PSTOEDIT

Dr. Wolfgang Glunz

November 21, 2026
Version 4.3

Abstract

pstoedit - a tool converting PostScript files into various vector graphic formats

1 Synopsis

1.1 From a command shell

pstoedit [**-v -help**]

pstoedit The following options are available: [**-include** *filename*] [**-df** *font name*] [**-nomaptoisolatin1**] [**-dis**] [**-pngimage** *filename*] [**-q**] [**-nq**] [**-nc**] [**-mergelines**] [**-filledrecttostroke**] [**-mergetext**] [**-dt**] [**-adt**] [**-ndt**] [**-dgbm**] [**-correctdefinefont**] [**-pti**] [**-pta**] [**-xscale** *number*] [**-yscale** *number*] [**-xshift** *number*] [**-yshift** *number*] [**-centered**] [**-minlinewidth** *number*] [**-pagenumberformat** *page number format specification*] [**-split**] [**-v**] [**-vl**] [**-usebbfrominput**] [**-ssp**] [**-sfill**] [**-uchar** *character*] [**-nb**] [**-rdb**] [**-page** *page number*] [**-flat** *flatness factor*] [**-sclip**] [**-ups**] [**-usefinddevice**] [**-rgb**] [**-useagl**] [**-noclip**] [**-t2fontsast1**] [**-keep**] [**-debugfonthandling**] [**-gstest**] [**-fakedateandversion**] [**-nfr**] [**-glyphs**] [**-useoldnormalization**] [**-rotate** *angle (0-360)*] [**-fontmap** *name of font map file for pstoedit*] [**-pagesize** *page format*] [**-help**] [**-bo**] [**-psarg** *argument string*] [**-pslanguagelevel** *PostScript Language Level 1, 2, or 3 to be used.*] **-f** *"format[:options]"* [**-gs** *either full path to the Ghostscript executable/DLL or - for Windows - just a version number (e.g. 10.01.0), in which case the version is used to look up the path from the registry.*] [**-gsregbase** *Ghostscript base registry path*] [inputfile [outputfile]]

1.2 From PstoeditQtGui

PstoeditQtGui provides an alternative to the command driven operation. The GUI provides access to almost all options and features that are supported by *pstoedit*. In addition it supports the conversion of multiple files in one job and also provides some shortcuts to some of *Ghostscript*'s high level output devices.

The GUI is implemented using QT (`https://www.qt.io`).

The various options provided by *pstoedit* are displayed in different tabs in the GUI according to their category. A link to a more detailed description of each option into this manual is provided with each option in the GUI.

Side note: None of the options are "hard coded" into the GUI. Instead the meta information is retrieved from *pstoedit* itself. By this the GUI is always up to date with respect to the options and formats provided by *pstoedit*.

2 Description

2.1 RELEASE LEVEL

This man-page documents release 4.3 of *pstoedit*.

2.2 USE

pstoedit converts PostScript files into various vector graphics formats. The resulting files can be edited or imported into various drawing packages. Type

pstoedit -help

for a list of supported output formats. Pstoedit comes with a large set of format drivers built into in the binary.

2.3 PRINCIPLE OF CONVERSION

pstoedit works by redefining some of PostScript's basic drawing operators, such as **stroke** or **show** (bitmaps drawn by the image operator are not supported by all output formats.) After redefining these operators, the PostScript file that needs to be converted is processed by a PostScript interpreter, e.g., Ghostscript (*gs*(1)). You normally need to have a PostScript interpreter installed in order to use this program. However, you can perform some "back end only" processing of files following the conventions of the pstoedit intermediate format by specifying the **-bo** option. See "Available formats and their specific options" below.

The output that is written by the interpreter due to the redefinition of the drawing operators is a kind of 'flat' PostScript file containing only simple operations such as moveto, lineto, show, etc. You can view this file using the **-f debug** option.

This output is read by the end-processing functions of *pstoedit* and triggers the drawing functions in the selected output format driver sometimes called also "back-end".

2.4 NOTES ON GHOSTSCRIPT

Although pstoedit was designed to allow the use of any kind of PostScript interpreter, it has only been tested in combination with *Ghostscript* (`https://ghostscript.com`).

Up to version 9.55 of *Ghostscript*, it's PDF interpreter was implemented in PostScript itself. That allowed *pstoedit* to handle PDF files in the same way as PostScript files since the same mechanisms for intercepting the drawing operations could be used.

However, from version 9.56 on, the PDF interpreter of *Ghostscript* was implemented in C and hence the interceptions used by pstoedit are no longer effective when processing PDF files.

You need to convert the PDF to PostScript at first before you can convert it into another format. You can use *Ghostscript* for that or also use the gs:ps2write (**-f gs:ps2write**) driver from *pstoedit*.

3 Options

3.1 General options

The following options are available:

[**-include** ***filename***] This option allows specifying an additional PostScript file that will be executed just before the normal input is read. This is helpful for including specific page settings or for disabling potentially unsafe PostScript operators, e.g., file, renamefile, or deletefile.

[**-xscale** ***number***] scale by a factor in x-direction

[**-yscale** ***number***] scale by a factor in y-direction

[**-xshift** ***number***] shift image in x-direction

[**-yshift** ***number***] shift image in y-direction

[**-centered**] center image before scaling or shifting

[**-minlinewidth** ***number***] minimal line width. All lines thinner than this will be drawn in this line width - especially zero-width lines

[**-pagenumberformat** ***page number format specification***] format specification for page numbers in filename if -split is used. The specification is used to create the page number using sprintf. The specification shall not include the leading % nor the trailing d. Default is empty string which results in formatting the page number using %d. This results in page numbers like 1, 2, ..., 10. Sometimes you may want to have fixed length with leading 0, so you might want to specify 02 which means 2 digits with leading 0.

[**-split**] Create a new file for each page of the input. For this the output filename must contain a %d which is replaced with the current page number. This option is automatically switched on for output formats that do not support multiple pages within one file, e.g. fig or gnuplot.

[**-usebbfrominput**] If specified, pstoedit uses the BoundingBox as is (hopefully) found in the input file instead of one that is calculated by its own.

[**-page** ***page number***] Select a single page from a multi-page PostScript or PDF file.

[**-rgb**] Since version 3.30 pstoedit uses the CMYK colors internally. The -rgb option turns on the old behavior to use RGB values.

[**-useagl**] use Adobe Glyph List instead of the ISO Latin-1 table (this is experimental)

[**-noclip**] do not use clipping (relevant only if output format supports clipping at all)

[**-rotate** ***angle (0-360)***] Rotate image by angle.

[**-pagesize** ***page format***] set page size for output medium. This option sets the page size for the output medium. Currently this is just used by the libplot output format driver, but might be used by other output format drivers in future. The page size is specified in terms of the usual page size names, e.g. letter or a4.

[**-help**] show the help information

[**-bo**] You can run backend processing only (without the PostScript interpreter frontend) by first running **pstoedit -f dump** *infile dumpfile* and then running **pstoedit -f** *format* **-bo** *dumpfile outfile*.

[**-psarg** ***argument string***] The string given with this option is passed directly to Ghostscript when Ghostscript is called to process the PostScript file for *pstoedit*. For example: **-psarg "-r300x300"**. This causes the resolution to be changed to 300x300 dpi. (With older versions of Ghostscript, changing the resolution this way has an effect only if the **-dis** option is given.) If you want to pass multiple options to Ghostscript you can use multiple -psarg options **-psarg opt1 -psarg opt2 -psarg opt2**. See the Ghostscript manual for other possible options.

[**-pslanguagelevel** ***PostScript Language Level 1, 2, or 3 to be used.***] PostScript Language Level 1, 2, or 3 to be used. You can switch Ghostscript into PostScript Level 1 only mode by **-pslanguagelevel 1**. This can be useful for example if the PostScript file to be converted uses some Level 2 specific custom color models that are not supported by pstoedit. However, this requires that the PostScript program checks for the PostScript level supported by the interpreter and "acts" accordingly. The default language level is 3.

-f ***"format[:options]"*** target output format recognized by *pstoedit*. Since other format drivers can be loaded dynamically, type `pstoedit -help` to get a full list of formats. See "Available formats and their specific options" below for an explanation of the [*:options*] to **-f** format. If the format option is not given, pstoedit tries to guess the target format from the suffix of the output filename. However, in a lot of cases, this is not a unique mapping and hence pstoedit demands the **-f** option.

[**-gs** ***either full path to the Ghostscript executable/DLL or - for Windows - just a version num*** tells pstoedit which Ghostscript executable/DLL to use - overwrites the internal search heuristic

[**-gsregbase** ***Ghostscript base registry path***] registry path to use as a base path when searching Ghostscript interpreter. This option provides means to specify a registry key under HKLM/Software where to search for GS interpreter key, version and `GS_DLL / GS_LIB` values. Example: "-gsregbase MyCompany" means that HKLM/Software/MyCompany/GPL Ghostscript would be searched instead of HKLM/Software/GPL Ghostscript.

3.2 Text and font handling related options

The following options are available:

[**-df *font name***] Sometimes fonts embedded in a PostScript program do not have a fontname. For example, this happens in PostScript files generated by *dvips*(1). In such a case *pstoedit* uses a replacement font. The default for this is Courier. Another font can be specified using the **-df** option. **-df Helvetica** causes all unnamed fonts to be replaced by Helvetica.

[**-nomaptoisolatin1**] Normally *pstoedit* maps all character codes to the ones defined by the ISO Latin1 encoding. If you specify **-nomaptoisolatin1** then the encoding from the input PostScript is passed unchanged to the output. This may result in strange text output but on the other hand may be the only way to get some fonts converted appropriately. Try what fits best to your concrete case.

[**-pngimage *filename***] for debugging purpose mainly. Write result of processing also to a PNG file

[**-dt**] draw text. Text is drawn as polygons. This might produce a large output file. This option is automatically switched on if the selected output format does not support text, e.g. *gnuplot*(1).

[**-adt**] automatic draw text. This option turns on the **-dt** option selectively for fonts that seem to be no normal text fonts, e.g. Symbol.

[**-ndt**] never draw text. Fully disable the heuristics used by pstoedit to decide when to "draw" text instead of showing it as text. This may produce incorrect results, but in some cases it might nevertheless be useful. "Use at own risk".

[**-dgbm**] experimental - draw also bitmaps generated by fonts/glyphs

[**-correctdefinefont**] Some PostScript files, e.g. such as generated by ChemDraw, use the PostScript definefont operator in a way that is incompatible with pstoedit's assumptions. The new font is defined by copying an old font without changing the FontName of the new font. When this option is applied, some "patches" are done after a definefont in order to make it again compatible with pstoedit's assumptions. This option is not enabled by default, since it may break other PostScript files. It is tested only with ChemDraw generated files.

[**-pti**] precision text. Normally a text string is drawn as it occurs in the input file. However, in some situations, this might produce wrongly positioned characters. This is due to limitations in most output formats of pstoedit. They cannot represent text with arbitrary inter-letter spacing which is easily possible in PDF and PostScript. With **-pta**, each character of a text string is placed separately. With **-pti**, this is done only in cases when there is a non zero inter-letter spacing. The downside of "precision text" is a bigger file size and hard to edit text.

[**-pta**] see -pti

[**-uchar *character***] Sometimes pstoedit cannot map a character from the encoding used by the PostScript file to the font encoding of the target format. In this case pstoedit replaces the input character by a special character

in order to show all the places that could not be mapped correctly. The default for this is a "#". Using the **-uchar** option it is possible to specify another character to be used instead. If you want to use a space, use -uchar " ".

[**-t2fontsast1**] Handle Type 2 fonts same as Type 1. Type 2 fonts sometimes occur as embedded fonts within PDF files. In the default mode, text using such fonts is drawn as polygons since pstoedit assumes that such a font is not available on the user's machine. If this option is set, pstoedit assumes that the internal encoding follows the same as for a standard font and generates normal text output. This assumption may not be true in all cases. But it is nearly impossible for pstoedit to verify this assumption - it would have to do a sort of OCR.

[**-nfr**] In normal mode pstoedit replaces bitmap fonts with a font as defined by the **-df** option. This is done, because most output formats cannot handle such fonts. This behavior can be switched off using the **-nfr** option but then it strongly depends on the application reading the generated file whether the file is usable and correctly interpreted or not. Any problems are then out of control of pstoedit.

[**-glyphs**] pass glyph names to the output format driver. So far no output format driver really uses the glyph names, so this does not have any effect at the moment. It is a preparation for future work.

[**-useoldnormalization**] Just use this option in case the new heuristic introduced in 3.5 does not produce correct results - however, this normalization of font encoding will always be a best-effort approach since there is no real general solution to it with reasonable effort

[**-fontmap** ***name of font map file for pstoedit***] The font map is a simple text file containing lines in the following format:

`document_font_name  target_font_name`
Lines beginning with `%` are considered comments.
For font names with spaces use the `"font name with spaces"` notation.

If a target_font_name starts with /, it is regarded as alias to a former entry.

Each font name found in the document is checked against this mapping and if there is a corresponding entry, the new name is used for the output.

If the **-fontmap** option is not specified, *pstoedit* automatically looks for the file *drivername*.fmp in the installation directory and uses that file as a default fontmap file if available. The installation directory is:

- MS Windows: The same directory where the *pstoedit* executable is located
- Unix:
  The default installation directory. If it fails, then <*The directory where the pstoedit executable is located*>`/../lib/`

The mpost.fmp in the misc directory of the pstoedit distribution is a sample map file with mappings from over 5000 PostScript font names to their TEX equivalents. This is useful because MetaPost is frequently used with TEX/LATEX and those programs do not use standard font names. This file and the MetaPost output format driver are provided by Scott Pakin (`scott+ps2ed_AT_pakin.org`). Another example is wemf.fmp to be used under Windows. See the misc directory of the pstoedit source distribution. After loading the implicit (based on driver name) or explicit (based on the -fontmap option) font map file, a system specific map file is searched and loaded from the installation directory (unix.fmp or windows.fmp). This file can be used to redirect certain fonts to system specific names using the /AliasName notation described above.

3.3 Drawing related options

The following options are available:

[**-nc**] no curves. Normally pstoedit tries to keep curves from the input and transfers them to the output if the output format supports curves. If the output format does not support curves, then pstoedit replaces curves by a series of lines (see also **-flat** option). However, in some cases the user might wish to have this behavior also for output formats that originally support curves. This can be forced via the **-nc** option.

[**-mergelines**] Some output formats permit the representation of filled polygons with edges that are in a different color than the fill color. Since PostScript does not support this by the standard drawing primitives directly, drawing programs typically generate two objects (the outline and the filled polygon) into the PostScript output. *pstoedit* is able to recombine these, if they follow each other directly and you specify **-mergelines**. However, this merging is not supported by all output formats due to restrictions in the target format.

[**-filledrecttostroke**] Rectangles filled with a solid color can be converted to a stroked line with a width that corresponds to the width of the rectangle. This is of primary interest for output formats which do not support filled polygons at all. But it is restricted to rectangles only, i.e. it is not supported for general polygons

[**-mergetext**] In order to produce nice looking text output, programs producing PostScript files often split words into smaller pieces which are then placed individually on adjacent positions. However, such split text is hard to edit later on and hence it is sometime better to recombine these pieces again to form a word (or even sequence of words). For this pstoedit implements some heuristics about what text pieces are to be considered parts of a split word. This is based on the geometrical proximity of the different parts and seems to work quite well so far. But there are certainly cases where this simple heuristic fails. So please check the results carefully.

[**-ssp**] simulate subpaths. Several output formats do not support PostScript paths containing subpaths, i.e. paths with intermediate movetos. In the normal case, each subpath is treated as an independent path for such

output formats. This can lead to bad looking results. The most common case where this happens is if you use the **-dt** option and show some text with letters like e, o, or b, i.e. letters that have a "hole". When the **-ssp** option is set, pstoedit tries to eliminate these problems. However, this option is CPU time intensive!

[**-sfill**] simulate filling by individual strokes.

[**-flat *flatness factor***] If the output format does not support curves in the way PostScript does or if the **-nc** option is specified, all curves are approximated by lines. Using the **-flat** option one can control this approximation. This parameter is directly converted to a PostScript **setflat** command. Higher numbers, e.g. 10 give rougher, lower numbers, e.g. 0.1, give finer approximations.

[**-sclip**] simulate clipping. Most output formats of pstoedit do not have native support for clipping. For that *pstoedit* offers an option to perform the clipping of the graphics directly without passing the clippath to the output driver. However, this results in curves being replaced by a lot of line segments and thus larger output files. So use this option only if your output looks different from the input due to clipping. In addition, this "simulated clipping" is not exactly the same as defined in PostScript. There might be lines drawn at double size. Also clipping of text is not supported unless you also use the **-dt** option.

3.4 Diagnostic and debug options

The following options are available:

[**-dis**] Open a display during processing by Ghostscript. Some files only work correctly this way.

[**-q**] quiet mode - do not write startup message

[**-nq**] no exit from the PostScript interpreter. Normally Ghostscript exits after processing the pstoedit input-file. For debugging it can be useful to avoid this. If you do, you will have to type quit at the `GS>` prompt to exit from Ghostscript.

[**-v**] Switch on verbose mode. Some additional information is shown during processing.

[**-vl**] Switch on verbose mode with a given level. Some additional information is shown during processing.

[**-nb**] Since version 3.10 *pstoedit* uses the `-dDELAYBIND` option when calling Ghostscript. Previously the `-dNOBIND` option was used instead but that sometimes caused problems if a user's PostScript file overloaded standard PostScript operator with totally new semantic, e.g. lt for lineto instead of the standard meaning of "less than". Using **-nb** the old style can be activated again in case the `-dDELAYBIND` gives different results as before. In such a case please also contact the author.

[**-rdb**] Since version 3.10 *pstoedit* uses the `-dDELAYBIND` option when calling Ghostscript. But in version 9.22 of Ghostscript, that option is not supported anymore because of security reasons. As a fallback, that version provides the REALLYDELAYBIND option and pstoedit can use this if you supply the **-rdb** option. Use this with caution as it might open security risks, e.g. a PostScript file injecting some malicious code into PostScript standard operators. However, not using this option can cause some of the PostScript drawings operations to be not seen by pstoedit, hence causing missing artefacts in the output. Later versions of Ghostscript will probably support -dDELAYBIND again. But also in that case the security risk remains. So be careful with what files you process with pstoedit and Ghostscript.

[**-ups**] write text as plain string instead of hex string in intermediate format - normally useful for trouble shooting and debugging only.

[**-usefinddevice**] use the ghostscript internal finddevice operator during image handling instead of standard operators - normally useful for trouble shooting and debugging only.

[**-keep**] keep the intermediate files produced by pstoedit - for debug purposes only

[**-debugfonthandling**] writes verbose messages related to internal font processing - for debug purposes only

[**-gstest**] perform a basic test of the interworking with Ghostscript

[**-fakedateandversion**] Just for regression testing - uses a constant date and version string.

3.5 Input and outfile file arguments

[inputfile [outputfile]]

If neither an input nor an output file is given as argument, pstoedit works as filter reading from standard input and writing to standard output. The special filename "-" can also be used. It represents standard input if it is the first on the command line and standard output if it is the second. So "pstoedit - output.xxx" reads from standard input and writes to output.xxx

4 Available formats and their specific options

pstoedit allows passing individual options to an output format driver. This is done by appending all options to the format specified after the **-f** option. The format specifier and its options must be separated by a colon (:). If more than one option needs to be passed to the output format driver, the whole argument to **-f** must be enclosed within double-quote characters, thus:

-f *"format[:option option ...]"*

To see which options are supported by a specific format, type: **pstoedit -f format:-help**

The following description of the different formats supported by pstoedit is extracted from the source code of the individual drivers.

4.1 Format group: psf ps debug dump ps2ai gs

This group consists of the following variants:

psf: Flattened PostScript (no curves).

ps: Simplified PostScript with curves.

debug: for test purposes.

dump: for test purposes (same as debug).

ps2ai: Adobe Illustrator via ps2ai.ps of Ghostscript.

gs: any device that Ghostscript provides - use gs:format, e.g. gs:pdfwrite.

No format specific options

4.2 gcode - emc2 gcode format

See also: `http://linuxcnc.org/`
No format specific options

4.3 cairo - cairo driver

generates compilable c code for rendering with cairo
The following driver specific options are available:

[**-pango**] use pango for font rendering

[**-funcname** ***string***] sets the base name for the generated functions and variables. e.g. myfig

[**-header** ***string***] sets the output file name for the generated C header file. e.g. myfig.h

4.4 lwo - LightWave 3D object format

No format specific options

4.5 rib - RenderMan Interface Bytestream

No format specific options

4.6 rpl - Real3D Programming Language format

No format specific options

4.7 Format group: dxf dxf_14 dxf_s

This group consists of the following variants:

dxf: CAD exchange format version 9 - only limited features. Consider using dxf_14 instead..

dxf_14: CAD exchange format version 14 supporting splines and linetypes.

dxf_s: CAD exchange format version 14 supporting splines and linetypes.

The following driver specific options are available in this group:

[**-polyaslines**] use LINE instead of POLYLINE in DXF

[**-mm**] use mm coordinates instead of points in DXF (mm=pt/72*25.4)

[**-ctl**] map colors to layers

[**-filltohatch**] generate hatch objects from fill operations (still experimental)

[**-splineaspolyline**] approximate splines with PolyLines (only for -f dxf_s)

[**-splineasnurb**] experimental (only for -f dxf_s)

[**-splineasbspline**] experimental (only for -f dxf_s)

[**-splineassinglespline**] experimental (only for -f dxf_s)

[**-splineasmultispline**] experimental (only for -f dxf_s)

[**-splineasbezier**] use Bezier splines in DXF format (only for -f dxf_s)

[**-splineprecision** ***number***] number of samples to take from spline curve when doing approximation with -splineaspolyline or -splineasmultispline - should be =2 (default 5)

[**-dumplayernames**] dump all layer names found to standard output

[**-layers** ***string***] layers to be shown (comma separated list of layer names, no space)

[**-layerfilter** ***string***] layers to be hidden (comma separated list of layer names, no space)

4.8 java1 - java 1 applet source code

The following driver specific options are available:

[**java_class_name** ***string***] name of java class to generate

4.9 java2 - java 2 source code

The following driver specific options are available:

[**java_class_name** ***string***] name of java class to generate

4.10 pdf - Adobe's Portable Document Format

No format specific options

4.11 kil - .kil format for Kontour

No format specific options

4.12 text - text in different forms

The following driver specific options are available:

[**-height** ***number***] page height in terms of characters

[**-width** ***number***] page width in terms of characters

[**-dump**] dump text pieces

4.13 sk - Sketch format

No format specific options

4.14 mpost - MetaPost format

No format specific options

4.15 asy - Asymptote Format

No format specific options

4.16 mma - Mathematica graphics

The following driver specific options are available:

[**-eofillfills**] Filling is used for eofill (default is not to fill)

4.17 latex2e - LaTeX2e picture format

The following driver specific options are available:

[**-integers**] round all coordinates to the nearest integer

4.18 noixml - Nemetschek NOI XML format

Nemetschek Object Interface XML format
The following driver specific options are available:

[**-r** ***string***] Allplan resource file

[**-bsl** ***number***] Bezier Split Level (default 3)

4.19 pic - PIC format for troff et.al.

The following driver specific options are available:

[**-troff**] troff mode (default is groff)

[**-landscape**] landscape output

[**-portrait**] portrait output

[**-keepfont**] print unrecognized literally

[**-text**] try not to make pictures from running text

[**-debug**] enable debug output

4.20 Format group: hpgl pcl

This group consists of the following variants:

hpgl: HPGL code.

pcl: PCL code.

The following driver specific options are available in this group:

[**-penplotter**] plotter is pen plotter (i.e. no support for specific line widths)

[**-pencolorsfromfile**] read pen colors from file drvhpgl.pencolors in pstoedit's data directory

[**-pencolors** ***number***] maximum number of pen colors to be used by pstoedit (default 0) -

[**-filltype** ***string***] select fill type e.g. FT 1

[**-hpgl2**] Use HPGL/2 instead of HPGL/1

[**-rot90**] rotate hpgl by 90 degrees

[**-rot180**] rotate hpgl by 180 degrees

[**-rot270**] rotate hpgl by 270 degrees

4.21 pcbi - engrave data - insulate/PCB format

No format specific options

4.22 pcb - pcb format

See also: `http://pcb.sourceforge.net` and `http://www.penguin.cz/~utx/pstoedit-pcb/`
The following driver specific options are available:

[**-grid** ***double number***] attempt to snap relevant output to grid (mils) and put failed objects to a different layer

[**-snapdist** ***double number***] grid snap distance ratio (0 ¡ snapdist ¡= 0.5, default 0.1)

[**-tshiftx** ***double number***] additional x shift measured in target units (mils)

[**-tshifty** ***double number***] additional y shift measured in target units (mils)

[**-mm**] switch to metric units (mm)

[**-stdnames**] use standard layer names instead of descriptive names

[**-forcepoly**] force all objects to be interpreted as polygons

4.23 pcbfill - pcb format with fills

See also: `http://pcb.sourceforge.net`
No format specific options

4.24 pcb-rnd - pcb-rnd format

See also: `http://repo.hu/projects/pcb-rnd` and `http://www.penguin.cz/~utx/pstoedit-pcb/`
The following driver specific options are available:

[**-grid** ***double number***] attempt to snap relevant output to grid (mils) and put failed objects to a different layer

[**-snapdist** ***double number***] grid snap distance ratio (0 ¡ snapdist ¡= 0.5, default 0.1)

[**-tshiftx** ***double number***] additional x shift measured in target units (mils)

[**-tshifty** ***double number***] additional y shift measured in target units (mils)

[**-mm**] switch to metric units (mm)

[**-forcepoly**] force all objects to be interpreted as polygons

4.25 gschem - gschem format

See also: `http://wiki.geda-project.org/geda:gaf`
No format specific options

4.26 cfdg - Context Free Design Grammar

Context Free Design Grammar, usable by Context Free Art (http://www.contextfreeart.org/)
No format specific options

4.27 tk - tk and/or tk applet source code

The following driver specific options are available:

[**-R**] swap HW

[**-I**] no impress

[**-n** ***string***] tagnames

4.28 vtk - VTK driver

this is a long description for the VTKe driver

The following driver specific options are available:

[**-VTKeoption** ***integer***] just an example

4.29 svm - StarView/OpenOffice.org metafile

StarView/OpenOffice.org metafile, readable from OpenOffice.org 1.0/StarOffice 6.0 and above.

The following driver specific options are available:

[**-m**] map to Arial

[**-nf**] emulate narrow fonts

4.30 Format group: svg xaml

This group consists of the following variants:

svg: Scalable Vector Graphics.

xaml: eXtensible Application Markup Language.

The following driver specific options are available in this group:

[**-localdtd**] use local DTD

[**-standalone**] create stand-alone type svg

[**-withdtd**] write DTD

[**-withgrouping**] write also ordinary save/restores as SVG group

[**-nogroupedpath**] do not write a group around paths

[**-noviewbox**] do not write a view box

[**-texmode**] TEX mode

[**-imagetofile**] write raster images to separate files instead of embedding them

[**-notextrendering**] do not write textrendering attribute

[**-border** ***number***] additional border to draw around bare bounding box (in percent of width and height)

[**-title** ***string***] text to use as title for the generated document

4.31 gnuplot - gnuplot format

No format specific options

4.32 tgif - Tgif .obj format

The following driver specific options are available:

[**-ta**] text as attribute

4.33 Format group: fig xfig tfig

This group consists of the following variants:

fig: .fig format for xfig.

xfig: .fig format for xfig.

tfig: .fig format for xfig - test only version.

The xfig format driver supports special fontnames, which may be produced by using a fontmap file. The following types of names are supported:

```
General notation:
"PostScript Font Name" ((LaTeX|PostScript|empty)(::special)::)XFigFontName

Examples:

Helvetica LaTeX::SansSerif
Courier LaTeX::special::Typewriter
GillSans "AvantGarde Demi"
Albertus PostScript::special::"New Century Schoolbook Italic"
Symbol ::special::Symbol (same as PostScript::special::Symbol)
```

See also the file examplefigmap.fmp in the misc directory of the pstoedit source distribution for an example font map file for xfig. Please note that the fontname has to be among those supported by xfig. See - `https://mcj.sourceforge.net/fig-format.html` for a list of legal font names

The following driver specific options are available in this group:

[**-startdepth** ***number***] set the initial depth (default 999)

[**-metric**] switch to centimeter display (default inches)

[**-usecorrectfontsize**] do not scale fonts for xfig. Use this if you also use this option with xfig

[**-depth** ***number***] set the page depth in inches (default 11)

4.34 idraw - Interviews draw format (EPS)

No format specific options

4.35 sample - sample driver

This is a long description for the sample driver

The following driver specific options are available:

[**-sampleoption** ***integer***] just an example

4.36 Format group: gmfa gmfb plot plot-pnm plot-cgm plot-ai plot-svg plot-ps plot-fig plot-pcl plot-hpgl plot-tek

This group consists of the following variants:

gmfa: ASCII GNU metafile.

gmfb: binary GNU metafile.

plot: GNU libplot output types, e.g. plot:-plotformat X.

plot-pnm: pnm via GNU libplot.

plot-cgm: cgm via GNU libplot.

plot-ai: ai via GNU libplot.

plot-svg: svg via GNU libplot.

plot-ps: ps via GNU libplot.

plot-fig: fig via GNU libplot.

plot-pcl: pcl via GNU libplot.

plot-hpgl: hpgl via GNU libplot.

plot-tek: tek via GNU libplot.

The following driver specific options are available in this group:

[**-plotformat** ***string***] plotutil format to generate

5 NOTES ON SPECIFIC FORMATS AND DRIVERS

5.1 autotrace

pstoedit cooperates with autotrace. Autotrace can now produce a dump file for further processing by pstoedit using the **-bo** (back-end only) option. Autotrace is a program written by a group around Martin Weber and can be found at `https://sourceforge.net/projects/autotrace/`.

5.2 Ps2ai

The ps2ai output format driver is not a native pstoedit output format driver. It does not use the pstoedit PostScript flattener, instead it uses the PostScript program ps2ai.ps which is installed in the Ghostscript distribution directory. It is included to provide the same "look-and-feel" for the conversion to AI. However, lot's of files do not convert nicely or at all using ps2ai.ps. So a native pstoedit driver would be much better. Anyone out there to take this? The AI format is usable for example by Mayura Draw (`http://www.mayura.com`). Also a driver to the Mayura native format would be nice.

An alternative to the ps2ai based driver is available via the -f plot:ai format if the libplot(ter) is installed.

You should use a version of Ghostscript greater than or equal to 6.00 for using the ps2ai output format driver.

5.3 MetaPost

Note that, as far as Scott knows, MetaPost does not support PostScript's eofill. The MetaPost output format driver just converts eofill to fill, and issues a warning if verbose is set. Fortunately, very few PostScript programs rely on the even-odd fill rule, even though many specify it.

For more on MetaPost see:
`http://tug.org/metapost`

5.4 Context Free - CFDG

The driver for the CFDG format (drvcfdg) defines one shape per page of PostScript, but only the first shape is actually rendered (unless the user edits the generated CFDG code, of course). CFDG does not support multi-page output, so this probably is a reasonable thing to do.

For more on Context Free see: `http://www.contextfreeart.org/`

5.5 LaTeX2e

- LaTeX2e's picture environment is not very powerful. As a result, many elementary PostScript constructs are ignored – fills, line thicknesses (besides "thick" and "thin"), and dash patterns, to name a few. Furthermore, complex pictures may overrun TeX's memory capacity. (The eepic package overcomes many such restrictions.)

- Some PostScript constructs are not supported directly by "picture", but can be handled by external packages. If a figure uses color, the top-level document will need to do a `"\usepackage{color}"` or `"\usepackage{xcolor}"`. And if a figure contains rotated text, the top-level document will need to do a `"\usepackage{rotating}"`.

- All lengths, coordinates, and font sizes output by the output format driver are in terms of `\unitlength`, so scaling a figure is simply a matter of doing a `"\setlength{\unitlength}{...}"`.

- The output format driver currently supports one output format driver specific option, "integers", which rounds all lengths, coordinates, and font sizes to the nearest integer. This makes hand-editing the picture a little nicer.

- Why is this output format driver useful? One answer is portability; any LaTeX2e system can handle the picture environment, even if it cannot handle PostScript graphics. (pdfLaTeX comes to mind here.) A second answer is that pictures can be edited easily to contain any arbitrary LaTeX2e code. For instance, the text in a figure can be modified to contain complex mathematics, non-Latin alphabets, bibliographic citations, or – the real

reason Scott wrote the LaTeX2e output format driver – hyperlinks to the surrounding document (with help from the hyperref package).

5.6 Creating a new output format driver

To implement a new output format driver you can start from `drvsampl.cpp` and `drvsampl.h`. See also comments in `drvbase.h` and `drvfuncs.h` for an explanation of methods that should be implemented for a new output format driver.

6 ENVIRONMENT VARIABLES

A default PostScript interpreter to be called by pstoedit is specified at compile time. You can overwrite the default by setting the GS environment variable to the name of a suitable PostScript interpreter.

You can check which name of a PostScript interpreter was compiled into pstoedit using: **pstoedit -help -v**.

See the Ghostscript manual for descriptions of environment variables used by Ghostscript, most importantly `GS_FONTPATH` and `GS_LIB`; other environment variables also affect output to display, print, and additional filtering and processing. See the related documentation.

pstoedit allocates temporary files using the function *tempnam*(3). Thus the location for temporary files might be controllable by other environment variables used by this function. See the *tempnam*(3) man-page for descriptions of environment variables used. On UNIX like system this is probably the `TMPDIR` variable, on DOS/WINDOWS either `TMP` or `TEMP`.

7 TROUBLE SHOOTING

If you have problems with *pstoedit* first try whether Ghostscript successfully displays your file. If yes, then try **pstoedit -f ps** *infile.ps testfile.ps* and check whether *testfile.ps* still displays correctly using Ghostscript. If this file does not look correctly then there seems to be a problem with *pstoedit*'s PostScript front-end. If this file looks good but the output for a specific format is wrong, the problem is probably in the output format driver for the specific format. In either case send bug fixes and reports to the author.

A common problem with PostScript files is that the PostScript file redefines one of the standard PostScript operators inconsistently. There is no effect of this if you just print the file since the original PostScript "program" uses these new operators in the new meaning and does not use the original ones anymore. However, when run under the control of pstoedit, these operators are expected to work with the original semantics.

So far I've seen redefinitions for:

- lt - "less-then" to mean "draw a line to"
- string - "create a string object" to mean "draw a string"
- length - "get the length of e.g. a string" to a "float constant"

I've included work-arounds for the ones mentioned above, but some others could show up in addition to those.

8 RESTRICTIONS

- Non-standard fonts (e.g. TEX bitmap fonts) are mapped to a default font which can be changed using the **-df** option. *pstoedit* chooses the size of the replacement font such that the width of the string in the original font is the same as with the replacement font. This is done for each text fragment displayed. Special character encoding support is limited in this case. If a character cannot be mapped into the target format, pstoedit displays a '#' instead. See also the -uchar option.
- pstoedit supports bitmap graphics only for some output format drivers.
- Some output format drivers, e.g. the Gnuplot output format driver or the 3D output format driver (rpl, lwo, rib) do not support text.
- For most output format drivers pstoedit does not support clipping (mainly due to limitations in the target format). You can try to use the **-sclip** option to simulate clipping. However, this does not work in all cases as expected.
- Special note about the Java output format drivers (java1 and java2). The java output format drivers generate a java source file that needs other files in order to be compiled and usable. These other files are Java classes (one applet and support classes) that allow stepping through the individual pages of a converted PostScript document. This applet can easily be activated from a HTML-document. See the `contrib/java/java1/readme_java1.txt` or `contrib/java/java2/readme_java2.htm` files for more details.

9 FAQs

1. Why do letters like O or B get strange if converted to TGIF/XFIG using the **-dt** option?

   Most output format drivers do not support composite paths with intermediate gaps (moveto's) and second do not support very well the (eo)fill operators of PostScript (winding rule). For such objects *pstoedit* breaks them into smaller objects whenever such a gap is found. This results in the "hole" being filled with black color instead of being transparent. Since version 3.11 you can try the **-ssp** option in combination with the XFIG output format driver.

2. Why does pstoedit produce ugly results from PostScript files generated by dvips?

   This is because TEX documents usually use bitmap fonts. Such fonts cannot be used as native font in other format. So pstoedit replaces the TEX font with another native font. Of course, the replacement font will in most cases produce another look, especially if mathematical symbols

are used. Try to use PostScript fonts instead of the bitmap fonts when generating a PostScript file from TEX or LATEX.

10 AUTHOR

Wolfgang Glunz, wglunz35_AT_pstoedit.net, https://de.linkedin.com/in/wolfgangglunz

11 CANONICAL ARCHIVE SITE

https://www.pstoedit.com

At this site you also find more information about *pstoedit* and related programs and hints how to subscribe to a mailing list in order to get informed about new releases and bug-fixes.

If you like pstoedit - please express so also at Facebook https://www.facebook.com/pstoedit.

12 ACKNOWLEDGMENTS

- Klaus Steinberger Klaus.Steinberger_AT_physik.uni-muenchen.de wrote the initial version of this man-page.
- Lar Kaufman revised the increasingly complex command syntax diagrams and updated the structure and content of this man-page following release 2.5.
- David B. Rosen rosen_AT_unr.edu provided ideas and some PostScript code from his ps2aplot program.
- Ian MacPhedran Ian_MacPhedran_AT_engr.USask.CA provided the XFIG output format driver.
- Carsten Hammer chammer_AT_hermes.hrz.uni-bielefeld.de provided the Gnuplot output format driver and the initial DXF output format driver.
- Christoph Jaeschke provided the OS/2 metafile (MET) output format driver. Thomas Hoffmann thoffman_AT_zappa.sax.de did some further updates on the OS/2 part.
- Jens Weber rz47b7_AT_PostAG.DE provided the MS Windows metafile (WMF) output format driver, and a graphical user interface (GUI).
- G. Edward Johnson lorax_AT_nist.gov provided the CGM Draw library used in the CGM output format driver.
- Gerhard Kircher kircher_AT_edvz.tuwien.ac.at provided some bug fixes.
- Bill Cheng bill.cheng_AT_acm.org provided help with the TGIF format and some changes to TGIF to make the output format driver easier to implement.
- Reini Urban rurban_AT_sbox.tu-graz.ac.at provided input for the extended DXF output format driver.
- Glenn M. Lewis glenn_AT_gmlewis.com provided RenderMan (RIB), Real3D (RPL), and LightWave 3D (LWO) output format drivers.

- Piet van Oostrum piet_AT_cs.ruu.nl made several bug fixes.
- Lutz Vieweg lkv_AT_mania.robin.de provided several bug fixes and suggestions for improvements.
- Derek B. Noonburg derekn_AT_vw.ece.cmu.edu and Rainer Dorsch rd_AT_berlepsch.wohnheim.uni-ulm.de isolated and resolved a Linux-specific core dump problem.
- Rob Warner rcw2_AT_ukc.ac.uk made pstoedit compile under RiscOS.
- Patrick Gosling jpmg_AT_eng.cam.ac.uk made some suggestions regarding the usage of pstoedit in Ghostscript's SAFER mode.
- Scott Pakin scott+ps2ed_AT_pakin.org for the Idraw output format driver and the autoconf support.
- Peter Katzmann p.katzmann_AT_thiesen.com for the HPGL output format driver.
- Chris Cox ccox_AT_airmail.net contributed the Tcl/Tk output format driver.
- Thorsten Behrens Thorsten_Behrens_AT_public.uni-hamburg.de and Bjoern Petersen for reworking the WMF output format driver.
- Leszek Piotrowicz leszek_AT_sopot.rodan.pl implemented the image support for the XFIG driver and a JAVA based GUI.
- Egil Kvaleberg egil_AT_kvaleberg.no contributed the pic output format driver.
- Kai-Uwe Sattler kus_AT_iti.cs.uni-magdeburg.de implemented the output format driver for Kontour.
- Scott Pakin, scott+ps2ed_AT_pakin.org provided the MetaPost and LaTeX2e and MS PowerPoint output format driver.
- The MS PowerPoint driver (pptx) uses the libzip library - https://libzip.org. Under MS Windows, libzip and libz are linked statically into the driver DLL. Thanks to the whole libzip team. The libzip license is explained here https://libzip.org/license.
- Burkhard Plaum plaum_AT_IPF.Uni-Stuttgart.de added support for complex filled paths for the XFIG output format driver.
- Bernhard Herzog herzog_AT_online.de contributed the output format driver for sketch (http://www.skencil.org/)
- Rolf Niepraschk (niepraschk_AT_ptb.de) converted the HTML man page to LaTeX format. This allows generating the UNIX style and the HTML manual from this base format.
- Several others sent smaller bug fixed and bug reports. Sorry if I do not mention them all here.
- Gisbert W. Selke (gisbert_AT_tapirsoft.de) for the Java 2 output format driver.
- Robert S. Maier (rsm_AT_math.arizona.edu) for many improvements on the libplot output format driver and for libplot itself.

- The authors of pstotext (mcjones_AT_pa.dec.com and birrell_AT_pa.dec.com) for giving me the permission to use their simple PostScript code for performing rotation.
- Daniel Gehriger gehriger_AT_linkcad.com for his help concerning the handling of Splines in the DXF format.
- Allen Barnett libemf_AT_lignumcomputing.com for his work on the libEMF which allows creating WMF/EMF files under *nix systems.
- Dave dave_AT_opaque.net for providing the libming which is a multiplatform library for generating SWF files.
- Masatake Yamoto for the introduction of autoconf, automake and libtool into pstoedit
- Bob Friesenhahn for his help and the building of the Magick++ API to ImageMagick.
- Barak Pearlmutter from Debian for his upfront testing of new versions.
- But most important: Peter Deutsch, Ken Sharp, Chris Liddell from the GhostScript team at Artifex, and Russell Lang gsview_AT_ghostgum.com.au for their help and answers regarding Ghostscript and gsview.

13 LEGAL NOTICES

Trademarks mentioned are the property of their respective owners.

Some code incorporated in the pstoedit package is subject to copyright or other intellectual property rights or restrictions including attribution rights. See the notes in individual files.

pstoedit is controlled under the Free Software Foundation GNU Public License (GPL). However, this does not apply to the additional plugins.

Ghostscript is a redistributable software package with copyright restrictions controlled by Artifex https://artifex.com/.

pstoedit has no other relation to Ghostscript besides calling it in a subprocess.

The authors, contributors, and distributors of pstoedit are not responsible for its use for any purpose, or for the results generated thereby.

Restrictions such as the foregoing may apply in other countries according to international conventions and agreements.