The xespotcolor package

Apostolos Syropoulos
Xanthi, Greece
asyropoulos@yahoo.com

2015/03/18

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. 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. In the following I first describe how to translate certain pdfTeX code snippets into XeTeX 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 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

Translating pdfTeX code, which adds PDF code to the output file, to XeTeX 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 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 `\edef`initions 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.

```
⟨*xespotcolor⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{xespotcolor}
          [2015/03/18 v.1.1, Package for adding Spot Color support to XeLaTeX.]
```

The package needs three packages in to operate properly: graphics, color, and everypage. The first two packages must be loaded with the `xetex` option, since auto-detection does not work in this case.

```
\RequirePackage[xetex]{graphics}
\RequirePackage[xetex]{color}
\RequirePackage{everypage}
```

The original package defines two "boolean" variables that are set when the package is loaded with the `hks` and the `pantone` options, respectively. When these variables are true, then the package pre-loads color values in the corresponding color space (i.e., "hks" and "pantone"). These color values are stored in two separate source files. These files are part of the original package distribution.

```
\newif\ifhks\hksfalse
\newif\ifpantone\pantonefalse
\DeclareOption{hks}{\hkstrue}
\DeclareOption{pantone}{\pantonetrue}
\ProcessOptions
```

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}{Name\SpotSpace in\SpotSpace PDF}{0.5 1.0 0.51 0}
```

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, I have opted to use command \AddEverypageHook of the everypage package.

```
\def\SetPageColorResource#1{%
```

```
  \AddEverypageHook{\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%
}%
```

The commands `\color@spotcolor` and `\c@lor@@spotcolor` are "low-level" commands and should never be used by anyone. These commands are used to define new spot colors. They are automatically executed everytime a someone uses a command that defines a new spot color:

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

Note that the second argument must always be `spotcolor`. In addition, this command is meaningful only when one has defined a few things using command like the following ones:

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

The code that follows has been taken and subsequently modified from `xetex.def`, which is *not yet* part of the Standard LaTeX "Graphics Bundle." Note that these command work only if the `xdvipdfm-x` driver has been patched to recognize the `spot` command.

```
\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}%
}
```

If pantone or hks option specified then load corresponding color tables

```
\ifhks\input{spotcolorhks}\fi
\ifpantone\input{spotcolorpantone}\fi
```

⟨/xespotcolor⟩