# The xespotcolor package: Spot Colors for XeLaTeX & LaTeX

Απόστολος Συρόπουλος
(Apostolos Syropoulos)
Xanthi, Greece
asyropoulos@yahoo.com

2016/03/22
updated 2021/03/01

**Abstract**

A spot color is one that is printed with its own ink. Typically, printers use spot colors in the production of books or other printed material. The spotcolor package by Jens Elstner is a first attempt to introduce the use of spot colors with pdfLaTeX. The xespotcolor package is a reimplementation of this package so to be usable with XeLaTeX and LaTeX+dvipdfmx. As such, it has the same user interface and the same capabilities.

## 1 Introduction

Using spot colors with XeLaTeX is very important since most printers use spot colors in the production of books and magazines. The spotcolor package makes it possible to use spot colors with pdfLaTeX but it cannot be used with XeLaTeX or LaTeX. In what follows I first describe how to translate certain pdfTeX code snippets into XeTeX or TeX+dvipdfmx and then I present the code of the package. Thus one can view this text as a short tutorial on how to port pdfTeX code to XeTeX or TeX+dvipdfmx as well as a description of the functionality of the xespotcolor package. Since the package is a port of a pdfTeX package, it has the same functionality as the original package.

## 2 Porting pdfTeX code to XeTeX or TeX+dvipdfmx

Translating pdfTeX code, which adds PDF code to the output file, to XeTeX or TeX+dvipdfmx is not a straightforward exercise since pdfTeX provides primitive commands that directly access and modify the structure of the resulting PDF file. In the case of XeTeX and TeX one has to use \special commands that pass code to the driver. In this particular case, I had to translate code snippets like the following one:

```
1. \newcount\theCNTa
2. \newcount\theCNTb
3. \def\obj{ 0 R}%
4. \pdfobj{Raw PDF code 1}%
5. \theCNTa=\the\pdflastobj%
6. \pdfobj{Raw PDF code \the\theCNTa \obj}%
7. \theCNTb=\the\pdflastobj%
8. \pdfrefobj\theCNTa%
9. \pdfrefobj\theCNTb%
```

Here pdfTeX creates two PDF objects, where the second contains a reference to the first one. The two counters defined in lines 1 and 2 are used to reference these two objects. The macro on line 3 is used to create code that references an object. The

commands on lines 5 and 7 assign the object reference numbers and these numbers are used by the \pdfrefobj primitive. After some experimentation and some… Googling, I have found out that the following XeTeX code is a reasonable translation of the previous code snipper:

```
\newcount\CNT
\newtoks\TOK
\TOK={@TOK \the\CNT}%
\edef\A{\the\TOK Raw PDF code 1}%
\edef\B{Raw PDF code \the\TOK}%
\special{pdf:obj \A}%
\special{pdf:obj @TOKB\the\CNT \B}%
\advance\CNT by1%
```

The two \edefinitions are used to do the work done by \pdfobj. Note that here there I introduce only one unique object and the first two lines define a counter and a token variable. The token variable uses the counter to create a unique identifier, which is passed to the driver. This way the driver will create a number of different objects, if required to do so. The last two commands pass the raw PDF code and the unique identifier to the driver.

The original package contains a definition identical to the following one:

```
\def\R#1{%
   \edef\act{\noexpand\pdfpageresources={\the\pdfpageresources\space
          /ColorSpace<<#1>>}}
   \act}
```

The net effect of this command is to add a specific color space to the page resources of all subsequent pages. Unfortunately, when the following code is executed, it adds the particular color space to the current page only:

```
\def\R#1{%
   \special{pdf:put @resources <</ColorSpace <<#1>>>>}}
```

In order to add the color space to all subsequent pages, I had to use the \AddEverypageHook command of package everypage. This command modifies the contents of the ship-out box by adding to it its argument. And this is done for every single page. Also, one should note how the page resources are augmented by the two systems. In the case of XeTeX we have to create a PDF dictionary that is merged with the current page resources, while in the case of pdfTeX one just "appends" what is supposed to be included in the page resources dictionary. Let me now proceed with the description of the source code of the package.

## 3 The Source Code and Package Usage

The first part of the code is the identification part.

```
1 ⟨*xespotcolor⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{xespotcolor}
4          [2021/03/01 v.2.1, Package for adding Spot Color support to LaTeX/XeLaTeX.]
```

The package needs three packages in order to operate properly: the graphics package, the everypage package, and either the package color or the package xcolor. The later is necessary if one wants to use spot colors with TikZ and other packages that rely on package xcolor. Akira Kakuto has discovered that this package can be used with either LaTeX or XeLaTeX and the dvipdfmx/xdvipdfmx driver, respectively. However, in order to properly load the various packages it is necessary to know which typesetting engine is being used. A simple solution is to use the iftex package:

```
5 \RequirePackage{iftex}
```

Before proceeding it is also necessary to know if the package xcolor has been loaded (package color is loaded by default). The following command check if xcolor is loaded. If it is, it defines a new macro.

```
6 \@ifpackageloaded{xcolor}{\let\use@xcolor\relax}{}
```

Now we are ready to process the three package options: `hks`, `pantone`, and `xcolor`. Here we employ the same "trick": if an option is requested, we define a corresponding macro.

```
\DeclareOption{hks}{\let\use@hks\relax}
\DeclareOption{pantone}{\let\use@pantone\relax}
\DeclareOption{xcolor}{\let\use@xcolor\relax}
\ProcessOptions
```

The code that follows loads the remaining two packages. If macro `\use@xcolor` is not defined, then the package should load the color package; otherwise it should load the xcolor package. The two inner `\if`s are used to load properly the packages when XeTeX or TeX is used to typeset a document. In addition, to avoid unnecessary package loadings, there are some tests to make sure the packages are not loaded.

```
\ifx\use@xcolor\undefined
\ifxetex
   \@ifpackageloaded{graphics}{}{\RequirePackage[xetex]{graphics}}
   \@ifpackageloaded{color}{}\RequirePackage[xetex]{color}
\else
   \@ifpackageloaded{graphics}{}{\RequirePackage[dvipdfmx]{graphics}}
   \@ifpackageloaded{color}{}\RequirePackage{color}
\fi
\else
\ifxetex
   \@ifpackageloaded{graphics}{}{\RequirePackage[xetex]{graphics}}
   \@ifpackageloaded{xcolor}{}{\RequirePackage[xetex]{xcolor}}
\else
   \@ifpackageloaded{graphics}{}{\RequirePackage[dvipdfmx]{graphics}}
   \@ifpackageloaded{xcolor}{}{\RequirePackage{xcolor}}
\fi\fi
```

The following command should be executed only when using TikZ and/or tcolorbox. Thus, always load this package after loading TikZ and/or tcolorbox.

```
\@ifpackageloaded{tikz}{\global\pgf@sys@pdf@colorspaces@existsfalse}{}
\@ifpackageloaded{tcolorbox}{\global\pgf@sys@pdf@colorspaces@existsfalse}{}
```

The `\NewSpotColorSpace` command should be used to define a new color space for spot colors. The new color space can be any imaginable word! This color space is the place where a new spot color will live. The `hks` and the `pantone` options create two color spaces where the corresponding colors live.

```
\def\NewSpotColorSpace#1{%
  \expandafter\newtoks\csname #1\endcsname%
  \csname #1\endcsname{}%
}
```

The `\AddSpotColor` macro should be used to introduce a new spot color. Thus it is one of the first commands on should play with when using this package. This command takes four parameters—the name of a new color space, which has been declared with `\NewSpotColorSpace`, the name of a new color, a name which will be used internally, and a CMYK representation of the new color. For example, here is a typical use of this macro:

```
\AddSpotColor{NEWCS}{NEWCOLOR}{emerald\SpotSpace city}{0.83 0 0.04 30}
```

CMYK tables usually specify the four color components using percentages, which must be "translated" to numbers between 0 and 1.

Macro `\SpotSpace` is used to introduce blanks in names. According to the PDF standard a blank is denoted by the character sequence #20 and this is the reason why `\SpotSpace` is defined as follows:

```
\catcode`\#=12%
\def\SpotSpace{#20}
\catcode`\#=6%
```

The macro `\csgrab` is quite unsual—it takes two arguments and creates a sequence of them.

```
\gdef\csgrab#1#2{#2\expandafter{\the#2 #1}}%
```

Note that this command should never be used by an ordinary package user.

The following variables are used in the definition of macro `\AddSpotColor`. Their functionality has been described in the previous section.

```
\newcount\colorprofilecnt
\newtoks\mycolorprofilename
```

Macro `\AddSpotColor` first defines a new color profile by assigning a value to `\mycolorprofilename`. The name consists of the word `@mycolorprofile` followed by an integer. This name is used in the construction of the corresponding PDF object.

```
\def\AddSpotColor#1#2#3#4{%
   \mycolorprofilename={@mycolorprofile\the\colorprofilecnt}%
```

The following two macros expand to PDF instructions that define the spot color. The PDF instructions are copied verbatim from the original spotcolor package.

```
   \edef\mycolorprofile{\the\mycolorprofilename
           <</C0[0 0 0 0]/FunctionType 2/C1[#4]/Domain[0 1]/N 1>>}%
   \edef\mycolor{[/Separation/#3 /DeviceCMYK \the\mycolorprofilename]}%
```

The next two lines have been copied verbatim from the original macro definition.

```
   \edef\tempcs{/#2 \mycolor}%
   \expandafter\csgrab\expandafter{\tempcs}{\csname #1\endcsname}%
```

In the last part of the macro definition, the driver is instructed to build two objects which should contain the definition of the new spot color. Note that here the macro specifies explicitly the object reference for the second object.

```
   \special{pdf:obj \mycolorprofile}%
   \special{pdf:obj @myowncolor\the\colorprofilecnt \mycolor}%
```

Since all names must be distinct, the macro increments the value of the `\colorprofilecnt` counter by one.

```
   \advance\colorprofilecnt by1%
}
```

Command `\SetPageColorResource` is used by command `\SetPageColorSpace` to set the color space. The `\special` command below sets the page resources only for the current page. Since the color space should be visible to every subsequent page, initially I had opted to use command the `\AddEverypageHook` command provided by the everypage package. However, since the latest version of LaTeX defines hooks, I had to use the new `shipout/background` hook.

```
\def\SetPageColorResource#1{%
   \AddToHook{shipout/background}{\put(1in,-1in){%
                \special{pdf:put @resources <</ColorSpace <<#1>>>>}}}%
}%
\def\SetPageColorSpace#1{%
   \expandafter\SetPageColorResource\expandafter{\the\csname #1\endcsname}%
}%
```

If a user wants to set a spot color as the default color, she should use the `\SpotColor` command:

```
\def\SpotColor#1#2{%
   \special{pdf:literal /#1 cs /#1 CS #2 sc #2 SC}%
   \aftergroup\reset@color%
}%
```

Suppose that a user wants to define a new spot color. Then, she has to issue a command like the following one:

```
\definecolor{Spots}{spotcolor}{SPCOLOR,1.0}
```

Note that the second argument of this command must always be `spotcolor`, since this is the color model we are using. In addition, this command is meaningful only when one has defined a new color space, a new spot color using commands like the following ones:

```
\NewSpotColorSpace{SPCOLORSPACE}
\AddSpotColor{SPCOLORSPACE}{SPCOLOR}{Some\SpotSpace Name}{0.5 1.0 0.51 0}
\SetPageColorSpace{SPCOLORSPACE}
```

Note that we must set the page color space!

Depending on which color package is in use, different low-level commands are executed in order to actually define the new color. These new commands are provided by this package. In particular, when the color package is used, these new commands are the `\color@spotcolor` command and `\c@lor@@spotcolor` command. On the other hand, when using the xcolor package these new commands are the `\XC@mod@spotcolor` command and the `\@@xespotcolordef@xspc` command.

If variable `\use@xcolor` is undefined, this means that the user has opted to use the color package.

```
\ifx\use@xcolor\undefined
```

The code that follows has been taken and subsequently modified from file `xetex.def`, which is *not yet* part of the Standard LaTeX "Graphics Bundle" (see the documentation of the package's source code for more details).

```
\def\color@spotcolor#1#2{\c@lor@@spotcolor#2\@@#1}
\def\c@lor@@spotcolor#1,#2\@@#3{%
   \c@lor@arg{#2}%
   \edef#3{spot #1 #2}%
}
```

The code that follows has been copied and subsequently modified from package xspotcolor. Obviously, the code uses the infrastructure provided by package xcolor to allow users to define new spot colors (see the documentation of the source code of package xcolor for more details).

```
\else
\XC@sdef\XC@mod@spotcolor{spotcolor}
\def\@@xespotcolordef@xspc#1,#2\@@{spot #1 #2}
\let\o@XC@definec@lor@xspc\XC@definec@lor
\def\XC@definec@lor[#1]#2[#3]#4#5%
{%
  \expandafter\ifx\csname XC@mod@#4\endcsname\XC@mod@spotcolor
    \expandafter\xdef\csname\string\color @#2\endcsname
    {\noexpand\xcolor@{}{\@@xespotcolordef@xspc#5\@@}{spotcolor}{#5}}%
   \else
    \o@XC@definec@lor@xspc[#1]{#2}[#3]{#4}{#5}%
  \fi
}%
```

Since TikZ is widely used today for drawing, the following commands are redefined to allow the use of spot colors in pictures created with this package.

```
\gdef\pgfsys@color@spotcolor@stroke#1#2{\special{pdf:literal /#1 CS #2 SC}}
\gdef\pgfsys@color@spotcolor@fill#1#2{\special{pdf:literal /#1 cs #2 sc}}
\gdef\pgfsys@color@spotcolor#1#2{\pgfsys@color@spotcolor@stroke{#1}{#2}%
                                 \pgfsys@color@spotcolor@fill{#1}{#2}}
\fi
```

If the `pantone` or the `hks` option have be specified, then the package loads the corresponding color tables

```
\ifx\use@hks\undefined\else\input{spotcolorhks}\fi
\ifx\use@pantone\undefined\else\input{spotcolorpantone}\fi
```

⟨/xespotcolor⟩

# 4 Simple Usage Example

The code that follows shows how one should use the package to create color drawings using Tikz.

```
% First define a new color space
\NewSpotColorSpace{MyPantone}
% Now define real colors
\AddSpotColor{MyPantone}{Blue}{Spot\SpotSpace Color\SpotSpace Blue}{0.92 0.24 0 0}
\AddSpotColor{MyPantone}{Green}{Spot\SpotSpace Color\SpotSpace Green} {0.65 0 0.96 0}
% Next we need to set the page color space
```

```
\SetPageColorSpace{MyPantone}
% Finally we can define colors!
\definecolor{MyGreen}{spotcolor}{Green,0.5}
\definecolor{MyBlue}{spotcolor}{Blue,0.5}
% And this how we use these colors in drawings.
\begin{tikzpicture}
  \filldraw[color=MyGreen] (0.1,0.1) rectangle (1.9,0.9);
  \draw[color=MyBlue, ultra thick] (0,0) rectangle (2,1);
\end{tikzpicture}
```

## Acknowledgements

I would like to thank Akira Kakuto for his comments and suggestions. Also, I would like to thank иеромонах Пантелеимон (Королев) [hieromonk Panteleimon] for his questions and testing.