The mpostinl Package

Niklas Beisert

Institut für Theoretische Physik
Eidgenössische Technische Hochschule Zürich
Wolfgang-Pauli-Strasse 27, 8093 Zürich, Switzerland

nbeisert@itp.phys.ethz.ch

17 January 2018, v1.21

Abstract

mpostinl is a LaTeX 2ε package which enables the embedding of METAPOST figures within a LaTeX document. The package automatically collects the embedded definitions and figures in a `.mp` file, adds an appropriate LaTeX document structure, and compiles it to `.mps` files. It also allows for various configuration options to manage the generation of files and compilation.

Contents

1 Introduction

METAPOST is a versatile tool to generate vector graphics files from a plain text source for inclusion in LaTeX documents which allows to typeset labels in native LaTeX code and fonts. The METAPOST compiler `mpost` typically compiles a metapost source file (extension `.mp`) to a set of metapost figure files (extension `.mps`) which are encapsulated postscript files (`.eps`) with a somewhat reduced scope. The figure files can be included in a LaTeX document by `\includegraphics`, and modern TeX distributions are typically able to handle the required conversions on the fly.

While many steps in the compilation process are automated, some management is left to the user:

- Link the figures in the metapost source with the LaTeX source by choosing an appropriate figure filename (or numeric identifier).
- When a figure is changed, the metapost source file must be compiled to update the figure files, and afterwards the LaTeX source must be compiled to introduce the changes into the output document.
- Keep any changes aligned between LaTeX and metapost source files.
- Set up a proper LaTeX document structure to compile labels via LaTeX.
- Distribute source and figures as separate files.

The current LaTeX package mpostinl helps in the management of metapost figures by embedding them into the LaTeX source:

- Figures are displayed at the location of their definition within the LaTeX source to facilitate alignment between text and figures.
- A metapost source file with LaTeX structure for the labels is generated.
- Figure files are compiled automatically from within the LaTeX compiler.

For example, a simple figure consisting of a circle might be represented as:

```
\begin{mpostfig}
draw fullcircle scaled 1cm;
\end{mpostfig}
```

The package also offers several options and customisations to streamline its use in several situations:

- Figures can be assigned labels or filenames for later usage.
- There are several options and mechanisms to minimise the need for multiple compiler passes to generate the desired output.
- The package can handle several metapost files in a row with common definitions or include files.
- The font generation, LaTeX structure and generated filenames can be customised.

1.1 Related CTAN Packages

There are at least four other LaTeX packages which offer a similar functionality:

- The package emp provides similar basic functionality to compose a metapost file, but it does not automatically compile it. Analogously to the `picture` environment the size for every figure must be specified explicitly.
- The package mpfig by Tomasz Cholewo (not available on CTAN) provides very basic functionality to compose a metapost file.
- The package mpgraphics provides similar functionality to compose and compile a metapost file. It processes all figures immediately and does not offer labels for recycling figures.
- The package gmp provides similar functionality to compose and compile a metapost file. It processes all figures immediately and allows to inject LaTeX definitions into the metapost code at the price of modifying the metapost syntax slightly.

The philosophy of the present package is to generate a single metapost file containing all figures as in a traditional metapost setup which can be compiled in one pass. The aim is to provide a metapost setup which works with as little configuration as possible, but which offers several configuration options to customise the management in the desired way. The package offers most of the functionality of the above packages, but (presently) misses out on some more advanced features, see section 3.4.

2 Usage

This manual assumes familiarity with the METAPOST figure description language. The METAPOST manual is a recommended introduction and an excellent reference.

To use the package mpostinl add the command

```
\usepackage{mpostinl}
```

to the preamble of your LaTeX document. If not yet present, the package graphicx will be loaded automatically. Metapost figures and definitions are to be specified using the environments `mpostfig` and `mpostdef`, respectively, as described in section 2.1.

The package collects the figure files contained in the LaTeX source, writes them to a metapost file, and compiles them at the end of the LaTeX document. This means that the figures (or their updates) will normally *not* be available in the first LaTeX run and a secondary run is required for the correct output, see section 2.5 for strategies to avoid a second pass.

You should make sure that LaTeX allows calling of external programs. If this feature is not enabled by default, it is achieved by calling `latex` with the command line option `-shell-escape` (or `-shell-restricted` if `mpost` is in the list of permissible commands):

`latex -shell-escape` *source*

In the MiKTeX distribution the appropriate command line option is `-enable-write18`. In a LaTeX front end this option may be configurable in the preferences. If the shell escape is not available, the generated metapost file(s) *filename* (typically the same as the LaTeX source *source*) must be compiled manually:

`mpost -tex=latex` *filename*

Some extended configuration options and situations are described in the following sections: package options are listed in section 2.2; some options for writing and compiling are discussed in section 2.3; the generation of multiple metapost files is described in 2.4; finally, some issues regarding other LaTeX packages are discussed in section 2.6.

2.1 Figures and Definitions

`mpostfig` The main functionality provided by the package is the `mpostfig` environment:

```
\begin{mpostfig}[opts]
  metapost code
\end{mpostfig}
```

The above block is translated to the following code in the metapost file:

```
filenametemplate:="filename";
beginfig(number)
  metapost code
endfig;
```

The optional argument *opts* of the `mpostfig` environment is a comma-separated list of options:

- `show`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Show the figure in place. If neither `file` nor `label` are specified, this option is forced to `true`.
- `file=`*filename* – Filename for the figure.
- `label=`*label* – Label for later use by the command `\mpostuse{`*label*`}`.
- `opt=`*opt* – Options to be passed on to `\includegraphics[`*opt*`]`.
- `now`[`=true`|`false`] (no value implies `true`, overrides global setting `nowall`) – Compile figure immediately. Requires global option `now` to work.
- `twice`[`=true`|`false`] (no value implies `true`, overrides global setting `twice`) – Compile this figure twice.

Please note the following restrictions due to the implementation via the package verbatim:

- The closing statement `\end{mpostfig}` must be on a line on its own. Any amount of leading whitespace is allowed, and trailing characters are ignored.
- The opening statement `\begin{mpostfig}` *without* optional arguments must not be followed immediately by empty lines or by the closing statement `\end{mpostfig}`. Note that commented lines do not help. Either avoid leading empty lines and empty bodies or use empty optional arguments: `\begin{mpostfig}[]`.
- Lines starting with a single '`%`' are ignored and do not appear in the metapost file. Double '`%%`' or leading spaces preserve comments in the metapost file.
- The environment `mpostfig` cannot be used within macro arguments or particular other environments. If you want to display a figure in these situations, you should declare the figure with a label and display it via the command `\mpostuse` (see below).

`\mpostuse` `\mpostgetname` Figures which have been previously declared with a label *label* can be recycled any number of times with the command:

```
\mpostuse[opts]{label}
```

The options are passed on to `\includegraphics[`*opt*`]{`*filename*`}`. Furthermore, the filename of a figure can be obtained by calling `\mpostgetname{`*label*`}`. The filename is returned in the macro `\mpostfigurename`.

`mpostdef` Plain metapost code which is not part of a figure (definitions, assignments) can be specified by the `mpostdef` environment:

```
\begin{mpostdef}[opts]
  code
\end{mpostdef}
```

Note that the same restrictions as for `mpostfig` (see above) apply to `mpostdef`. The optional argument *opts* is a comma-separated list of options:

- `tex`[`=true`|`false`] (no value implies `true`, initially set to `false`) – The block *code* represents TEX or LATEX definitions rather than METAPOST code, which will be enclosed in the metapost file by `verbatimtex` and `etex`.
- `global`[`=true`|`false`] (no value implies `true`, overrides global setting `globaldef`) – In a setup with multiple metapost files, the block *code* is applied to all files, not just the current file.

2.2 Package Options

`\mpostsetup` Options can be passed to the package by:

```
    \usepackage[opts]{mpostinl}
or  \PassOptionsToPackage{opts}{mpostinl}
or  \mpostsetup{opts}
```

`\PassOptionsToPackage` must be used before `\usepackage`; `\mpostsetup` must be used afterwards (for selected options). *opts* is a comma-separated list of options. Below we provide a complete list of available options while some of the more relevant options are discussed in detail in the following sections:

- `draft`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable `draft` mode by declaring `draft:=1` at the top of the metapost file.
- `final` – Same as `draft=false`.
- `write`[`=true`|`false`] (no value implies `true`, initially set to `true`) – Enable/disable writing to metapost file.
- `compile`[`=true`|`false`] (no value implies `true`, initially set to `true`) – Enable/disable automatic compilation of metapost file. Requires shell escapes to work properly.
- `twice`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable secondary metapost compilation. Some metapost files may require this to produce the intended output.
- `fonts`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable embedding of fonts in metapost figures by setting `prologues:=3` thus making them proper encapsulated postscript files. This option may be required if figure files are used outside a TEX environment, e.g. if the files are to be viewed or processed by `.eps` tools. Has no effect if `prologues` option is specified.
- `prologues=`*value* – Declares `prologues:=`*value* at the top of the metapost file. Set *value* to empty to disable the `prologues` statement.
- `lineno`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable line number indicators in the metapost file. All blocks in the metapost file will start with the line number where this block can be found in the LATEX source file. To view the source file name you should load the package currfile.

- `labelnames`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Use the figure label instead of a consecutive number to construct the name for figure files. The benefit of this mode is that the label will usually stay fixed while the number may change when adding or removing figures. The drawback is that the LaTeX internal labels will appear as part of the filenames cluttering the directory slightly more. Moreover, one has to make sure that the figure labels are valid filenames in your operating system, i.e. use special characters with care, better only use alphanumeric characters, and bear in mind that some operating systems do not distinguish upper- and lowercase letters.

- `latex`[`=true`|`false`] (no value implies `true`, initially set to `true`) – Switch between TeX (`false`) and LaTeX (`true`) processing of labels. In LaTeX mode a basic LaTeX document structure is provided by the metapost file.

- `compiler=`*compiler* – Set the compiler program for labels to *compiler* (command line option `-tex=`*compiler* for METAPOST). The default is `tex` (in TeX mode) or `latex` (in LaTeX mode). Set *compiler* to empty to specify no compiler.

- `format=`*tag* – Write format tag `%&`*tag* to specify the compiler program for labels. The default is `tex` (in TeX mode) or `latex` (in LaTeX mode). Set *tag* to empty to write no format tag.

- `class=`*class* (initially set to `article`) – Set the document class to be used in LaTeX mode.

- `classopt=`*options* – Set the options for the `\documentclass` statement in LaTeX mode, e.g. `12pt` or `11pt`. No options are specified initially resulting in the `10pt` font set.

- `mem=`*mem* – Set the metapost format file to *mem*.

- `command=`*command* – Use *command* to process the generated file(s). By default the METAPOST program `mpost` is invoked to compile the generated metapost files (with appropriate command line options).

- `now`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Activate/deactivate immediate mode. In immediate mode, figures can be processed immediately so that a secondary LaTeX pass is not required to display the figure correctly. Note that this option merely enables recording of all the required definitions so that immediate processing will be possible, but it does not activate immediate processing per se.

- `nowall`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable immediate processing for all figures by default. This option will call the METAPOST compiler for every figure. While convenient, it requires heavier processing.

- `nowkeep`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable immediate generation of figures by individual metapost files. If this mode is enabled, the filename for immediate processing of the figure *filename*`.mps` will be *filename*`.mp`. Otherwise the metapost code is stored in a temporary file and is overwritten by any subsequent immediate processing.

- `globaldef`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable the `global` option for the `mpostdef` environments by default. This option can be used to specify a global block of definitions by enclosing it with appropriate `\mpostsetup` statements.

- `template=`*template* (initially set to `\mpostfilename-#1.mps`) – Set the template for figure filenames for which it is not given explicitly. The parameter `#1` carries the number (or label) to be used. Moreover, the macro `\mpostfilename` carries the current metapost file.

- `extension=`*ext* (initially set to `mps`) – Set the default extension for figure filenames.
- `numberwithin=`*counter* – Declares the figure counter to be a child of *counter*. In other words, the figure counter is reset when *counter* is increased, and the figure number will be composed as *counter*`-`*figure*. Using this option with a top-level counter such as `section` or `chapter` stabilises the figure numbering by making changes to the sequence have effects only within the present section or chapter.

Admittedly, some of these options are hardly necessary as they will have little impact on output or performance in ordinary situations. They are provided for completeness, to make the package work in more exotic situations, and/or to satisfy some personal taste regarding how things should be managed.

`\mpostplaceholder` Finally, the package allows to customise the placeholder which is displayed when a figure file is not (yet) present after the first LaTeX pass (or in case of some compile error) or a figure label does not exist. In these situations the following macro is called:

`\mpostplaceholder[`*type*`]{`*name*`}`

type is either '`file`' or '`label`' and *name* is the missing filename or label. By default this command displays a 1"×0.6" box containing the missing filename or label. This behaviour can be customised by overwriting the macro.

2.3 Writing and Compiling Options

The following discusses some package options regarding the writing and compiling of metapost files in more detail.

Label Typesetting Options. Metapost figure files may or may not include the METAFONT fonts which are used by the figure labels. When the generated metapost figures are only used within a LaTeX document in a standard TeX distribution, it is not necessary to include the required fonts as they are automatically supplied by the LaTeX compiler. This omission reduces the size of the figure files (but apparently it has no impact on the size of the compiled LaTeX document in modern TeX distributions). However, the postscript structure of such figure files is incomplete, and therefore the labels typically appear distorted in external viewing or processing tools. If the figure files are to be viewed, processed or passed on to a publisher, it makes sense to include the required fonts. The latter is achieved by enabling the package option `fonts`. In this context, one can also set the default figure file extension to `eps` by means of the package option `extension`.

By default, the package provides a LaTeX document structure for processing labels by LaTeX. The default document class is `article` without options. An alternative class and options can be specified by the package options `class` and `classopt`. Further packages or macros should be declared as usual by specifying them in a `mpostdef` enviroment in `tex` mode. If no LaTeX structure is desired, set the package option `latex` to `false`. If the typesetting requires an advanced compiler beyond `latex` or `tex`, it should be specified by the package option `compiler`.

Compiling Options. When the figure files are in a final form it may make sense to disable the compiling or even the writing of the metapost file(s) by setting the package options `compile` or `write` to `false`. In particular, this may be desirable if the LaTeX source is uploaded to a repository or passed on to a publisher.

By default the package checks whether the generated files change with respect to the previous LaTeX pass, and only modified files are compiled thereupon. This check can be disabled by setting the option `checksum` to `false`, in which case all files are compiled.

When the metapost file compiles with errors, one can inspect the generated metapost file. To this end it may be helpful to know which part of the LaTeX source file is responsible

for which part of the metapost file. The package option `lineno` activates prepending every block in the metapost file by the corresponding location in the LaTeX source. If the LaTeX source is distributed over several files, the source filename can also be provided if the package currfile is loaded.

Filename Options. By default the names of figure files take the form

filename-*specifier*.*ext*

where *filename* is the name of the metapost file, *specifier* is an integer number which enumerates the figures and *ext* is '`mps`'. There are several options to customise the scheme:

- The figure option `file` allows to explicitly specify the desired filename.
- The package option `numberwithin=`*section* can be used to associate the counting with a top-level section counter (such as '`section`' or '`chapter`'). Then *specifier* takes the form *section*-*number*, where *number* is an integer that enumerates the figures within the present *section*.
- The package option `labelnames` lets *specifier* be the figure label. This option should be used with care, as the operating system does not necessarily allow or distinguish all characters which are available for TeX macros.
- The extension *ext* can be customised by the package option `extension`.
- The package option `template` allows to customise the above template. In composing the template, the argument `#1` carries the *specifier* and `\mpostfilename` carries the name of the present metapost file.

2.4 Multiple Files

By default the package mpostinl writes out a single file *source*`.mp` if the LaTeX source is called *source*`.tex`. However, the package can also be configured to write out several metapost files. This feature can be used to declare one metapost file for each top-level section of a large LaTeX document (e.g. section or chapter). Alternatively, one could define different metapost files for figures of different kinds (e.g. technical drawings, diagrams, graphs, charts). Note that these could well use different sets of metapost macros and variables.

This feature can make sense if the metapost source or generated figures are to be passed on to someone else (e.g. publisher) in order to help clarify the placement of figures. It may also be useful if selected sections of the document are generated individually by means of `\includeonly`, in which case only the relevant metapost file is generated and compiled.

The contents of each metapost file should be enclosed by `\mpostfile` and `\mpostdone`.

`\mpostfile` A new metapost file is started by the command:

`\mpostfile[`*opts*`]{`*filename*`}`

filename is the filename without `.mp` extension. *opt* is a comma-separated list of options: the only available option is `include`[`=true`|`false`] (no value implies `true`, initially set to `false`) which declares whether the file is an include file.

`\mpostdone` The present metapost file is completed by the command `\mpostdone` which also compiles the contained figures. Note that this command is called automatically at the end of the LaTeX document.

Global Definitions. Definitions in `mpostdef` environments apply to the present metapost file only. However, definitions can also be specified for all metapost files by means of the option `global`:

```
\begin{mpostdef}[global]
```

These definitions are stored internally and will be written to all subsequent metapost files.

To declare several consecutive (or all) blocks of definitions as global, one can use the package option `globaldef`. To that end, enclose the blocks by `\mpostsetup{globaldef=true}` and `\mpostsetup{globaldef=false}`.

Include Files. Alternatively, global definitions can be saved to an include file. This may make sense if there is a large amount of definitions which should not be written to each and every metapost file. An include file is declared by `\mpostfile[include]{`*filename*`}` and will be used by all subsequent main metapost files. Include files are equivalent to global definitions, but they cannot contain TEX definitions. Moreover they must not contain figures.

2.5 Immediate Processing

By default, the metapost file is compiled at the end of the LaTeX document. Therefore all new figures and any changes to existing figures are not reflected in the compiled LaTeX document after the first pass. A second LaTeX pass is needed to generate the desired output. Moreover, any change to the sequence of figures (figures inserted, deleted or moved) can only be seen after the second pass; the first pass shows the old sequence.

In the following we present some strategies to minimise or avoid this issue.

Collecting Figures at the Top. One option is to declare all figures with labels near the top of the document, and close the metapost file by the command `\mpostdone`. The compiled figures are immediately available afterwards. A side effect is that metapost and LaTeX source is necessarily separated (which may be considered a benefit or a drawback depending on philosophy). Note that in a setup with multiple metapost files, the figures can be defined at the top of each section thereby allowing some association between figures and manuscript.

Stable Filenames. A change of figure sequence can have irritating effects because some figures are displayed in the wrong place in the first pass. Even though a second pass resolves this issue, the first pass may have upset the page composition so that some labels containing page numbers can be messed up in the second pass thus requiring a third pass. There are some methods to reduce this effect:

- Provide a filename for each figure by the option `file`.
- Automatically align the figure filenames with the labels by the package option `labelnames`.
- Use the package option `numberwithin` with the counter of the top-level section. Changes of the figure order will then be contained within each section.
- Use one metapost file for each top-level section. Changes of the figure order will then be contained within each metapost file.

Immediate Processing. The above strategies are workarounds, but the package also provides a mechanism to compile individual figures or all figures at once. The drawback is that one metapost file with all relevant definitions needs to be generated and compiled for each figure. This may have some negative impact on performance for very large documents with excessively many figures, if many such documents are to be generated in a row, if the computer is slow or busy otherwise, or if the laptop battery is low on a long journey (or eventually if the user is old-fashioned or paranoid).

The mechanism is enabled by the package option `now`. While composing or editing individual figures, one can specify the figure option `now`, and remove it when the figure is final. Alternatively one can activate immediate processing for all figures with the package option `nowall` (together with `now`). Ordinarily, a temporary file is used for the metapost source for immediate processing. If the metapost sources for individual figures are needed, the package option `nowkeep` stores the metapost source for every generated figure *filename*`.mps` in the file *filename*`.mp`. When the figures are in a final state, all immediate processing can be disabled by turning off the package option `now`.

Note that changing some variables within the figure blocks can have undesired effects for immediate processing: In immediate processing only the present figure is included in the metapost file, otherwise the metapost file contains the sequence of all figures. Therefore, a change or (re)definition of a variable within a figure block will not be visible to all subsequent figures in immediate mode! To avoid such situations:

- Do not (re)define variables within a figure block. Use a definition block before the figure instead.
- Provide an initial value for all used variables which could change within other figures.
- For internal variables (such as the size `ahlength` of an arrow head) you can `save` the initial value before the first redefinition within a figure block. Alternatively you can prepend the first change to the variable by `interim`. Note that you can declare variables as internal by means of `newinternal`.

2.6 Interaction with Other Packages

The following lists some potential issues in the interaction with other LaTeX packages:

Package inputenc. This package should not interfere with the input encoding selected via the package inputenc, e.g. `utf8` encoding. Extended characters are passed on unchanged by the `mpostfig` environment. If you declare an input encoding for your LaTeX source by:

```
\usepackage[enc]{inputenc}
```

you should select the *same* input encoding for the labels in the metapost figures by means of:

```
\begin{mpostdef}[tex]
\usepackage[enc]{inputenc}
\end{mpostdef}
```

Package beamer. The package beamer is a popular package for preparing slideshow presentations. In particular, slides can be presented in several steps by means on an overlay mechanism. To that end the `frame` environment saves the enclosed block and processes it in several passes. Therefore, the environments `mpostfig` and `mpostdef` must not be used within the `frame` environment. Instead, figures should be declared outside the `frame` environment and can be displayed by `\mpostuse` within the `frame` environment.

Package graphbox. Figures are eventually displayed by the `\includegraphics` command which aligns the graphics with the bottom of the current line. To achieve different alignments or placements takes some efforts. The package graphbox extends the optional arguments of `\includegraphics` to customise the alignment conveniently. Since graphics arguments are passed on directly to `\includegraphics` the graphbox package can be used without restrictions. For instance, to align a figure vertically with the centre of the line, you may use `\begin{mpostfig}[opt={align}]` or `\mpostuse[align]{`*label*`}`.

Package latexmp. The METAPOST package latexmp writes its own LaTeX structure to the metapost file. Therefore mpostinl must not write the LaTeX structure, but still use the `latex` compiler. Furthermore latexmp needs two METAPOST passes. The required options are:

```
\mpostsetup{latex=false,format,compiler=latex,twice}
```

Script mplatex. The METAPOST processing script mplatex expects the default metapost file naming convention (*filename*`.`*nn*) and no `prologues` statement. Moreover, it provides the LaTeX structure. The required options are:

```
\mpostsetup{latex=false,format,prologues}
\mpostsetup{template={\mpostfilename.#1}}
\mpostsetup{command={mplatex opt}}
```

In *opt* you can specify command line options for `mplatex`, e.g. to include LaTeX packages. Make sure to enable full access to external programs, and that `mplatex` is in the path (or specify its location explicitly). Note that the file naming convention (*filename*`.`*nn*) does not seem to work with pdfLaTeX.

3 Information

3.1 Copyright

This work consists of the files `README.txt`, `mpostinl.ins` and `mpostinl.dtx` as well as the derived files `mpostinl.sty`, `mpinlsmp.tex` and `mpostinl.pdf`.

3.2 Files and Installation

The package consists of the files:

`README.txt`	readme file
`mpostinl.ins`	installation file
`mpostinl.dtx`	source file
`mpostinl.sty`	package file
`mpinlsmp.tex`	sample file
`mpostinl.pdf`	manual

The distribution consists of the files `README.txt`, `mpostinl.ins` and `mpostinl.dtx`.

- Run (pdf)LaTeX on `mpostinl.dtx` to compile the manual `mpostinl.pdf` (this file).
- Run LaTeX on `mpostinl.ins` to create the package `mpostinl.sty` and the sample `mpinlsmp.tex`. Copy the file `mpostinl.sty` to an appropriate directory of your LaTeX distribution, e.g. *texmf-root*`/tex/latex/mpostinl`.

3.3 Interaction with Other Packages and Software

The package relies on other packages and software:

- This package relies on some functionality of the package verbatim to read verbatim code from the LaTeX source without expansion of macros. Compatibility with the verbatim package has been tested with v1.5q (2014/10/28).
- This package uses the package graphicx from the graphics bundle to include graphics files. Compatibility with the graphicx package has been tested with v1.0g (2014/10/28).
- This package uses the package keyval from the graphics bundle to process the options for the package, environments and macros. Compatibility with the keyval package has been tested with v1.15 (2014/10/28).
- This package uses the command `\currfilename` provided by the package currfile (if available and loaded) to indicate the LaTeX source file in the generated metapost file. Compatibility with the currfile package has been tested with v0.7c (2015/04/23).
- The package assumes a TeX installation with METAPOST configured appropriately. Recent texlive and MiKTeX distributions should work well. Compatibility with the texlive distribution has been tested with the 2016 issue containing pdfTeX version 3.14159265-2.6-1.40.17 and METAPOST version 1.9991.

3.4 Feature Suggestions

The following is a list of features which may be useful for future versions of this package:

- A method to expand LaTeX macros to the metapost code to match the corresponding feature of package gmp: One difficulty is that direct insertion of LaTeX macros requires a substantially different implementation of the block scanning method because the method supplied by verbatim does not work. Furthermore, some escape mechanism is required either to expand selected LaTeX macros or to prevent their expansion (within `btex` ... `etex` blocks).
- A simpler approach to the above feature would be to implement a command or option to prepend some metapost statements to a given metapost block where LaTeX macros are expanded. This may work well for passing very few macros to metapost variables. However, one would have to decide how aggressively macros are to be expanded.
- Write generated files to a subdirectory or remove files not needed after processing. However, both appear to be somewhat in contrast to the usual TeX philosophy.

3.5 Revision History

v1.21: 2018/01/17

- remark on leading empty lines added
- manual rearranged

v1.2: 2018/01/05

- new option `checksum` to compile only modified files (requires the pdfTEX engine, otherwise disabled)
- warn if some figures may not be up to date
- bugfix for example if graphbox is not available

v1.12: 2017/06/24

- bugfix for blank `mpostfig` display due to changes in the basic LATEX or graphicx system

v1.11: 2017/04/01

- implementation of `fonts` option repaired

v1.1: 2017/02/27

- improved compatibility with package latexmp and script mplatex (thanks to Walter Entenmann for encouragement and testing)
- options improved to fine-tune structures written to metapost files and to specify compiler programs

v1.0: 2017/01/04

- manual and install package
- first version published on CTAN

v0.7–0.9: 2016/11/15 – 2017/01/04

- package options
- metapost environments simplified and renamed
- immediate and multiple file processing
- customisation options
- internal buffer processing

v0.6–0.63: 2015/07/11 – 2016/10/03

- minor improvements

v0.5: 2010/11/01

- basic functionality

A Sample File

In this section we provide a LATEX example how to use some of the mpostinl features.

A.1 Preamble

Standard document class:

```
\documentclass[12pt,a4paper]{article}
```

Adjust the paragraph shape:

```
\parindent0pt
\parskip6pt
```

Include the mpostinl package, include METAFONT fonts in the metapost figures:

```
\usepackage[fonts=true]{mpostinl}
```

We will test labels in UTF-8, so include package inputenc:

```
\usepackage[utf8]{inputenc}
```

Include packages currfile and graphbox if available, declare dummy option `align` for `\includegraphics` if graphbox is not available:

```
%% optional: add filename to position labels in metapost code
\IfFileExists{currfile.sty}{\usepackage{currfile}}{}
%% optional: tools to align graphics
\makeatletter\define@key{Gin}{align}[]{}\makeatother
\IfFileExists{graphbox.sty}{\usepackage{graphbox}}{}
```

Enable immediate mode and line number indicators, prepare some mpostinl options for testing:

```
%% some sample package options:
%% \mpostsetup{write=false}
%% \mpostsetup{compile=false}
\mpostsetup{now}
%% \mpostsetup{nowall}
\mpostsetup{lineno}
%% \mpostsetup{latex=false}
%% \mpostsetup{classopt={12pt}}
```

Include the package inputenc for preparing LaTeX labels within the metapost figures; as we will be generating several metapost files later on, make sure this statement in included in all of them:

```
%% declare packages to be used for processing labels:
\begin{mpostdef}[tex,global]
\usepackage[utf8]{inputenc}
\end{mpostdef}
```

Define an internal variable `unit` and initialise to 1cm; as we will be generating several metapost files later on, make sure this statement in included in all of them:

```
%% specify global definitions:
\begin{mpostdef}[global]
newinternal unit;
unit:=1cm;
\end{mpostdef}
```

Begin document body:

```
\begin{document}
```

A.2 Basic Functionality

We start by demonstrating the basic functionality of the package:

```
\section{Basic Functionality}
```

First, draw a circle of diameter 1cm and write a '1' in the centre:

```
a plain circle:\\
\begin{mpostfig}
draw fullcircle scaled unit;
label(btex 1 etex, (0,0));
\end{mpostfig}
```

Use the options for `\includegraphics`. Draw another circle containing a '2' and scale it by factor 1.5. Also vertically align to the centre (if graphbox is available):

```
scaled (and aligned to centre if available):\\
X
\begin{mpostfig}[opt={scale=1.5,align}]
draw fullcircle scaled unit;
label(btex 2 etex, (0,0));
\end{mpostfig}
X
```

Declare a figure with label `fig`, do not show:

```
declare figure with label (no display).
\begin{mpostfig}[label={fig}]
draw fullcircle scaled unit;
label(btex 3 etex, (0,0));
\end{mpostfig}
```

Display the figure:

```
display:\\
\mpostuse{fig}
```

Display the figure with options for `\includegraphics`:

```
display with options:\\
\mpostuse[scale=1.5,align]{fig}
```

Display the figure within a box:

```
display in a box:\\
\fbox{\mpostuse{fig}}
```

Display the figure at the centre of a line:

```
centred display:
\begin{center}
\mpostuse[scale=1.5]{fig}
\end{center}
```

Display the figure within an equation:

```
display in equation (align if possible):
\begin{equation}
\mpostuse[scale=1.5,align]{fig}
\end{equation}
```

Show the filename of the figure:

```
filename: \mpostgetname{fig}
\texttt{\mpostfigurename}
```

Declare a figure with filename, label and show it, then display filename:

```
figure with filename:\\
\begin{mpostfig}[file={\jobname-name.mps},label={name},show]
draw fullcircle scaled unit;
label(btex 4 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{name}
\texttt{\mpostfigurename}
```

Display the file via `\includegraphics`. As the figure may not exist (in the first pass) check for existence first to avoid a compile error:

```
display by \verb+\includegraphics+ (if file exists):\\
\IfFileExists{\jobname-name.mps}{\includegraphics{\jobname-name.mps}}{}
```

Show a figure which does not exist, triggers a warning and displays a box:

```
label does not exist:\\
\mpostuse{notexist}
```

Display a figure with a label containing special characters in UTF-8. Note that the internal variable `unit` is changed locally (`interim`) so that subsequent figures will see the old value:

```
utf-8 test:\\
\begin{mpostfig}
interim unit:=1.5cm;
draw fullcircle scaled unit;
label(btex àáâãäåæ etex, (0,0));
\end{mpostfig}
```

A.3 Immediate Processing

Next, we demonstrate immediate processing (make sure to enable the package option `now`):

```
\section{Immediate Processing}
```

Declare a figure for immediate processing. You may change the size `unit` and the figure will be adjusted after the first LaTeX pass:

```
immediate processing per figure:\\
\begin{mpostfig}[now]
interim unit:=1.5cm;
draw fullcircle scaled unit;
label(btex 5 etex, (0,0));
\end{mpostfig}
```

One can also enable immediate processing for all subsequent figures. Furthermore, store the metapost source in individual files:

```
turn on immediate processing for all figures and keep sources.
\mpostsetup{nowall,nowkeep}
```

Now declare a figure with individual metapost source, display via `\includegraphics`, and confirm that source exists:

```
generate file \texttt{\jobname-now.mps}:
```

```
\begin{mpostfig}[file={\jobname-now.mps}]
draw fullcircle scaled unit;
label(btex 6 etex, (0,0));
\end{mpostfig}
\\
display by \verb+\includegraphics+:\\
\includegraphics{\jobname-now.mps}
\\
source \texttt{\jobname-now.mp}
\IfFileExists{\jobname-now.mp}{exists}{does not exist}.
```

Reset immediate processing options for further examples (normally not necessary):

```
turn off immediate processing and discard sources.
\mpostsetup{nowall=false,nowkeep=false}
```

A.4 Filename Composition

Now, we demonstrate how to adjust the composition of figure filenames:

```
\section{Filename Composition}
```

Use the figure label instead of a figure counter:

```
by label:\\
\mpostsetup{labelnames=true}
\begin{mpostfig}[label={circle},show]
draw fullcircle scaled unit;
label(btex 7 etex, (0,0));
\end{mpostfig}
\mpostsetup{labelnames=false}
\\
filename: \mpostgetname{circle}
\texttt{\mpostfigurename}
```

Change extension to `.eps`:

```
change extension:
\mpostsetup{extension=eps}
\begin{mpostfig}[label={ext}]
draw fullcircle scaled unit;
label(btex 8 etex, (0,0));
\end{mpostfig}
\mpostsetup{extension=mps}
\\
filename: \mpostgetname{ext}
\texttt{\mpostfigurename}
```

Change template altogether:

```
change template:\\
\mpostsetup{template=\mpostfilename-figure-#1.mps}
\begin{mpostfig}[label={template},show]
draw fullcircle scaled unit;
label(btex 9 etex, (0,0));
\end{mpostfig}
\mpostsetup{template=\mpostfilename-#1.mps}
\\
filename: \mpostgetname{template}
\texttt{\mpostfigurename}
```

Demonstrate numbering within a section (here: `subsection`)

```
\mpostsetup{numberwithin=subsection}
```

One figure in the first section:

```
\subsection{Section 1}

\begin{mpostfig}[show,label={sec1}]
draw fullcircle scaled unit;
label(btex 10 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec1}
\texttt{\mpostfigurename}
```

Two figures in the second section:

```
\subsection{Section 2}

\begin{mpostfig}[show,label={sec2}]
draw fullcircle scaled unit;
label(btex 11 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec2}
\texttt{\mpostfigurename}

\begin{mpostfig}[show,label={sec3}]
draw fullcircle scaled unit;
label(btex 12 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec3}
\texttt{\mpostfigurename}
```

One figure in the third section:

```
\subsection{Section 3}

\begin{mpostfig}[show,label={sec4}]
draw fullcircle scaled unit;
label(btex 13 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec4}
\texttt{\mpostfigurename}
```

Reset numbering for further examples (normally not necessary, setting cannot be undone completely):

```
\makeatletter
\def\thempi@count{\arabic{mpi@count}}
\makeatother
```

A.5 Multiple Files

Finally, demonstrate the generation and usage of several metapost files:

```
\section{Multiple Files}
```

First, close the old metapost file to get to a clean state:

```
\mpostdone
```

Make a definition which will apply to all metapost files. `global` option can be activated for a block of definitions (here only one):

```
\mpostsetup{globaldef=true}

\begin{mpostdef}
unit:=0.8cm;
\end{mpostdef}

\mpostsetup{globaldef=false}
```

Alternatively, we can define an include file to be included by all main metapost files. This has the same effect as a global definition, but is intended for long definitions:

```
\mpostfile[include]{\jobname-inc}

\begin{mpostdef}
def drawcircle =
draw fullcircle scaled unit
enddef;
\end{mpostdef}

\mpostdone
```

The first file contains one figure:

```
\subsection*{Section 1}
\mpostfile{\jobname-sec1}

\begin{mpostfig}[show,label={sec5}]
drawcircle;
label(btex 14 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec5}
\texttt{\mpostfigurename}

\mpostdone
```

The second file contains a local redefinition of `unit` and two figures:

```
\subsection*{Section 2}
\mpostfile{\jobname-sec2}

\begin{mpostfig}[show,label={sec6}]
drawcircle;
label(btex 15 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec6}
\texttt{\mpostfigurename}

\begin{mpostdef}
unit:=1.5cm;
\end{mpostdef}

\begin{mpostfig}[show,label={sec7}]
```

```
drawcircle;
label(btex 16 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec7}
\texttt{\mpostfigurename}

\mpostdone
```

The third file contains one figure. Note that the redefinition of the second file does not apply here:

```
\subsection*{Section 3}
\mpostfile{\jobname-sec3}

\begin{mpostfig}[show,label={sec8}]
drawcircle;
label(btex 17 etex, (0,0));
\end{mpostfig}
\\
filename: \mpostgetname{sec8}
\texttt{\mpostfigurename}

\mpostdone
```

End of document body:

```
\end{document}
```

B Implementation

In this section we describe the package `mpostinl.sty`.

Required Packages. The package loads the packages verbatim, graphicx and keyval if not yet present. verbatim is used for reading verbatim metapost code. graphicx is used for including graphics files. keyval is used for extended options processing.

```
\RequirePackage{verbatim}
\RequirePackage{graphicx}
\RequirePackage{keyval}
```

Package Options.

`\mpostfilename` `\mpi@nowname` `\mpi@extension` `\mpi@template`

`\mpostfilename` stores the metapost filename, `\mpi@nowname` stores the filename for immediate processing, and `\mpi@template` is the template to generate the figure filenames:

```
\def\mpostfilename{\jobname}
\def\mpi@nowname{\jobname-tmp}
\def\mpi@extension{mps}
\def\mpi@template#1{\mpostfilename-#1%
  \ifx\mpi@extension\mpi@empty\else.\fi\mpi@extension}
```

`mpi@count`

Declare a counter for figure filenames:

```
\newcounter{mpi@count}
\def\thempi@count{\arabic{mpi@count}}
```

The package has some boolean keyval options which can be set to `true` or `false`.

```
\newif\ifmpi@draft\mpi@draftfalse
\newif\ifmpi@latex\mpi@latextrue
\newif\ifmpi@fonts\mpi@fontsfalse
\newif\ifmpi@write\mpi@writetrue
\newif\ifmpi@compile\mpi@compiletrue
\newif\ifmpi@twice\mpi@twicefalse
\newif\ifmpi@lineno\mpi@linenofalse
\newif\ifmpi@labelnames\mpi@labelnamesfalse
\newif\ifmpi@nowactive\mpi@nowactivefalse
\newif\ifmpi@now\mpi@nowfalse
\newif\ifmpi@nowkeep\mpi@nowkeepfalse
\newif\ifmpi@include\mpi@includefalse
\newif\ifmpi@defglobal\mpi@defglobalfalse
\newif\ifmpi@checksum\mpi@checksumtrue
```

`\mpi@mpostmem` `\mpi@mpostcompiler` `\mpi@latexclass` `\mpi@latexoptions` `\mpi@documentclass`

These definitions store the options for processing labels via TEX or LATEX:

```
\def\mpi@mpostmem{}
\def\mpi@mpostcompiler{}
\def\mpi@latexclass{article}
\def\mpi@latexoptions{}
\def\mpi@documentclass{\@backslashchar documentclass%
  \mpi@latexoptions{\mpi@latexclass}}
```

`\mpi@warncompile`

Warn and disable compiling if `\write18` is unavailable:

```
\def\mpi@warncompile{\ifmpi@compile\ifeof18%
  \PackageWarning{mpostinl}{write18 disabled, %
    manual metapost compiling required}{}%
  \global\mpi@compilefalse\fi\fi}
```

Process package options:

```
\def\mpi@group{mpi@}
\DeclareOption{final}{\mpi@draftfalse}
\define@key{\mpi@group}{draft}[true]{\csname mpi@draft#1\endcsname}
\define@key{\mpi@group}{write}[true]{\csname mpi@write#1\endcsname}
\define@key{\mpi@group}{latex}[true]{\csname mpi@latex#1\endcsname}
\define@key{\mpi@group}{compile}[true]{\csname mpi@compile#1\endcsname}
\define@key{\mpi@group}{twice}[true]{\csname mpi@twice#1\endcsname}
\define@key{\mpi@group}{checksum}[true]{\csname mpi@checksum#1\endcsname}
\define@key{\mpi@group}{fonts}[true]{\csname mpi@fonts#1\endcsname}
\define@key{\mpi@group}{prologues}[]{\def\mpi@prologues{#1}}
\define@key{\mpi@group}{lineno}[true]{\csname mpi@lineno#1\endcsname}
\define@key{\mpi@group}{labelnames}[true]{\csname mpi@labelnames#1\endcsname}
\define@key{\mpi@group}{compiler}[]{\def\mpi@texcompiler{#1}}
\define@key{\mpi@group}{format}[]{\def\mpi@texformat{#1}}
\define@key{\mpi@group}{mem}[]{\def\mpi@mpostmem{#1}}
\define@key{\mpi@group}{command}[]{\def\mpi@mpostcompiler{#1}}
\define@key{\mpi@group}{class}{\def\mpi@latexclass{#1}}
\define@key{\mpi@group}{classopt}[]{\def\mpi@latexoptions{[#1]}}
\define@key{\mpi@group}{now}[true]{\csname mpi@nowactive#1\endcsname}
\define@key{\mpi@group}{nowall}[true]{\csname mpi@now#1\endcsname}
\define@key{\mpi@group}{nowkeep}[true]{\csname mpi@nowkeep#1\endcsname}
\define@key{\mpi@group}{globaldef}[true]{\csname mpi@defglobal#1\endcsname}
\define@key{\mpi@group}{extension}[]{\def\mpi@extension{#1}}
\define@key{\mpi@group}{template}{\def\mpi@template##1{#1}}
```

```
\define@key{\mpi@group}{numberwithin}{%
  \@addtoreset{mpi@count}{#1}%
  \def\thempi@count{\arabic{#1}-\arabic{mpi@count}}%
}
```

Pass undeclared options on to keyval processing:

```
\DeclareOption*{\expandafter\setkeys\expandafter\mpi@group%
  \expandafter{\CurrentOption}}
```

Process package options, warn if `\write18` mechanism is not available, and disable checksum if `\pdfmdfivesum` is not available:

```
\ProcessOptions
\mpi@warncompile
\ifdefined\pdfmdfivesum\else\mpi@checksumfalse\fi
```

Internal Commands and Definitions.

`\mpi@empty` Define an empty macro for comparison via `\ifx`:

```
\def\mpi@empty{}
```

`\mpi@dblquotchar` Define a bare double quotation character for writing to the file:

```
\begingroup\catcode`\"=12\relax\gdef\mpi@dblquotchar{"}\endgroup
```

`\ifmpi@infile` `\ifmpi@inbody` `\ifmpi@infile` indicates whether a file is open, `\ifmpi@inbody` indicates whether the content section has started:

```
\newif\ifmpi@infile\mpi@infilefalse
\newif\ifmpi@inbody\mpi@inbodyfalse
```

`\ifmpi@inclmod` `\ifmpi@filemod` `\ifmpi@inclmod` indicates whether an include file has been modified, `\ifmpi@filemod` indicates whether the current file is modified:

```
\newif\ifmpi@inclmod\mpi@inclmodfalse
\newif\ifmpi@filemod
```

`\ifmpi@warnmod` `\ifmpi@showinfile` `\ifmpi@warnmod` indicates whether a rerun warning is to be issued, `\ifmpi@showinfile` indicates whether a figure has been displayed while a file is being composed:

```
\newif\ifmpi@warnmod\mpi@warnmodfalse
\newif\ifmpi@showinfile
```

`\mpi@out` `\mpi@outnow` File handles for the metapost file (`\mpi@out`) and for immediate output (`\mpi@outnow`):

```
\newwrite\mpi@out
\newwrite\mpi@outnow
```

`\mpi@writebuf` Write to the file:

```
\def\mpi@writebuf{\ifmpi@write\immediate\write\mpi@out{\the\mpi@buf}\fi}
```

`\mpi@writenow` Write to the immediate buffer:

```
\def\mpi@writenow{\ifmpi@nowactive\mpi@addtoexp\mpi@nowbuf{\the\mpi@buf^^J}\fi}
```

\mpi@buf
\mpi@defbuf
\mpi@nowbuf

Declare three token buffers to store the current block (\mpi@buf), global definitions (\mpi@defbuf) and the definitions for immediate processing (\mpi@nowbuf):

```
\newtoks\mpi@buf
\newtoks\mpi@defbuf
\newtoks\mpi@nowbuf
\mpi@defbuf={}
```

\mpi@addto
\mpi@addtoexp

\mpi@addto adds the second argument to a global token buffer without expansion. \mpi@addtoexp first expands the second argument (once) and adds it to the token buffer:

```
\def\mpi@addto#1#2{\global#1=\expandafter{\the#1#2}}
\def\mpi@addtoexp#1#2{\expandafter\mpi@addto\expandafter#1\expandafter{#2}}
```

\mpi@clearbuf
\mpi@addbufexp
\mpi@addbuf

\mpi@clearbuf clears the current block buffer. \mpi@addbufexp expands (once) and adds to the current block buffer. \mpi@addbuf adds to the current block buffer via \protected@edef:

```
\def\mpi@clearbuf{\global\mpi@buf={}}
\def\mpi@addbufexp#1{\mpi@addtoexp\mpi@buf{#1^^J}}
\def\mpi@addbuf#1{{\protected@edef\mpi@tmp{#1}\mpi@addbufexp\mpi@tmp}}
```

\mpi@stripext

Strip .mps or .eps ending of a figure filename, return result in \mpi@stripped:

```
\def\mpi@stripext#1{\edef\mpi@tmp{#1}\expandafter%
  \mpi@stripstart\expandafter{\mpi@tmp}}
\def\mpi@ifeq#1#2#3#4{\def\mpi@tmpa{#1}\def\mpi@tmpb{#2}%
  \ifx\mpi@tmpa\mpi@tmpb#3\else#4\fi}
\def\mpi@stripstart#1{\mpi@stripfor{\@gobble}#1.\@@.}
\def\mpi@stripfor#1#2.#3.{%
  \begingroup%
  \mpi@ifeq{#3}{\@@}{%
    \def\mpi@tmp{\def\mpi@stripped{#1.#2}}%
    \mpi@ifeq{#1}{\@gobble}{}{%
      \mpi@ifeq{#2}{eps}{\def\mpi@tmp{\def\mpi@stripped{#1}}}{}%
      \mpi@ifeq{#2}{mps}{\def\mpi@tmp{\def\mpi@stripped{#1}}}{}%
      \ifx\mpi@extension\mpi@empty\else%
        \expandafter\mpi@ifeq\expandafter{\mpi@extension}{#2}%
          {\def\mpi@tmp{\def\mpi@stripped{#1}}}{}%
      \fi%
    }%
  }{\def\mpi@tmp{\mpi@stripfor{#1.#2}#3.}}%
  \expandafter\endgroup\mpi@tmp%
}
```

\mpi@warnmod

Warn if some figure has been displayed while a file has been written (potentially the displayed figure is old):

```
\newcommand{\mpi@warnmod}{%
  \ifmpi@checksum\ifmpi@warnmod%
    \PackageWarning{mpostinl}{figure(s) may have changed. %
      Rerun to update figures}{}%
  \fi\fi%
}
```

\mpostplaceholder

Display a placeholder for non-existing files or labels; this function may be overwritten by the user for customisation purposes (optional argument contains either '`file`' or '`label`'):

```
\newcommand{\mpostplaceholder}[2][]{\parbox[c]{1in}{%
```

```
  \hrule\vrule\hfill%
  \parbox[c]{0pt}{\rule{0cm}{0.6in}}\makebox[0pt][c]{\scriptsize\tt #2}%
  \hfill\vrule\hrule}}
```

`\mpi@graphics` Display a figure; if the file does not exist (yet) issue a warning and display a placeholder, otherwise expand filename properly and pass on to `\includegraphics`:

```
\newcommand{\mpi@graphics}[2][]{%
  \IfFileExists{#2}%
    {\edef\mpi@tmp{#2}\includegraphics[#1]{\mpi@tmp}}%
    {\typeout{graphics file `#2' missing}\mpostplaceholder[file]{#2}}%
}
```

`\mpi@verbatim` Start reading the block from the source file using the verbatim package; add each line to the buffer:

```
\newcommand{\mpi@verbatim}{%
  \begingroup%
  \@bsphack%
  \let\do\@makeother\dospecials%
  \catcode`\^^M\active%
  \def\verbatim@processline{\mpi@addbufexp{\the\verbatim@line}}%
  \verbatim@start%
}
```

`\mpi@endverbatim` End reading the block from the source file:

```
\newcommand{\mpi@endverbatim}{%
  \@esphack%
  \endgroup%
}
```

`\mpi@putlineno` Write current position in source file to buffer; write line number and source file name (if available via package currfile):

```
\newcommand{\mpi@putlineno}{%
  \ifmpi@lineno%
    \mpi@addbuf{\@percentchar----------------------------------------}%
    \mpi@addbuf{\@percentchar%
      \ifx\currfilename\@undefined\else\currfilename\space\fi%
      l.\the\inputlineno}%
  \fi%
}
```

`\mpi@beginfig` Write beginning of figure block to buffer; write filename and `beginfig` statement:

```
\newcommand{\mpi@beginfig}[1]{%
  \mpi@addbuf{filenametemplate \mpi@dblquotchar#1\mpi@dblquotchar;}%
  \mpi@addbuf{beginfig(\arabic{mpi@count})}%
}
```

`\mpi@endfig` Write end of figure block to buffer; write `endfig` statement:

```
\newcommand{\mpi@endfig}{%
  \mpi@addbuf{endfig;}%
}
```

`\mpi@declaredoc` Write `\documentclass` statement in LaTeX mode to buffer:

```
\newcommand{\mpi@declaredoc}{%
  \ifmpi@latex%
    \mpi@addbuf{verbatimtex}%
    \mpi@addbuf{\mpi@documentclass}%
    \mpi@addbuf{etex}%
    \mpi@addbuf{}%
  \fi%
}
```

`\mpi@begindoc` Write beginning of content section to buffer; write `\begin{document}` statement in LaTeX mode:

```
\newcommand{\mpi@begindoc}{%
  \ifmpi@latex%
    \mpi@putlineno%
    \mpi@addbuf{verbatimtex}%
    \mpi@addbuf{\@backslashchar begin{document}}%
    \mpi@addbuf{etex}%
  \fi%
}
```

`\mpi@enddoc` Write end of content section to buffer; write `\end{document}` statement in LaTeX mode:

```
\newcommand{\mpi@enddoc}{%
  \ifmpi@latex%
    \mpi@putlineno%
    \mpi@addbuf{verbatimtex}%
    \mpi@addbuf{\@backslashchar end{document}}%
    \mpi@addbuf{etex}%
  \fi%
}
```

`\mpi@declareformat` Write TeX format specifier to buffer:

```
\newcommand{\mpi@declareformat}{%
  \let\mpi@tmp\mpi@texformat%
  \ifx\mpi@tmp\@undefined\def\mpi@tmp{\ifmpi@latex latex\else tex\fi}\fi%
  \ifx\mpi@tmp\mpi@empty\else%
    \mpi@addbuf{verbatimtex}%
    \mpi@addbuf{\@percentchar &\mpi@tmp}%
    \mpi@addbuf{etex}%
    \mpi@addbuf{}%
  \fi%
}
```

`\mpi@composehead` Write file header to buffer; declare font inclusion and draft mode, write TeX format specifier and LaTeX header:

```
\newcommand{\mpi@composehead}{%
  \mpi@putlineno%
  \let\mpi@tmp\mpi@prologues%
  \ifx\mpi@tmp\@undefined\def\mpi@tmp{\ifmpi@fonts 3\else 2\fi}\fi%
  \ifx\mpi@tmp\mpi@empty\else%
    \mpi@addbuf{prologues:=\mpi@tmp;}%
  \fi%
  \ifmpi@draft\mpi@addbuf{draft:=1;}\fi%
  \mpi@addbuf{}%
  \mpi@declareformat%
  \mpi@declaredoc%
```

```
}
```

`\mpi@beginfile` Write beginning of file to buffer; write generated file comment and header:

```
\newcommand{\mpi@beginfile}{%
  \ifx\mpi@mpostmem\mpi@empty\else%
    \mpi@addbuf{\@percentchar &\mpi@mpostmem}%
  \fi%
  \mpi@addbuf{\@percentchar generated from file `\jobname' by mpostinl.sty}%
  \ifmpi@include\else%
    \mpi@composehead%
    \mpi@addbufexp{\the\mpi@defbuf}%
  \fi%
}
```

`\mpi@endfile` Write end of file to buffer; write `end` statement:

```
\newcommand{\mpi@endfile}{%
  \mpi@putlineno%
  \ifmpi@include\else%
    \mpi@addbuf{end}%
  \fi%
}
```

`\mpi@getchecksum` Compute the MD5 checksum for a metapost file and store in macro:

```
\newcommand{\mpi@getchecksum}[2]{%
  \IfFileExists{#2}{\xdef#1{\pdfmdfivesum file{#2}}}{\global\let#1=\@undefined}%
}
```

`\mpi@startfile` Start a new file if not already open:

```
\newcommand{\mpi@startfile}{%
  \ifmpi@infile\else%
```

Prevent reopening and overwriting the previous file:

```
    \ifx\mpostfilename\mpi@empty%
      \PackageError{mpostinl}{no filename provided to write to}{}%
    \fi%
```

Compute checksum of old file:

```
    \ifmpi@checksum\ifmpi@inclmod\else%
      \mpi@getchecksum{\mpi@checksum@before}{\mpostfilename.mp}%
    \fi\fi%
```

Open file for writing, prepare and write header to file:

```
    \global\mpi@infiletrue%
    \global\mpi@showinfilefalse%
    \ifmpi@write\immediate\openout\mpi@out\mpostfilename.mp\fi%
    \mpi@clearbuf%
    \mpi@beginfile%
    \mpi@writebuf%
```

For include files, write `input` statement to definition buffer so that the file will be included by all main files:

```
    \ifmpi@include%
      \mpi@clearbuf%
```

```
      \mpi@putlineno%
      \mpi@addbuf{input \mpostfilename}%
      \mpi@addtoexp\mpi@defbuf{\the\mpi@buf^^J}%
```

If immediate mode is available fill immediate buffer with header:

```
    \else%
      \global\mpi@nowbuf={}%
      \mpi@writenow%
    \fi%
  \fi%
}
```

`\mpi@startcontent` Start content section of file; make sure the file is open, prepare start of content section and write to file, if immediate mode is available also add to immediate buffer:

```
\newcommand{\mpi@startcontent}{%
  \mpi@startfile%
  \ifmpi@inbody\else%
    \global\mpi@inbodytrue%
    \mpi@clearbuf%
    \mpi@begindoc%
    \mpi@writebuf%
    \mpi@writenow%
  \fi%
}
```

`\mpi@compile` Compile the metapost file (if writing and compiling is enabled):

```
\newcommand{\mpi@compile}[1]{%
  \ifmpi@write\ifmpi@compile%
    \ifx\mpi@mpostcompiler\mpi@empty%
```

Compose command line for `mpost`; pass on interactionmode setting to METAPOST, specify mem filem, specify tex compiler:

```
      \def\mpi@imode{}%
      \ifcase\the\interactionmode%
        \def\mpi@imode{-interaction=batchmode}\or%
        \def\mpi@imode{-interaction=nonstopmode}\or%
        \def\mpi@imode{-interaction=scrollmode}\or%
        \def\mpi@imode{-interaction=errorstopmode}\fi%
      \let\mpi@texswitch\mpi@texcompiler%
      \ifx\mpi@texswitch\@undefined%
        \def\mpi@texswitch{\ifmpi@latex latex\else tex\fi}%
      \fi%
      \def\mpi@execute{mpost\space%
        \mpi@imode\space%
        \ifx\mpi@mpostmem\mpi@empty\else -mem=\mpi@mpostmem\space\fi%
        \ifx\mpi@texswitch\mpi@empty\else -tex=\mpi@texswitch\space\fi%
        #1}%
    \else%
```

Compose custom command:

```
      \def\mpi@execute{\mpi@mpostcompiler\space#1}%
    \fi%
```

Execute METAPOST by `\write18` command; do it again if needed:

```
    \immediate\write18{\mpi@execute}%
```

```
    \ifmpi@twice%
      \immediate\write18{\mpi@execute}%
    \fi%
  \fi\fi%
}
```

`\mpi@closefile` Close the file, compile and reset:

```
\newcommand{\mpi@closefile}{%
  \ifmpi@infile%
```

Write end of content section (if started):

```
    \mpi@clearbuf%
    \ifmpi@inbody%
      \mpi@enddoc%
      \mpi@addbuf{}%
    \fi%
```

Write end of file and close:

```
    \mpi@endfile%
    \mpi@writebuf%
    \ifmpi@write\immediate\closeout\mpi@out\fi%
```

Determine whether file has changed:

```
    \mpi@filemodtrue%
    \ifmpi@checksum\ifmpi@inclmod\else%
      \mpi@getchecksum{\mpi@checksum@after}{\mpostfilename.mp}%
      \ifx\mpi@checksum@before\mpi@checksum@after%
        \mpi@filemodfalse%
      \else%
        \ifmpi@include\mpi@inclmodtrue\fi%
      \fi%
    \fi\fi%
```

Compile if file has changed and if it contains figures. Activate warning if file has changed and figures were displayed during composition:

```
    \ifmpi@filemod\ifmpi@inbody\mpi@compile{\mpostfilename.mp}\fi\fi%
    \ifmpi@filemod\ifmpi@showinfile\global\mpi@warnmodtrue\fi\fi%
```

Reset variables:

```
    \global\mpi@infilefalse%
    \global\let\mpostfilename\mpi@empty%
    \global\mpi@inbodyfalse%
    \setcounter{mpi@count}{0}%
  \fi%
}
```

`\mpi@processnow` Write present figure to an individual file and process immediately:

```
\newcommand{\mpi@processnow}{%
  \ifmpi@nowactive\ifmpi@write\ifmpi@compile%
```

If immediate file is to be kept, use *filename*`.mp` as source for *filename*`.mps` output file.

```
    \ifmpi@nowkeep%
      \mpi@stripext{\mpi@figfile}%
      \edef\mpi@nowname{\mpi@stripped}%
    \fi%
```

Open immediate file, write the immediate buffer, the present figure and the end of file, close the file and compile:

```
    \immediate\openout\mpi@outnow\mpi@nowname.mp%
    \immediate\write\mpi@outnow{\the\mpi@nowbuf}%
    \immediate\write\mpi@outnow{\the\mpi@buf}%
    \mpi@clearbuf%
    \mpi@enddoc%
    \mpi@addbuf{}%
    \mpi@endfile%
    \immediate\write\mpi@outnow{\the\mpi@buf}%
    \immediate\closeout\mpi@outnow%
    \mpi@compile{\mpi@nowname.mp}%
  \fi\fi\fi%
}
```

Make sure to close and process the file at the end. Warn if figures may need another compiler pass:

```
\AtEndDocument{\mpi@closefile}
\AtEndDocument{\mpi@warnmod}
```

External Commands. The following commands are the interface of the package.

`\mpostsetup` `\mpostsetup` processes package options when the package has already been loaded:

```
\newcommand{\mpostsetup}[1]{%
  \setkeys\mpi@group{#1}%
  \mpi@warncompile%
  \ifdefined\pdfmdfivesum\else\mpi@checksumfalse\fi%
}
```

`mpostdef` Declare options for the `mpostdef` environment:

```
\newif\ifmpi@deftex
\define@key{mpi@def}{tex}[true]{\csname mpi@deftex#1\endcsname}
\define@key{mpi@def}{global}[true]{\csname mpi@defglobal#1\endcsname}
```

The environment `mpostdef` adds a block of definitions to the metapost file:

```
\newenvironment{mpostdef}[1][]{%
```

Process optional arguments:

```
  \mpi@deftexfalse%
  \setkeys{mpi@def}{#1}%
```

TEX definitions cannot be in an include file:

```
  \ifmpi@defglobal\else\ifmpi@deftex\ifmpi@include%
    \PackageWarning{mpostinl}{tex definitions within an include file %
      will be ignored by mpost; switching to global definition}{}%
    \mpi@defglobaltrue%
  \fi\fi\fi%
```

Prepare for recording; start the file if not open and not global, clear buffer, write current position, and add '`verbatimtex`' if in TEX mode:

```
  \ifmpi@defglobal\else%
    \mpi@startfile%
  \fi%
```

```
  \mpi@clearbuf%
  \mpi@putlineno%
  \ifmpi@deftex%
    \mpi@addbuf{verbatimtex}%
  \fi%
  \mpi@verbatim%
}%
```

Postprocessing; add '`etex`' if in TEX mode, add to appropriate buffer(s).

```
{%
  \mpi@endverbatim%
  \ifmpi@deftex%
    \mpi@addbuf{etex}%
  \fi%
  \ifmpi@defglobal%
    \mpi@addtoexp\mpi@defbuf{\the\mpi@buf^^J}%
    \ifmpi@include\else\ifmpi@infile%
      \mpi@writebuf%
      \mpi@writenow%
    \fi\fi%
  \else%
    \mpi@writebuf%
    \ifmpi@include\else\mpi@writenow\fi%
  \fi%
}
```

`mpostfig` Declare options for the `mpostfig` environment:

```
\newif\ifmpi@figshow
\define@key{mpi@fig}{show}[true]{\csname mpi@figshow#1\endcsname}
\define@key{mpi@fig}{twice}[true]{\csname mpi@twice#1\endcsname}
\define@key{mpi@fig}{file}{\def\mpi@figfile{#1}}
\define@key{mpi@fig}{label}{\def\mpi@figlabel{#1}}
\define@key{mpi@fig}{opt}{\def\mpi@figopt{[#1]}}
\define@key{mpi@fig}{now}[true]{\csname mpi@now#1\endcsname}
```

The environment `mpostfig` adds a figure to the metapost file:

```
\newenvironment{mpostfig}[1][]{%
```

Make sure that include files do not contain figures:

```
  \ifmpi@include%
    \PackageError{mpostinl}{cannot write figure to include file}{}%
  \fi%
```

Process optional arguments:

```
  \def\mpi@figfile{}%
  \def\mpi@figlabel{}%
  \def\mpi@figopt{}%
  \mpi@figshowfalse%
  \setkeys{mpi@fig}{#1}%
```

Display figure if no filename or label is provided:

```
  \ifx\mpi@figlabel\mpi@empty\ifx\mpi@figfile\mpi@empty\mpi@figshowtrue\fi\fi%
```

Compose filename from label (if desired and specified):

```
  \ifmpi@labelnames\ifx\mpi@figfile\mpi@empty\ifx\mpi@figlabel\mpi@empty\else%
```

```
    \edef\mpi@figfile{\mpi@template{\mpi@figlabel}}%
  \fi\fi\fi%
```

Compose filename from counter if no filename is provided:

```
  \ifx\mpi@figfile\mpi@empty%
    \addtocounter{mpi@count}{1}%
    \edef\mpi@figfile{\mpi@template{\thempi@count}}%
  \fi%
```

Save filename to label, warn if label has already been defined:

```
  \ifx\mpi@figlabel\mpi@empty\else%
    \expandafter\ifx\csname mpi@l@\mpi@figlabel\endcsname\relax\else%
      \PackageWarning{mpostinl}{label `\mpi@figlabel' already defined; %
        overwriting}{}%
    \fi%
    \expandafter\xdef\csname mpi@l@\mpi@figlabel\endcsname{\mpi@figfile}%
  \fi%
```

Prepare for recording; start file and content section (if needed), clear buffer, write current position, begin figure block:

```
  \mpi@startcontent%
  \mpi@clearbuf%
  \mpi@putlineno%
  \mpi@beginfig{\mpi@figfile}%
  \mpi@verbatim%
}%
```

Postprocessing; end figure block, add to buffer, process immediately if desired:

```
{%
  \mpi@endverbatim%
  \mpi@endfig%
  \mpi@writebuf%
  \ifmpi@now%
    \mpi@processnow%
  \fi%
```

Display figure. Remember immediate display of figure unless processed immediately:

```
  \ifmpi@figshow%
    \expandafter\mpi@graphics\mpi@figopt{\mpi@figfile}%
    \ifmpi@now\else\global\mpi@showinfiletrue\fi%
  \fi%
}
```

`\mpostuse` `\mpostuse` includes a metapost figure which was declared earlier via its label. The optional argument is passed as the optional argument for `\includegraphics`. Remember if a file is currently open for writing:

```
\newcommand{\mpostuse}[2][]{%
  \expandafter\ifx\csname mpi@l@#2\endcsname\relax%
    \PackageWarning{mpostinl}{unknown label `#2'}{}%
    \mpostplaceholder[label]{#2}%
  \else%
    \mpi@graphics[#1]{\csname mpi@l@#2\endcsname}%
    \ifmpi@infile\global\mpi@showinfiletrue\fi%
  \fi%
}
```

\mpostgetname `\mpostgetname` gets the filename of a figure declared earlier via its label and returns it in the macro `\mpostfigurename`:

```
\newcommand{\mpostgetname}[1]{%
  \expandafter\ifx\csname mpi@l@#1\endcsname\relax%
    \PackageWarning{mpostinl}{unknown label `#1'}{}%
    \let\mpostfigurename\relax%
  \else%
    \edef\mpostfigurename{\csname mpi@l@#1\endcsname}%
  \fi%
}
```

\mpostfile Declare options for `\mpostfile`:

```
\define@key{mpi@file}{include}[true]{\csname mpi@include#1\endcsname}
```

`\mpostfile` sets up a new metapost file. If the previous file is still open, it will be closed and processed first.

```
\newcommand{\mpostfile}[2][]{%
  \mpi@closefile%
  \mpi@includefalse%
  \setkeys{mpi@file}{#1}%
  \xdef\mpostfilename{#2}%
}
```

\mpostdone `\mpostdone` closes the present metapost file and processes it if applicable.

```
\newcommand{\mpostdone}{\mpi@closefile}
```