

# The graphicx-psmin package \*

Hendri Adriaens

<http://stuwwww.uvt.nl/~hendri>

v1.1 (2005/09/20)

## Abstract

This package is an extension of the standard graphics bundle [1] and provides a way to include repeated PostScript graphics (ps, eps) only once in a PostScript document. This provides a way to get smaller PostScript documents when having, for instance, a logo on every page. This package only works when post-processed with dvips, *which should at least have version 5.95b!* The difference for the pdf file is minimal (as Ghostscript already includes only a single copy of graphics files).

## Contents

<b>1 Introduction</b>	<b>1</b>	<b>Acknowledgements</b>	<b>8</b>
<b>2 Using the package</b>	<b>2</b>	<b>Version history</b>	<b>8</b>
<b>3 Implementation</b>	<b>3</b>	<b>Index</b>	<b>8</b>
<b>References</b>	<b>7</b>		

## 1 Introduction

Including a PostScript graphic only once in a PostScript document has been a hot issue for a long time in the  $\TeX$  world. It pops up regularly at many newsgroups, but never a fully satisfying answer was supplied. Some answers vaguely describe some `\special` commands, but do not give the details, some suggest to use a `\TeX` box (which does prevent  $\TeX$  from reading the graphic many times, but still includes it many times in the PostScript) and some do suggest to modify the PostScript file manually, which is far too cumbersome. This package is an extension of the standard graphics bundle [1] and supplies an automated method to do this job.

The technique used by this package has been described by Thomas Greer [2] (see for more details [3]). The implementation in  $\TeX$  boils down to the following steps.

1. Scan the graphics file for the bounding box.
2. Wrap the graphics file in a PostScript function that defines it as an object which can be used multiple times. This function needs the bounding box.
3. Write the entire code to the header of the PostScript file, via the dvi file, using `\specials`.

---

\* This package can be downloaded from the CTAN mirrors: `/macros/latex/contrib/graphicx-psmin`. See `graphicx-psmin.dtx` for information on installing `graphicx-psmin` into your  $\TeX$  or  $\LaTeX$  distribution and for the license of this package.

This will load the graphic in the header of the PostScript file. When reusing the graphic, there is another PostScript function to be inserted which will load the graphic from the graphic object.

In this scheme, step 2 is the hardest one. The first implementation of this method in this package attempted to load the entire graphics file, line by line (which is slow), in a macro, add the necessary PostScript code and write the content of the macro to the dvi using `\special{!...}`. Unfortunately, this didn't work as dvips applied its own line breaking mechanism to the content without taking into account manual line breaks. This meant that the content of the graphics file was modified when it arrived in the PostScript document, often too much to be processed successfully again.

The second attempt in this package wrote the entire graphics file to hard disk, together with the PostScript function so that it could be loaded using `\special{header=...}`. Besides still being slow, this also meant a lot of hard disk overhead.

After some discussion with Karl Berry a proposal for extending the header special was posted on the T<sub>E</sub>X-k mailinglist. The alternative of changing the line breaking mechanism of dvips was not considered as that could change the behavior in existing documents. The proposal was to extend the header special by allowing

```
\special{header={some file.ps} pre={pre code} post={post code}}
```

The package uses the two extra parameters in this special to pass the PostScript function that will make an object of the graphics file in `some file.ps` to the PostScript file. The header itself still copies the file directly into the PostScript file, which avoids needing to read the entire file with T<sub>E</sub>X.

As mentioned in the abstract, this only works when post-processed with dvips, *version v5.95b or newer!* It will, in general, not make a significant difference in the size of your pdf file generated by ps2pdf (Ghostscript).<sup>1</sup> However, it will decrease the size of your PostScript file if you use the same graphic over and over in your document.

## 2 Using the package

Using the package is very simple. Replace the loading of the graphicx package, like

```
\usepackage[your options]{graphicx}
```

by

```
\usepackage[your options]{graphicx-psmin}
```

This will load the graphicx package internally, pass on all of the options specified in your `options` to the graphicx package (for instance, `draft`, `dvips`) and make all definitions necessary to do the job.

```
\loadgraphics[<bb>]{list of graphics}
```

`\loadgraphics`

Each graphic that you want to reuse in the document, but which you want to be included only once, should be listed in the command `\loadgraphics`. This command can only be used *in the preamble* of your document. Example:

```
\loadgraphics{mylogo1.ps,mylogo2.eps}
```

<sup>1</sup>As Ghostscript already embeds graphics with size larger than `MaxInlineImageSize` only once in the pdf.

If a graphics file does not contain a bounding box, you can include that in the optional argument. This argument will be used for every graphic in the list. Example

```
\loadgraphics[0 0 250 420]{mylogo1.ps,mylogo2.eps}
\loadgraphics{mylogo3.ps}
```

`\includegraphics`

That's all! You can use the usual `graphicx \includegraphics` command to include and scale or rotate your graphics. This command will pick up the loaded graphics and use a PostScript function to insert the graphic from the PostScript header. If the graphic has not been loaded with `\loadgraphic`, the `\includegraphics` command works as usual (see also the `graphicx` documentation [1] or a  $\text{\LaTeX}$  manual like [4]).

If the graphic does not contain bounding box information, then this should be included in the `\loadgraphics` command as described above. You can include it in the `\includegraphics` command as well, but if the two bounding boxes are not equal, the result might be unexpected.

As mentioned before, the package only works with the `dvips` driver, version 5.95b or newer. When running  $\text{\LaTeX}$  (not `pdf\text{\LaTeX}`), the `dvips` driver will be loaded automatically by `graphicx`, but you can also specify it yourself when loading the `graphicx-psmin` package. Any options that you specify will be passed on to the `graphicx` package. When `graphicx-psmin` finds out that you are using another driver than `dvips`, it will generate an error message and disable itself, after which the `\loadgraphics` command has no function anymore and all graphics inclusion jobs are left to the `graphicx` package.

To finish with, here is another example demonstrating the use of the package. The example assumes that `figure.ps` is present in the `figures` subdirectory and that `graphic.eps` is in the `graphics` subdirectory.

```
\documentclass{article}
\usepackage{graphicx-psmin}
\graphicspath{{figures/}{graphics/}}
\loadgraphics{figure.ps,graphic.eps}
\begin{document}
\includegraphics[scale=.2]{figure.ps}
\includegraphics[scale=.4]{figure.ps}
\includegraphics[angle=45]{graphic.eps}
\includegraphics[angle=90]{graphic.eps}
\end{document}
```

When running a file like the one above and generating the PostScript using `dvips`, one will see that both graphics are included only once in the PostScript document and that including one of those graphics another time, has almost no effect on the size of the generated PostScript document.

### 3 Implementation

```
1% Initialize.
2<*graphicx-psmin)
3\NeedsTeXFormat{LaTeX2e}[1995/12/01]
4\ProvidesPackage{graphicx-psmin}
5 [2005/09/20 v1.1 single PostScript graphics inclusion (HA)]
6\DeclareOption*{\PassOptionsToPackage\CurrentOption{graphicx}}
7\ProcessOptions\relax
8\RequirePackage{graphicx}
```

Check the requested driver.

```

9 \def\gxpsm@tempa{dvips.def}
10 \ifx\Gin@driver\gxpsm@tempa\else
11   \PackageError{graphicx-psmin}{This package cannot be used with any
12     \MessageBreak back end driver other than dvips!}\@ehd
13   \def\loadgraphics{\@testopt\gxpsm@loadgraphics{}}
14   \def\gxpsm@loadgraphics[#1]#2{}
15   \expandafter\endinput
16 \fi

```

In draft mode, no graphics should be loaded, so we eat the argument and hence all graphics will be handled by graphicx from here.

```

17 \ifGin@draft
18   \def\loadgraphics{\@testopt\gxpsm@loadgraphics{}}
19   \def\gxpsm@loadgraphics[#1]#2{}
20   \expandafter\endinput
21 \fi

```

`\gxpsm@loaded` Initialize the list of loaded graphics.  
`22 \def\gxpsm@loaded{}`

`\@namexdef` `{\langle csname \rangle}`  
 Defines the macro with name `\langle csname \rangle` using `\xdef`.  
`23 \def\@namexdef#1{\expandafter\xdef\csname#1\endcsname}`

`\loadgraphics` Load graphics.  
`24 \def\loadgraphics{\@testopt\gxpsm@loadgraphics{}}`

`\gxpsm@loadgraphics` `[\langle bb \rangle]{\langle list of graphics \rangle}`  
 Load each of the graphics in `\langle list of graphics \rangle` into the dvi and eventually the PostScript using `\specials`. Use `\langle bb \rangle` for the bounding box for all these graphics if specified. If not, we search the file for the bounding box.

```
25 \def\gxpsm@loadgraphics[#1]#2{%
```

Loop over all graphics.

```
26   \@for\gxpsm@file:=#2\do{%
27     \begingroup
```

If the file exists in the graphics path, continue.

```
28     \gxpsm@checkfile\gxpsm@file{%
```

If no explicit bounding box in #1, try finding one in the file, otherwise use the one in #1.

```
29     \ifx\@empty#1\@empty
30       \Gread@eps{\Gin@base\Gin@ext}%
31     \else
32       \Gread@parse@bb#1 \
33     \fi
```

Save the bounding box for this graphic.

```
34     \@namexdef{\Gin@base\Gin@ext @llx}{\Gin@llx}%
35     \@namexdef{\Gin@base\Gin@ext @lly}{\Gin@lly}%
36     \@namexdef{\Gin@base\Gin@ext @urx}{\Gin@urx}%
37     \@namexdef{\Gin@base\Gin@ext @ury}{\Gin@ury}%
```

Transform the PostScript internal name of the graphic as `/` can't be used in variable names.

```
38     \gxpsm@getcfile
```

Write the graphic body together with the extra functions to the dvi file using a header special. This requires dvips v5.95b or newer to work!

```

39     \AtBeginDvi{\special{header={\Gin@base\Gin@ext}
40     pre={/\gxpsm@cfile-data^^Jcurrentfile^^J%
41         << /Filter /SubFileDecode^^J/DecodeParms << /EOdCount 0
42         /EOdString (*HA-EOD-??3.1416926!!*) >>^^J>>
43         /ReusableStreamDecode filter^^J%
44         \@percentchar\@percentchar BeginDocument:
45         \Gin@base\Gin@ext^^J%
46     }
47     post={\@percentchar\@percentchar EndDocument^^J%
48         *HA-EOD-??3.1416926!!*^^Jdef^^J/\gxpsm@cfile-form^^J%
49         << /FormType 1^^J/BBox
50         [\Gin@llx\space\Gin@lly\space\Gin@urx\space\Gin@ury]^^J%
51         /Matrix [1 0 0 1 0 0]^^J/PaintProc^^J{ pop^^J%
52         /ostate save def^^J/showpage {} def^^J%
53         /setpagedevice /pop load def^^J%
54         \gxpsm@cfile-data 0 setfileposition
55         \gxpsm@cfile-data cvx exec^^J%
56         ostate restore^^J} bind^^J>> def%
57     }
58     }}%

```

Add the file to the list of loaded graphics for `\includegraphics`.

```

59     \xdef\gxpsm@loaded{%
60         \gxpsm@loaded\ifx\gxpsm@loaded\@empty\else,\fi
61         \Gin@base\Gin@ext
62     }%
63     }%
64     \endgroup
65     }%
66 }

```

Avoid using `\loadgraphics` outside the preamble.

```

67 \@onlypreamble\loadgraphics
68 \@onlypreamble\gxpsm@loadgraphics

```

`\gxpsm@getcfile`    These two macros replace any occurrence of `'/'` by `'_'` so that the name can be used  
`\gxpsm@g@tcfile`    inside PostScript. This uses a well known `\lowercase` trick.

```

69 \def\gxpsm@getcfile{%
70     \edef\gxpsm@tempa{%
71         \noexpand\gxpsm@g@tcfile\Gin@base\Gin@ext\noexpand\@nil
72     }%
73     \gxpsm@tempa
74 }
75 \def\gxpsm@g@tcfile#1\@nil{%
76     \begingroup\lccode'\/'\_ \lowercase{\endgroup\def\gxpsm@cfile{#1}}%
77 }

```

`\Gin@base\Gin@ext`    `{\file}`

We redefine this internal of graphics. If the graphic has been loaded, use a PostScript function to reload it from the header. Else, follow the usual track of `graphicx`.

```

78 \def\Gin@base\Gin@ext#1{%
79     \begingroup

```

Graphic exists? This will also produce `\Gin@base` and `\Gin@ext`.

```
80 \gxpsm@checkfile{#1}{%
```

Graphic loaded?

```
81 \expandtwoargs\in@{,\Gin@base\Gin@ext,}{,\gxpsm@loaded,}%
82 \ifin@
```

If the user didn't supply a bounding box in the `\includegraphics` command, use the one that we found while scanning the graphic.

```
83 \ifGin@bbox\else
84 \Gin@bboxtrue
85 \edef\Gin@llx{\@nameuse{\Gin@base\Gin@ext @llx}}%
86 \edef\Gin@lly{\@nameuse{\Gin@base\Gin@ext @lly}}%
87 \edef\Gin@urx{\@nameuse{\Gin@base\Gin@ext @urx}}%
88 \edef\Gin@ury{\@nameuse{\Gin@base\Gin@ext @ury}}%
89 \fi
```

Use graphics internals to do computations etcetera and in the end, use the graphic.

```
90 \Gin@setfile{psdirect}{\Gin@base\Gin@ext}%
91 \else
```

This is the usual route from `\Gin@include@graphics` from `graphics` for non-loaded graphics.

```
92 \ifundefined{Gin@rule@\Gin@ext}{%
93 \ifx\Gin@rule@*\@undefined
94 \@latex@error{Unknown graphics extension: \Gin@ext}\@ehc
95 \else
96 \expandafter\Gin@setfile\Gin@rule@*\Gin@base\Gin@ext}%
97 \fi
98 }{%
99 \expandafter\expandafter\expandafter\Gin@setfile
100 \csname Gin@rule@\Gin@ext\endcsname{\Gin@base\Gin@ext}%
101 }%
102 \fi
103 }%
104 \endgroup
105 }
```

```
\gxpsm@checkfile {<file>}{<actions>}
```

This is part of `graphics`' `\Gin@include@graphics` which checks a graphic file in the `graphics` path. We perform `<actions>` when the file is all right. We separated this part as it is reused several times in this package.

```
106 \def\gxpsm@checkfile#1#2{%
107 \let\input@path\Gin@input@path
108 \filename@parse{#1}%
109 \ifx\filename@ext\relax
110 \@for\Gin@temp:=\Gin@extensions\do{%
111 \ifx\Gin@ext\relax
112 \Gin@getbase\Gin@temp
113 \fi
114 }%
115 \else
116 \Gin@getbase{\Gin@sepdefault\filename@ext}%
117 \ifx\Gin@ext\relax
118 \@warning{File '#1' not found}%
```

```

119     \def\Gin@base{\filename@area\filename@base}%
120     \edef\Gin@ext{\Gin@sepdefault\filename@ext}%
121     \fi
122     \fi
123     \ifx\Gin@ext\relax
124     \latex@error{File ‘#1’ not found}%
125     {I could not locate the file with any of these extensions:^^J%
126     \Gin@extensions^^J\@ehc}%
127     \else#2\fi
128 }

```

`\Gininclude@psdirect`  $\langle file \rangle$

This inserts the PostScript function needed to reload  $\langle file \rangle$  from the PostScript header. This is based on `\Gininclude@eps` from `dvips.def` (graphics).

```

129 \def\Gininclude@psdirect#1{%
130   \message{<#1>}%
131   \bgroup
132   \def\@tempa{!}%
133   \gxpsm@getcfile
134   \dimen@\Gin@req@width
135   \dimen@ii.1bp%
136   \divide\dimen@\dimen@ii
137   \@tempdima\Gin@req@height
138   \divide\@tempdima\dimen@ii
139   \special{ps:@beginspecial
140     \Gin@llx\space @llx \Gin@lly\space @lly
141     \Gin@urx\space @urx \Gin@ury\space @ury
142     \ifx\Gin@scalex\@tempa\else\number\dimen@\space @rwi\fi
143     \ifx\Gin@scaley\@tempa\else\space\number\@tempdima\space @rhi\fi
144     \ifGin@clip\space @clip\fi\space @setspecial^^J
145     save \gxpsm@cfile-form execform restore showpage @endspecial
146   }%
147   \egroup
148 }

149 %</graphicx-psmin>

```

## References

- [1] David Carlisle. graphics bundle. CTAN:/macros/latex/required/graphics.
- [2] Thomas Greer. Reusable content caching in postscript. [http://www.tgreer.com/eps\\_vdp2.html](http://www.tgreer.com/eps_vdp2.html).
- [3] Adobe Systems Incorporated. Postscript language reference manual. <http://www.adobe.com/products/postscript/pdfs/PLRM.pdf>.
- [4] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The L<sup>A</sup>T<sub>E</sub>X Companion, Second Edition*. Addison-Wesley, 2004.

## Acknowledgements

The author is grateful to Thomas Greer and Uwe Kern for help and suggestions. Many thanks to Akira Kakuto for providing a dvips patch which makes this package possible. The author is greatly indebted to Karl Berry for support and for providing a test environment for the dvips patch on the TUG server.

## Version history

<b>v1.0</b>	<i>(2005/08/18)</i>
General: Initial release .....	1
<b>v1.1</b>	<i>(2005/09/20)</i>
\Ginclude@psdirect: Added missing \space .....	7

## 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.

<b>Symbols</b>	
\@namexdef .....	<u>23</u>
<b>G</b>	
\Ginclude@graphics .....	<u>78</u>
\Ginclude@psdirect .....	<u>129</u>
\gxpsm@checkfile .....	<u>106</u>
\gxpsm@g@tcfile .....	<u>69</u>
\gxpsm@getcfile .....	<u>69</u>
\gxpsm@loaded .....	<u>22</u>
\gxpsm@loadgraphics .....	<u>25</u>
<b>I</b>	
\includegraphics .....	3
<b>L</b>	
\loadgraphics .....	2, <u>24</u>