup macro does not exist.

As already noted, the intention is to discourage users from writing their own crop styles. We will therefore only mention that the package defines a few useful macros that can be used in the crop style definition. If your temptation to design your own crop style is really strong, you should better study the package internals yourself.

17 Summary of driver specific features

The driver specific features are described throughout the manual but it may not be clear from the option or macro description that it is driver dependent. These features with the reference to the section, where it is described, are written below. All these features are disabled if an `unknown` driver is being used (section 6).

- Page bounding boxes, section 7.3
- Page reflection, section 10
- Overprinting, section 12.3
- Setting PDF information, section 13
- PDF/X-1a compliance, section 14

Color support is, of course, driver dependent too. However, the ZWPAGELAYOUT has no pretention to deal with it. It is fully handled within the COLOR package.

18 Known bugs and unimplemented features

This section describes features that are known not to work in all case or those that do not work with all drivers.

18.1 Driver repertoire

Only a few most frequent drivers are automatically detected and supported. Their list is presented in section 6. If you use another driver, you should probably disable all driver specific features.

18.2 Shifted cropmarks when the running foot overflows

If there is not enough space for the running foot, the cropmarks will most likely be shifted. It usually happens if you set option FOOTSKIP to zero (which is its default value) and forget to use `\pagestyle{empty}`.

18.3 Page boxes and Adobe Distiller

Setting page boxes causes a fatal error in Adobe Distiller (verified with versions 4 and 9). See section 7.3.

18.4 Page reflection

Option ReflectVertically seems to produce slightly shifted output. If any of these options is used in XeTeX or more specifically with (x)dvipdfm(x) drivers, other `bop` or `eop` code must not be used. See section 10 for details on its implementation.

18.5 Overprinting

Overprinting works in (x)dvipdfm(x) drivers but may be cumbersome in some situations. More details are given in section 12.3.

It was also found that overprinting does not work if the PostScript file is converted to PDF by GhostScript version 7.x. This is a bug in GhostScript, overprinting works fine if GhostScript 8.x is used.

The effect of oveprinting was tested in Adobe Acrobat 9 Pro in MS Windows XP Home Service Pack 3.

18.6 Inconsistent dates

Values of `CreationDate` and `ModDate` (see section 13) supplied by the driver contain information on the time zone. Unfortunatelly, time zone setting is not available in TeX. If both fields come from the same origin, they are set consistently. If one of them is set by the driver and the other by the zwpagelayout package, then only the field set by he driver will contain the time zone value. Depending on your time zone it may look as if the document were modified before it was created.

18.7 PDF/X conformance

The PDF/X conformance is only partial. The details are given in section 14 and its subsections.

19 Changes

This section summarizes the changes. The version and the package date is given. It may be useful to specify the date in the `\usepackage` or `\RequirePackage` command if you rely on a specific feature not available in the old version of the package.

19.1 Version 1.4d, 2020/02/07

- Bug fix, packages ifpdf and ifxetex replaced with iftex.
- Bug fix, identification of PDF/X-1a corrected.
- Feature request, luatex85 loaded in order to support LuaLaTeX compatibility.
- Modification, distribution file flattened as required by CTAN, see section 2.

19.2 Version 1.4c, 2013/01/13

Bug fix, the PDF boxes are properly set even in the (x)dvipdfm(x) family of drivers, i. e. in XeLaTeX.

19.3 Version 1.4b, 2012/10/04

New feature, the color name in the cropmarks can be displayed in white on a colored background, see page 15.

19.4 Version 1.4a, 2012/05/20

Bug fix, if a user requested unexistent page style, the cropmark mechanism looped forever until all main memory was exhausted. Now the package issues an error message and uses the "empty" page style.

19.5 Version 1.4, 2012/05/13

Black overprint is implemented for the (x)dvipdfm(x) family of drivers. It means that it now works in XeLaTeX, see section 12.3.

19.6 Version 1.3a, 2012/01/10

- Bug fix, code rearrangement in order to prevent an error message if ONLYCROPMARKS is used. It leads to the following changes in the functionality:

  1. Page size is always set in the driver specific way. It is still assumed that `\paperheight` and `\paperwidth` contained correct values before the package was loaded.
  2. Bounding boxes are set unless they are explicitly suppressed.
  3. PDF options are honoured. It is therefore possible to request PDF/X compliance, set the PDF title and other information. If you do not wish it, you have to suppress it explicitely by NOPDFINFO.

19.7 Version 1.3, 2011/11/22

- Bug fix, weird errors appeared if the IFXETEX package was loaded previously either directly or indirectly. Loading of both IFPDF and IFXETEX packages written in a different way.

- Bug fix, pagestyle patching redesigned (r. 409); in the previous versions the crop marks may disapear if `\pagestyle` was issued again, mainly when the FANCYHDR package was used.

- New feature, when defining a page style, a part of another style may now be inherited. The following code now can be used:

```
\def\ps@mypagestyle{\ps@plain
    \def\@oddhead{\hfill My running head}\let\@evenhead\@oddhead}
\pagestyle{mypagestyle}
```

- Bug fix: `\globaldefs` is set to zero for printing crop marks. If it were nonzero, crop marks will not work and will spoil the document. LaTeX users probably do not know this primitive but who knows...

- Bug fix: without the TWOSIDE class option and with asymetric margins the crop marks were printed on wrong positions on the even pages.

- New feature: crop marks style for leaflets added (section 11.3).

19.8 Version 1.2, 2011/09/04

- Bug fix, `\fi` was missing in the definition of `\ZWifdriver`.

19.9 Version 1.1, 2010/12/21

Two bugs were fixed and a few new features were introduced. Documentation is modified and extended as well.

- Bug fix, if `\pagestyle` or `\thispagestyle` was issued within a group while crop marks were active, the package often failed into an endless loop which resulted in exhausting the memory.
- Bug fix, the code for page reflection for `dvips` did not preserve properly the old `bop-hook`.
- New feature, the driver can be explicitly selected (section 6) and the code can be branched according to the driver (section 15.3).
- New feature, page reflection implemented for (x)dvipdfm(x) drivers, i. e. for XeTeX (section 10).
- New feature, CMYK and RGB colors explicitly defined (section 12).
- New feature, colors may be redefined to use the CMYK model (section 12).
- New feature, overprinting implemented for `pdftex` and `dvips` (section 12.3).
- New feature, PDF information can be set without the need of loading the HYPERREF package (section 13).
- New feature, partial PDF/X-1a compliance (section 14).

19.10 Version 1.0a, 2008/12/26

The sample files distributed with the package no longer require encTeX and private macro packages.

20 License

The package can be used and distributed according to the LaTeX Project Public License version 1.3 or later the text of which can be found at the `License.txt` file or at `http://www.latex-project.org/lppl.txt`

21 Trade marks

This document makes use of trade marks owned by Adobe Systems Incorporated and Microsoft Corporation when refering to their products.