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

2025/04/26, v1.5.2

https://ctan.org/pkg/mpostinl
https://github.com/nbeisert/latex-pkg-nb

Abstract

mpostinl is a LaTeX 2_ε 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 present 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

$$\text{\textbackslash usepackage\{mpostinl\}}$$

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):

$$\text{latex -shell-escape } source$$

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:

$$\text{mpost -tex=latex } filename$$

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 section 2.4; finally, some issues regarding other LaTeX packages are discussed in section 2.6.

2.1 Figures and Definitions

mpostfig (*env.*) The main functionality provided by the package is the `mpostfig` environment:

$$\begin{array}{l} \texttt{\textbackslash begin\{mpostfig\}[}\textit{opts}\texttt{]} \\ \quad \textit{metapost code} \\ \texttt{\textbackslash end\{mpostfig\}} \end{array}$$

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

$$\begin{array}{l} \texttt{filenametemplate:="}\textit{filename}\texttt{";} \\ \texttt{beginfig(}\textit{number}\texttt{)} \\ \quad \textit{metapost code} \\ \texttt{endfig;} \end{array}$$

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}` has to find out whether an optional argument is provided by scanning into the block of metapost content. If none is given, all the code already scanned needs to be (re)interpreted as metapost content. In special circumstances this may cause problems. It should then help to use an empty optional argument: `\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 such as `align`, `gather`, `multline`, `equations` and `\<...\>` of amsmath and eqnlines (in certain situations `equation` and `\[...\]` may be problematic as well). 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 Figures which have been previously declared with a label *label* can be recycled any number of times with the command:

4

$$\texttt{\textbackslash mpostuse[}opts\texttt{]\{}label\texttt{\}}$$

The options are passed on to `\includegraphics[`*opt*`]{`*filename*`}`. Note that `\mpostuse` accepts the overlay specification from the `beamer` package

$$\texttt{\textbackslash mpostuse<}overlay\texttt{>[}opts\texttt{]\{}label\texttt{\}}$$

and passes it on to `\includegraphics`.

`\mpostgetname`   Furthermore, the filename of a figure can be obtained by calling `\mpostgetname{`*label*`}`. The filename is returned in the macro `\mpostfigurename`.

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

$$\texttt{\textbackslash begin\{mpostdef\}[}opts\texttt{]}$$
$$code$$
$$\texttt{\textbackslash end\{mpostdef\}}$$

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

- `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.

- `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`. This feature is superseded by the environment `mposttex` described below. The `tex` option is deprecated, and provided for backward compatibility only.

`mposttex` (*env.*)   TEX or LATEX definitions for use in metapost labels can be specified by the `mposttex` environment:

$$\texttt{\textbackslash begin\{mposttex\}[}opts\texttt{]}$$
$$code$$
$$\texttt{\textbackslash end\{mposttex\}}$$

The code will be enclosed in the metapost file by `verbatimtex` and `etex`. Note that the same restrictions as for `mpostfig` (see above) apply to `mposttex`. The optional argument *opts* is a comma-separated list of options:

- `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.

- `dual`[=`true`|`false`] (no value implies `true`, initially set to `false`) – The definitions within the block *code* will be available to both metapost figures and subsequent LATEX code.

2.2   Package Options

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

$$\texttt{\textbackslash usepackage[}opts\texttt{]\{mpostinl\}}$$
$$\text{or} \quad \texttt{\textbackslash PassOptionsToPackage\{}opts\texttt{\}\{mpostinl\}}$$
$$\text{or} \quad \texttt{\textbackslash mpostsetup\{}opts\texttt{\}}$$

`\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.

- `clean`[=`true`|`false`] (no value implies `true`, initially set to `true`) – Enable/disable cleaning of figure files prior to compilation. A dummy figure will be shown in case metapost cannot generate a figure.

- `checksum`[=`true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable update detection. Compiles figures only if the metapost code has changed.

- `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` and `mposttex` 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.

- `beamer[=true|false]` (no value implies `true`, initially set to `true`) – Enable/disable processing of overlay parameters `<...>` by `\mpostuse` defined by the `beamer` package.

- `warnunused[=true|false]` (no value implies `true`, initially set to `true`) – Enable/disable warnings for unused figure labels.

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
`\mpostdisplaylabel`   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:

$$\mpostplaceholder[\textit{type}]\{\textit{name}\}$$

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. It makes sense to display the filename or label through \mpostdisplaylabel which can handle some special characters such as an underscore.

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 META-FONT 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 mposttex enviroment. 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 cleaning, compiling or even the writing of the metapost file(s) by setting the package options compile, clean 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.

The package can check whether the generated files change with respect to the previous LaTeX pass, and only modified files are compiled thereupon. This check can be enabled by setting the option checksum to true. Note that under some conditions (in particular, if the previous LaTeX pass produced compile errors), figures may not get compiled even though the source changed. In that case you can trigger compilation by deleting the generated metapost source or figures or by introducing further changes to the metapost code. By default 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

$$filename\text{-}specifier\,.\,ext$$

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:

$$\text{\texttt{\textbackslash mpostfile[}}\textit{opts}\text{\texttt{]\{}}\textit{filename}\text{\texttt{\}}}$$

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

$$\text{\texttt{\textbackslash begin\{mpostdef\}[global]}}$$

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:

$$\verb|\usepackage[|enc\verb|]{inputenc}|$$

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

```
\begin{mposttex}
\usepackage[enc]{inputenc}
\end{mposttex}
```

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`, `mpostdef` and `mposttex` 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. Note that `\mpostuse` accepts the overlay parameter `<...>`.

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.

Packages amsmath, eqnlines. The packages amsmath and eqnlines declare several math display environments (`align`, `gather`, `multline`, `equations`, `\<...\>`; potentially also `equation` and `\[...\]`) which scan the contents before typesetting them as mathematical expressions. The scanning process eliminates line breaks in the source code and thus breaks the scanning for the end of the metapost code block which has to be on a separate line. Furthermore, other aspects of the metapost code may get distorted, so no attempts are made to recover. Please declare the metapost figures outside of math display environments and print then via `\mpostuse`.

3 Information

3.1 Copyright

Copyright © 2010–2025 Niklas Beisert

This work may be distributed and/or modified under the conditions of the LATEX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in `https://www.latex-project.org/lppl.txt` and version 1.3c or later is part of all distributions of LATEX version 2008 or later.

This work has the LPPL maintenance status 'maintained'.

The Current Maintainer of this work is Niklas Beisert.

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 Required 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 package shellesc to invoke `mpost`. Compatibility with the shellesc package has been tested with v1.0d (2023/07/08).
- 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

v1.5.2:   2025/04/26

- use package `shellesc` to invoke `mpost` for compatibility with LuaTeX

v1.5.1:   2025/03/25

- maintenance and manual update

v1.5:   2018/12/30

- new environment `mposttex` for TeX or LaTeX definitions to replace option `tex` of `mpostdef`; can be used in `dual` mode
- allow underscore, caret and ampersand in figure labels and filenames
- improve handling of optional arguments and special situations when scanning metapost blocks

v1.4:   2018/03/01

- `\mpostuse` now accepts overlay options `<...>` of the `beamer` package; new option `beamer` to control the feature
- issue warnings for unused labels; new option `warnunused` to control the feature

v1.3:   2018/02/09

- new `clean` option to initialise figure files with dummy figures to show METAPOST compiler errors
- `checksum` behaviour improved

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:

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

Adjust the paragraph shape:

```
2 \parindent0pt
3 \parskip6pt
```

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

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

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

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

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

```
6 %% optional: add filename to position labels in metapost code
7 \IfFileExists{currfile.sty}{\usepackage{currfile}}{}
8 %% optional: tools to align graphics
9 \makeatletter\define@key{Gin}{align}[]{}\makeatother
10 \IfFileExists{graphbox.sty}{\usepackage{graphbox}}{}
```

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

```
11 %% some sample package options:
12 %% \mpostsetup{write=false}
13 %% \mpostsetup{compile=false}
14 \mpostsetup{checksum}
15 \mpostsetup{now}
16 %% \mpostsetup{nowall}
17 \mpostsetup{lineno}
18 %% \mpostsetup{latex=false}
19 %% \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:

```
20 %% declare packages to be used for processing labels:
21 \begin{mposttex}[global]
22 \usepackage[utf8]{inputenc}
23 \end{mposttex}
```

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:

```
24 %% specify global definitions:
25 \begin{mpostdef}[global]
26 newinternal unit;
27 unit:=1cm;
28 \end{mpostdef}
```

Begin document body:

```
29 \begin{document}
```

A.2   Basic Functionality

We start by demonstrating the basic functionality of the package:

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

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

```
31 a plain circle:\\
32 \begin{mpostfig}
```

```
33 draw fullcircle scaled unit;
34 label(btex 1 etex, (0,0));
35 \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):

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

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

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

Display the figure:

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

Display the figure with options for `\includegraphics`:

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

Display the figure within a box:

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

Display the figure at the centre of a line:

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

Display the figure within an equation:

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

Show the filename of the figure:

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

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

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

```
69 \\
70 filename: \mpostgetname{name}
71 \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:

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

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

```
74 label does not exist:\\
75 \mpostuse{not_exist}
```

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:

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

Write a totally empty figure

```
82 empty figure:\\
83 \begin{mpostfig}
84 \end{mpostfig}
```

Write a figure with an empty line

```
85 empty figure:\\
86 \begin{mpostfig}
87
88 \end{mpostfig}
```

Make a dual definition

```
89 \begin{mposttex}[dual]
90 \def\mymacro{hello!}
91 \end{mposttex}
```

and test it

```
92 dual definition '\mymacro':\\
93 \begin{mpostfig}
94 draw unitsquare shifted (-0.5,-0.5) xscaled 3 scaled unit;
95 label(btex \mymacro etex, (0,0));
96 \end{mpostfig}
```

A.3 Immediate Processing

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

```
97 \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:

```
98 immediate processing per figure:\\
```

```
 99 \begin{mpostfig}[now]
100 interim unit:=1.5cm;
101 draw fullcircle scaled unit;
102 label(btex 5 etex, (0,0));
103 \end{mpostfig}
```

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

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

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

```
106 generate file \texttt{\jobname-now.mps}:
107 \begin{mpostfig}[file={\jobname-now.mps}]
108 draw fullcircle scaled unit;
109 label(btex 6 etex, (0,0));
110 \end{mpostfig}
111 \\
112 display by \verb+\includegraphics+:\\
113 \includegraphics{\jobname-now.mps}
114 \\
115 source \texttt{\jobname-now.mp}
116 \IfFileExists{\jobname-now.mp}{exists}{does not exist}.
```

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

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

A.4   Filename Composition

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

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

Use the figure label instead of a figure counter:

```
120 by label:\\
121 \mpostsetup{labelnames=true}
122 \begin{mpostfig}[label={circle},show]
123 draw fullcircle scaled unit;
124 label(btex 7 etex, (0,0));
125 \end{mpostfig}
126 \mpostsetup{labelnames=false}
127 \\
128 filename: \mpostgetname{circle}
129 \texttt{\mpostfigurename}
```

Change extension to .eps:

```
130 change extension:
131 \mpostsetup{extension=eps}
132 \begin{mpostfig}[label={ext}]
133 draw fullcircle scaled unit;
134 label(btex 8 etex, (0,0));
135 \end{mpostfig}
136 \mpostsetup{extension=mps}
```

```
137 \\
138 filename: \mpostgetname{ext}
139 \texttt{\mpostfigurename}
```

Change template altogether:

```
140 change template:\\
141 \mpostsetup{template=\mpostfilename-figure-#1.mps}
142 \begin{mpostfig}[label={template},show]
143 draw fullcircle scaled unit;
144 label(btex 9 etex, (0,0));
145 \end{mpostfig}
146 \mpostsetup{template=\mpostfilename-#1.mps}
147 \\
148 filename: \mpostgetname{template}
149 \texttt{\mpostfigurename}
```

Demonstrate numbering within a section (here: subsection)

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

One figure in the first section:

```
151 \subsection{Section 1}
152
153 \begin{mpostfig}[show,label={sec1}]
154 draw fullcircle scaled unit;
155 label(btex 10 etex, (0,0));
156 \end{mpostfig}
157 \\
158 filename: \mpostgetname{sec1}
159 \texttt{\mpostfigurename}
```

Two figures in the second section:

```
160 \subsection{Section 2}
161
162 \begin{mpostfig}[show,label={sec2}]
163 draw fullcircle scaled unit;
164 label(btex 11 etex, (0,0));
165 \end{mpostfig}
166 \\
167 filename: \mpostgetname{sec2}
168 \texttt{\mpostfigurename}
169
170 \begin{mpostfig}[show,label={sec3}]
171 draw fullcircle scaled unit;
172 label(btex 12 etex, (0,0));
173 \end{mpostfig}
174 \\
175 filename: \mpostgetname{sec3}
176 \texttt{\mpostfigurename}
```

One figure in the third section:

```
177 \subsection{Section 3}
178
179 \begin{mpostfig}[show,label={sec4}]
180 draw fullcircle scaled unit;
181 label(btex 13 etex, (0,0));
182 \end{mpostfig}
183 \\
```

```
184 filename: \mpostgetname{sec4}
185 \textttt{\mpostfigurename}
```

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

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

A.5 Multiple Files

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

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

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

```
190 \mpostdone
```

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

```
191 \mpostsetup{globaldef=true}
192
193 \begin{mpostdef}
194 unit:=0.8cm;
195 \end{mpostdef}
196
197 \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:

```
198 \mpostfile[include]{\jobname-inc}
199
200 \begin{mpostdef}
201 def drawcircle =
202 draw fullcircle scaled unit
203 enddef;
204 \end{mpostdef}
205
206 \mpostdone
```

The first file contains one figure:

```
207 \subsection*{Section 1}
208 \mpostfile{\jobname-sec1}
209
210 \begin{mpostfig}[show,label={sec5}]
211 drawcircle;
212 label(btex 14 etex, (0,0));
213 \end{mpostfig}
214 \\
215 filename: \mpostgetname{sec5}
216 \textttt{\mpostfigurename}
217
218 \mpostdone
```

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

```
219 \subsection*{Section 2}
220 \mpostfile{\jobname-sec2}
221
222 \begin{mpostfig}[show,label={sec6}]
223 drawcircle;
224 label(btex 15 etex, (0,0));
225 \end{mpostfig}
226 \\
227 filename: \mpostgetname{sec6}
228 \texttt{\mpostfigurename}
229
230 \begin{mpostdef}
231 unit:=1.5cm;
232 \end{mpostdef}
233
234 \begin{mpostfig}[show,label={sec7}]
235 drawcircle;
236 label(btex 16 etex, (0,0));
237 \end{mpostfig}
238 \\
239 filename: \mpostgetname{sec7}
240 \texttt{\mpostfigurename}
241
242 \mpostdone
```

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

```
243 \subsection*{Section 3}
244 \mpostfile{\jobname-sec3}
245
246 \begin{mpostfig}[show,label={sec8}]
247 drawcircle;
248 label(btex 17 etex, (0,0));
249 \end{mpostfig}
250 \\
251 filename: \mpostgetname{sec8}
252 \texttt{\mpostfigurename}
253
254 \mpostdone
```

End of document body:

```
255 \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.

```
256 \RequirePackage{verbatim}
257 \RequirePackage{graphicx}
258 \RequirePackage{keyval}
259 \RequirePackage{shellesc}
```

22

Package Options.

\mpostfilename   \mpostfilename stores the metapost filename, \mpi@nowname stores the filename for imme-
\mpi@nowname   diate processing, and \mpi@template is the template to generate the figure filenames:
\mpi@extension
\mpi@template
```
260 \def\mpostfilename{\jobname}
261 \def\mpi@nowname{\jobname-tmp}
262 \def\mpi@extension{mps}
263 \def\mpi@template#1{\mpostfilename-#1%
264   \ifx\mpi@extension\mpi@empty\else.\fi\mpi@extension}
```

mpi@count   Declare a counter for figure filenames:

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

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

```
267 \newif\ifmpi@draft\mpi@draftfalse
268 \newif\ifmpi@latex\mpi@latextrue
269 \newif\ifmpi@fonts\mpi@fontsfalse
270 \newif\ifmpi@write\mpi@writetrue
271 \newif\ifmpi@compile\mpi@compiletrue
272 \newif\ifmpi@twice\mpi@twicefalse
273 \newif\ifmpi@clean\mpi@cleantrue
274 \newif\ifmpi@checksum\mpi@checksumfalse
275 \newif\ifmpi@lineno\mpi@linenofalse
276 \newif\ifmpi@labelnames\mpi@labelnamesfalse
277 \newif\ifmpi@nowactive\mpi@nowactivefalse
278 \newif\ifmpi@now\mpi@nowfalse
279 \newif\ifmpi@nowkeep\mpi@nowkeepfalse
280 \newif\ifmpi@include\mpi@includefalse
281 \newif\ifmpi@defglobal\mpi@defglobalfalse
282 \newif\ifmpi@texdual\mpi@texdualfalse
283 \newif\ifmpi@warnunused\mpi@warnunusedtrue
284 \newif\ifmpi@beamer\mpi@beamertrue
```

\mpi@mpostmem   These definitions store the options for processing labels via TeX or LaTeX:
\mpi@mpostcompiler
\mpi@latexclass
\mpi@latexoptions
\mpi@documentclass
```
285 \def\mpi@mpostmem{}
286 \def\mpi@mpostcompiler{}
287 \def\mpi@latexclass{article}
288 \def\mpi@latexoptions{}
289 \def\mpi@documentclass{\@backslashchar documentclass%
290   \mpi@latexoptions{\mpi@latexclass}}
```

\mpi@warncompile   Warn and disable compiling if shell escape is unavailable:

```
291 \def\mpi@warncompile{\ifmpi@compile\ifnum\ShellEscapeStatus=0\relax%
292   \PackageWarningNoLine{mpostinl}{shell escape (write18) disabled, %
293     manual metapost compiling required}{}%
294   \global\mpi@compilefalse\fi\fi}
```

Process package options:

```
295 \def\mpi@group{mpi@}
296 \DeclareOption{final}{\mpi@draftfalse}
297 \define@key{\mpi@group}{draft}[true]{\csname mpi@draft#1\endcsname}
298 \define@key{\mpi@group}{write}[true]{\csname mpi@write#1\endcsname}
299 \define@key{\mpi@group}{latex}[true]{\csname mpi@latex#1\endcsname}
```

```
300 \define@key{\mpi@group}{compile}[true]{\csname mpi@compile#1\endcsname}
301 \define@key{\mpi@group}{twice}[true]{\csname mpi@twice#1\endcsname}
302 \define@key{\mpi@group}{clean}[true]{\csname mpi@clean#1\endcsname}
303 \define@key{\mpi@group}{checksum}[true]{\csname mpi@checksum#1\endcsname}
304 \define@key{\mpi@group}{fonts}[true]{\csname mpi@fonts#1\endcsname}
305 \define@key{\mpi@group}{prologues}[]{\def\mpi@prologues{#1}}
306 \define@key{\mpi@group}{lineno}[true]{\csname mpi@lineno#1\endcsname}
307 \define@key{\mpi@group}{labelnames}[true]{\csname mpi@labelnames#1\endcsname}
308 \define@key{\mpi@group}{compiler}[]{\def\mpi@texcompiler{#1}}
309 \define@key{\mpi@group}{format}[]{\def\mpi@texformat{#1}}
310 \define@key{\mpi@group}{mem}[]{\def\mpi@mpostmem{#1}}
311 \define@key{\mpi@group}{command}[]{\def\mpi@mpostcompiler{#1}}
312 \define@key{\mpi@group}{class}{\def\mpi@latexclass{#1}}
313 \define@key{\mpi@group}{classopt}[]{\def\mpi@latexoptions{[#1]}}
314 \define@key{\mpi@group}{beamer}[true]{\csname mpi@beamer#1\endcsname}
315 \define@key{\mpi@group}{warnunused}[true]{\csname mpi@warnunused#1\endcsname}
316 \define@key{\mpi@group}{now}[true]{\csname mpi@nowactive#1\endcsname}
317 \define@key{\mpi@group}{nowall}[true]{\csname mpi@now#1\endcsname}
318 \define@key{\mpi@group}{nowkeep}[true]{\csname mpi@nowkeep#1\endcsname}
319 \define@key{\mpi@group}{globaldef}[true]{\csname mpi@defglobal#1\endcsname}
320 \define@key{\mpi@group}{extension}[]{\def\mpi@extension{#1}}
321 \define@key{\mpi@group}{template}{\def\mpi@template##1{#1}}
322 \define@key{\mpi@group}{numberwithin}{%
323   \@addtoreset{mpi@count}{#1}%
324   \def\thempi@count{\arabic{#1}-\arabic{mpi@count}}%
325 }
```

Pass undeclared options on to keyval processing:

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

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

```
328 \ProcessOptions
329 \mpi@warncompile
330 \ifdefined\pdfmdfivesum\else\mpi@checksumfalse\fi
```

Internal Commands and Definitions.

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

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

\mpi@unusedlist Initialise a list of unused figure labels:

```
332 \def\mpi@unusedlist{}
```

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

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

\ifmpi@infile \ifmpi@infile indicates whether a file is open, \ifmpi@inbody indicates whether the con-
\ifmpi@inbody tent section has started:

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

24

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

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

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

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

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

\mpi@writebuf   Write to the file:

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

\mpi@writenow   Write to the immediate buffer:

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

\mpi@buf   Declare three token buffers to store the current block (\mpi@buf), global definitions
\mpi@defbuf   (\mpi@defbuf) and the definitions for immediate processing (\mpi@nowbuf):
\mpi@nowbuf
```
344 \newtoks\mpi@buf
345 \newtoks\mpi@defbuf
346 \newtoks\mpi@nowbuf
347 \newtoks\mpi@verbbuf
348 \mpi@defbuf={}
```

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

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

\mpi@clearbuf   \mpi@clearbuf clears the current block buffer. \mpi@addbufexp expands the first token
\mpi@addbufexp   and adds to the current block buffer. \mpi@addbufbuf adds a buffer to the current block
\mpi@addbufbuf   buffer. \mpi@addbuf adds to the current block buffer via \protected@edef:
\mpi@addbuf
```
351 \def\mpi@clearbuf{\global\mpi@buf={}}
352 \def\mpi@addbufexp#1{\mpi@addtoexp\mpi@buf{#1^^J}}
353 \def\mpi@addbufbuf#1{\mpi@addtoexp\mpi@buf{\the#1}}
354 \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:

```
355 \def\mpi@stripext#1{\edef\mpi@tmp{#1}\expandafter%
356   \mpi@stripstart\expandafter{\mpi@tmp}}
357 \def\mpi@ifeq#1#2#3#4{\def\mpi@tmpa{#1}\def\mpi@tmpb{#2}%
358   \ifx\mpi@tmpa\mpi@tmpb#3\else#4\fi}
359 \def\mpi@stripstart#1{\mpi@stripfor{\@gobble}#1.\@@.}
360 \def\mpi@stripfor#1#2.#3.{%
361   \begingroup%
362   \mpi@ifeq{#3}{\@@}{%
```

```
363        \def\mpi@tmp{\def\mpi@stripped{#1.#2}}%
364        \mpi@ifeq{#1}{\@gobble}{}{%
365          \mpi@ifeq{#2}{eps}{\def\mpi@tmp{\def\mpi@stripped{#1}}}{}%
366          \mpi@ifeq{#2}{mps}{\def\mpi@tmp{\def\mpi@stripped{#1}}}{}%
367          \ifx\mpi@extension\mpi@empty\else%
368            \expandafter\mpi@ifeq\expandafter{\mpi@extension}{#2}%
369              {\def\mpi@tmp{\def\mpi@stripped{#1}}}{}%
370          \fi%
371        }%
372      }{\def\mpi@tmp{\mpi@stripfor{#1.#2}#3.}}%
373      \expandafter\endgroup\mpi@tmp%
374  }
```

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

```
375  \newcommand{\mpi@warnmod}{%
376    \ifmpi@checksum\ifmpi@warnmod%
377      \PackageWarningNoLine{mpostinl}{figure(s) may have changed. %
378        Rerun to update figures}{}%
379    \fi\fi%
380  }
```

\mpi@warnunused This macro processes the list of all figure labels and issues a warning for every unused ones:

```
381  \def\mpi@warnunused#1{%
382    \begingroup\def\mpi@tmp{#1}\ifx\mpi@tmp\mpi@empty\else%
383      \ifcsname mpi@lu@#1\endcsname\else
384        \PackageWarningNoLine{mpostinl}{figure label '#1' not used}{}%
385      \fi%
386      \def\mpi@tmp{\mpi@warnunused}%
387    \fi\expandafter\endgroup\mpi@tmp%
388  }
```

Run \mpi@warnunused at the end of the document:

```
389  \AtEndDocument{\expandafter\mpi@warnunused\mpi@unusedlist{}}
```

\mpostdisplaylabel Rescan a string potentially containing the special characters underscore ('_'), caret ('^') and ampersand ('&'):

```
390  \newcommand{\mpostdisplaylabel}[1]{{%
391    \catcode'\_=11\catcode'\^=11\catcode'\&=11%
392    \edef\mpi@tmp{#1}\expandafter\scantokens\expandafter{\mpi@tmp}}}
```

\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'):

```
393  \newcommand{\mpostplaceholder}[2][]{\parbox[c]{1in}{%
394    \hrule\vrule\hfill%
395    \parbox[c]{0pt}{\rule{0cm}{0.6in}}%
396      \makebox[0pt][c]{\scriptsize\tt\mpostdisplaylabel{#2}}%
397    \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:

```
398  \newcommand{\mpi@graphics}[2][]{%
399    \IfFileExists{#2}%
```

26

```
400        {\edef\mpi@tmp{#2}\includegraphics[#1]{\mpi@tmp}}%
401        {\typeout{graphics file '#2' missing}\mpostplaceholder[file]{#2}}%
402 }
```

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

```
403 \newcommand{\mpi@verbatim}{%
404    \begingroup%
405    \@bsphack%
406    \let\do\@makeother\dospecials%
407    \catcode`\^^M\active%
408    \global\mpi@verbbuf={}%
409    \def\verbatim@processline{\mpi@addtoexp\mpi@verbbuf{\the\verbatim@line^^J}}%
410    \verbatim@start%
411 }
```

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

```
412 \newcommand{\mpi@endverbatim}{%
413    \@esphack%
414    \endgroup%
415 }
```

\mpi@scanblock Read one optional argument and pass on to an environment which reads a block from the source file. Handle special cases of an initial empty line or a completely empty block. Pass prefetched tokens as second argument:

```
416 \def\mpi@scanblock#1{%
417    \@ifnextchar\par{\mpi@scanblock@par{#1}}{\mpi@scanblock@sel{#1}}}
418 \long\def\mpi@scanblock@par#1\par{\mpi@scanblock@sel{#1}[]}
419 \def\mpi@scanblock@sel#1{\@ifnextchar[{\mpi@scanblock@opt{#1}}%
420    {\@ifnextchar\end{\mpi@scanblock@end{#1}}{\mpi@scanblock@noopt{#1}}}}
421 \def\mpi@scanblock@end#1\end#2{%
422    \def\mpi@tmp{#2}\ifx\mpi@tmp\@currenvir%
423       \def\mpi@verbatim{\global\mpi@verbbuf={}}\def\mpi@endverbatim{}%
424    \fi%
425    #1{}{\scantokens{\end{#2}}}}
426 \def\mpi@scanblock@noopt#1#2{#1{}{\scantokens#2}}
427 \def\mpi@scanblock@opt#1[#2]{#1{#2}{}}
```

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

```
428 \newcommand{\mpi@putlineno}{%
429    \ifmpi@lineno%
430       \mpi@addbuf{\@percentchar------------------------------------}%
431       \mpi@addbuf{\@percentchar%
432          \ifx\currfilename\@undefined\else\currfilename\space\fi%
433          l.\the\inputlineno}%
434    \fi%
435 }
```

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

```
436 \newcommand{\mpi@beginfig}[1]{%
437    \mpi@addbuf{filenametemplate \mpi@dblquotchar#1\mpi@dblquotchar;}%
438    \mpi@addbuf{beginfig(\arabic{mpi@count})}%
439 }
```

27

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

```
440 \newcommand{\mpi@endfig}{%
441   \mpi@addbuf{endfig;}%
442 }
```

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

```
443 \newcommand{\mpi@declaredoc}{%
444   \ifmpi@latex%
445     \mpi@addbuf{verbatimtex}%
446     \mpi@addbuf{\mpi@documentclass}%
447     \mpi@addbuf{etex}%
448     \mpi@addbuf{}%
449   \fi%
450 }
```

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

```
451 \newcommand{\mpi@begindoc}{%
452   \ifmpi@latex%
453     \mpi@putlineno%
454     \mpi@addbuf{verbatimtex}%
455     \mpi@addbuf{\@backslashchar begin{document}}%
456     \mpi@addbuf{etex}%
457   \fi%
458 }
```

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

```
459 \newcommand{\mpi@enddoc}{%
460   \ifmpi@latex%
461     \mpi@putlineno%
462     \mpi@addbuf{verbatimtex}%
463     \mpi@addbuf{\@backslashchar end{document}}%
464     \mpi@addbuf{etex}%
465   \fi%
466 }
```

\mpi@declareformat Write TeX format specifier to buffer:

```
467 \newcommand{\mpi@declareformat}{%
468   \let\mpi@tmp\mpi@texformat%
469   \ifx\mpi@tmp\@undefined\def\mpi@tmp{\ifmpi@latex latex\else tex\fi}\fi%
470   \ifx\mpi@tmp\mpi@empty\else%
471     \mpi@addbuf{verbatimtex}%
472     \mpi@addbuf{\@percentchar &\mpi@tmp}%
473     \mpi@addbuf{etex}%
474     \mpi@addbuf{}%
475   \fi%
476 }
```

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

```
477 \newcommand{\mpi@composehead}{%
478   \mpi@putlineno%
479   \let\mpi@tmp\mpi@prologues%
```

```
480    \ifx\mpi@tmp\@undefined\def\mpi@tmp{\ifmpi@fonts 3\else 2\fi}\fi%
481    \ifx\mpi@tmp\mpi@empty\else%
482      \mpi@addbuf{prologues:=\mpi@tmp;}%
483    \fi%
484    \ifmpi@draft\mpi@addbuf{draft:=1;}\fi%
485    \mpi@addbuf{}%
486    \mpi@declareformat%
487    \mpi@declaredoc%
488 }
```

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

```
489 \newcommand{\mpi@beginfile}{%
490    \ifx\mpi@mpostmem\mpi@empty\else%
491      \mpi@addbuf{\@percentchar &\mpi@mpostmem}%
492    \fi%
493    \mpi@addbuf{\@percentchar generated from file '\jobname' by mpostinl.sty}%
494    \ifmpi@include\else%
495      \mpi@composehead%
496      \mpi@addbufbuf{\mpi@defbuf}%
497    \fi%
498 }
```

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

```
499 \newcommand{\mpi@endfile}{%
500    \mpi@putlineno%
501    \ifmpi@include\else%
502      \mpi@addbuf{end}%
503    \fi%
504 }
```

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

```
505 \newcommand{\mpi@getchecksum}[3]{%
506    \IfFileExists{#3}{#1\edef#2{\pdfmdfivesum file{#3}}}{#1\let#2=\@undefined}%
507 }
```

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

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

Prevent reopening and overwriting the previous file:

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

Compute checksum of old file:

```
513      \ifmpi@checksum\ifmpi@inclmod\else%
514        \mpi@getchecksum{\global}{\mpi@checksum@before}{\mpostfilename.mp}%
515      \fi\fi%
516      \ifmpi@checksum\global\mpi@filemodfalse\else\global\mpi@filemodtrue\fi%
517      \ifmpi@inclmod\global\mpi@filemodtrue\fi%
```

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

```
518      \global\mpi@infiletrue%
519      \global\mpi@showinfilefalse%
```

```
520     \gdef\mpi@filelist{}%
521     \ifmpi@write\immediate\openout\mpi@out\mpostfilename.mp\fi%
522     \mpi@clearbuf%
523     \mpi@beginfile%
524     \mpi@writebuf%
```

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

```
525     \ifmpi@include%
526       \mpi@clearbuf%
527       \mpi@putlineno%
528       \mpi@addbuf{input \mpostfilename}%
529       \mpi@addtoexp\mpi@defbuf{\the\mpi@buf^^J}%
```

If immediate mode is available fill immediate buffer with header:

```
530     \else%
531       \global\mpi@nowbuf={}%
532       \mpi@writenow%
533     \fi%
534   \fi%
535 }
```

\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:

```
536 \newcommand{\mpi@startcontent}{%
537   \mpi@startfile%
538   \ifmpi@inbody\else%
539     \global\mpi@inbodytrue%
540     \mpi@clearbuf%
541     \mpi@begindoc%
542     \mpi@writebuf%
543     \mpi@writenow%
544   \fi%
545 }
```

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

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

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

```
549       \def\mpi@imode{}%
550       \ifcase\the\interactionmode%
551         \def\mpi@imode{-interaction=batchmode}\or%
552         \def\mpi@imode{-interaction=nonstopmode}\or%
553         \def\mpi@imode{-interaction=scrollmode}\or%
554         \def\mpi@imode{-interaction=errorstopmode}\fi%
555       \let\mpi@texswitch\mpi@texcompiler%
556       \ifx\mpi@texswitch\@undefined%
557         \def\mpi@texswitch{\ifmpi@latex latex\else tex\fi}%
558       \fi%
559       \def\mpi@execute{mpost\space%
560         \mpi@imode\space%
561         \ifx\mpi@mpostmem\mpi@empty\else -mem=\mpi@mpostmem\space\fi%
```

```
562          \ifx\mpi@texswitch\mpi@empty\else -tex=\mpi@texswitch\space\fi%
563          #1}%
564      \else%
```

Compose custom command:

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

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

```
567      \ShellEscape{\mpi@execute}%
568      \ifmpi@twice%
569        \ShellEscape{\mpi@execute}%
570      \fi%
571    \fi\fi%
572 }
```

\mpi@cleanfile Write a mps file containing a plain cross:

```
573 \newcommand{\mpi@cleanfile}[1]{%
574    \immediate\openout\mpi@outnow#1%
575    \immediate\write\mpi@outnow{%
576      \@percentchar!PS-Adobe-3.0 EPSF-3.0^^J%
577      \@percentchar\@percentchar Title: missing figure file^^J%
578      \@percentchar\@percentchar Creator: mpostinl.sty^^J%
579      \@percentchar\@percentchar BoundingBox: -36 -22 36 22^^J%
580      \@percentchar\@percentchar EndComments^^J%
581      \@percentchar\@percentchar Page: 1 1^^J%
582      0 setgray 0 1.5 dtransform truncate idtransform setlinewidth pop^^J%
583      [] 0 setdash 1 setlinecap 1 setlinejoin 10 setmiterlimit^^J%
584      newpath -35 21 moveto 35 -21 lineto stroke^^J%
585      newpath -35 -21 moveto 35 21 lineto stroke^^J%
586      \@percentchar\@percentchar EOF}%
587    \immediate\closeout\mpi@outnow%
588 }
```

\mpi@cleanfiles Clean all figure files prior to compilation:

```
589 \newcommand{\mpi@cleanfiles}[1]{%
590    \begingroup\def\mpi@tmp{#1}%
591    \ifx\mpi@tmp\mpi@empty\else\def\mpi@tmp{\mpi@cleanfiles}%
592      \mpi@cleanfile{#1}\fi%
593    \expandafter\endgroup\mpi@tmp%
594 }
```

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

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

Write end of content section (if started):

```
597      \mpi@clearbuf%
598      \ifmpi@inbody%
599        \mpi@enddoc%
600        \mpi@addbuf{}%
601      \fi%
```

Write end of file and close:

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

Determine whether file has changed:

```
605     \ifmpi@filemod\else%
606       \mpi@getchecksum{}{\mpi@checksum@after}{\mpostfilename.mp}%
607       \ifx\mpi@checksum@before\mpi@checksum@after\else\mpi@filemodtrue\fi%
608     \fi%
609     \ifmpi@include\ifmpi@filemod\global\mpi@inclmodtrue\fi\fi%
```

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

```
610     \ifmpi@filemod\ifmpi@inbody%
611       \ifmpi@clean\ifmpi@write\ifmpi@compile%
612         \expandafter\mpi@cleanfiles\mpi@filelist{}\fi\fi\fi%
613       \mpi@compile{\mpostfilename.mp}%
614     \fi\fi%
615     \ifmpi@filemod\ifmpi@showinfile\global\mpi@warnmodtrue\fi\fi%
```

Reset variables:

```
616     \global\mpi@infilefalse%
617     \global\let\mpostfilename\mpi@empty%
618     \global\mpi@inbodyfalse%
619     \setcounter{mpi@count}{0}%
620   \fi%
621 }
```

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

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

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

```
624     \ifmpi@nowkeep%
625       \mpi@stripext{\mpi@figfile}%
626       \edef\mpi@nowname{\mpi@stripped}%
627     \fi%
```

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

```
628     \immediate\openout\mpi@outnow\mpi@nowname.mp%
629     \immediate\write\mpi@outnow{\the\mpi@nowbuf}%
630     \immediate\write\mpi@outnow{\the\mpi@buf}%
631     \mpi@clearbuf%
632     \mpi@enddoc%
633     \mpi@addbuf{}%
634     \mpi@endfile%
635     \immediate\write\mpi@outnow{\the\mpi@buf}%
636     \immediate\closeout\mpi@outnow%
637     \ifmpi@clean\mpi@cleanfile{\mpi@figfile}\fi%
638     \mpi@compile{\mpi@nowname.mp}%
639   \fi\fi\fi%
640 }
```

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

```
641 \AtEndDocument{\mpi@closefile}
642 \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:

```
643 \newcommand{\mpostsetup}[1]{%
644   \setkeys\mpi@group{#1}%
645   \mpi@warncompile%
646   \ifdefined\pdfmdfivesum\else\mpi@checksumfalse\fi%
647 }
```

mpostdef Declare options for the mpostdef environment:

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

The environment mpostdef adds a block of definitions to the metapost file. Pass on to \mpi@scanblock mechanism:

```
651 \def\mpostdef{\mpi@scanblock{\mpi@mpostdef}}
652 \def\mpi@mpostdef#1#2{%
```

Process optional arguments:

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

TEX definitions cannot be in an include file:

```
655   \ifmpi@defglobal\else\ifmpi@deftex\ifmpi@include%
656     \PackageWarning{mpostinl}{tex definitions within an include file %
657       will be ignored by mpost; switching to global definition}{}%
658     \mpi@defglobaltrue%
659   \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:

```
660   \ifmpi@defglobal\else%
661     \mpi@startfile%
662   \fi%
663   \mpi@clearbuf%
664   \mpi@putlineno%
665   \ifmpi@deftex%
666     \mpi@addbuf{verbatimtex}%
667   \fi%
668   \mpi@verbatim#2%
669 }
```

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

```
670 \def\endmpostdef
671 {%
672   \mpi@endverbatim%
673   \mpi@addbufbuf{\mpi@verbbuf}%
674   \ifmpi@deftex%
675     \mpi@addbuf{etex}%
676   \fi%
```

33

```
677   \ifmpi@defglobal%
678     \mpi@addtoexp\mpi@defbuf{\the\mpi@buf^^J}%
679     \ifmpi@include\else\ifmpi@infile%
680       \mpi@writebuf%
681       \mpi@writenow%
682     \fi\fi%
683   \else%
684     \mpi@writebuf%
685     \ifmpi@include\else\mpi@writenow\fi%
686   \fi%
687 }
```

mposttex Declare options for the `mposttex` environment:

```
688 \newif\ifmpi@deftex
689 \define@key{mpi@tex}{dual}[true]{\csname mpi@texdual#1\endcsname}
690 \define@key{mpi@tex}{global}[true]{\csname mpi@defglobal#1\endcsname}
```

The environment `mposttex` adds a block of tex definitions to the metapost file. Pass on to `\mpi@scanblock` mechanism:

```
691 \def\mposttex{\mpi@scanblock{\mpi@mposttex}}
692 \def\mpi@mposttex#1#2{%
```

Process optional arguments:

```
693   \mpi@texdualfalse%
694   \setkeys{mpi@tex}{#1}%
```

TEX definitions cannot be in an include file:

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

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

```
700   \ifmpi@defglobal\else%
701     \mpi@startfile%
702   \fi%
703   \mpi@clearbuf%
704   \mpi@putlineno%
705   \mpi@addbuf{verbatimtex}%
706   \mpi@verbatim#2%
707 }
```

Postprocessing; add 'etex', add to appropriate buffer(s).

```
708 \def\endmposttex
709 {%
710   \mpi@endverbatim%
711   \mpi@addbufbuf{\mpi@verbbuf}%
712   \mpi@addbuf{etex}%
713   \ifmpi@defglobal%
714     \mpi@addtoexp\mpi@defbuf{\the\mpi@buf^^J}%
715     \ifmpi@include\else\ifmpi@infile%
716       \mpi@writebuf%
717       \mpi@writenow%
718     \fi\fi%
```

```
719  \else%
720    \mpi@writebuf%
721    \ifmpi@include\else\mpi@writenow\fi%
722  \fi%
```

Hack \ignorespacesafterend mechanism to ship out block contents after environment group:

```
723  \ifmpi@texdual%
724    \ignorespacesafterend%
725    \global\let\mpi@ignorespaces\ignorespaces%
726    \gdef\ignorespaces{\expandafter\scantokens\expandafter{\the\mpi@verbbuf}%
727      \global\let\ignorespaces\mpi@ignorespaces}%
728  \fi%
729 }
```

mpostfig Declare options for the mpostfig environment:

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

The environment mpostfig adds a figure to the metapost file. Pass on to \mpi@scanblock mechanism:

```
737 \def\mpostfig{\mpi@scanblock{\mpi@mpostfig}}
738 \def\mpi@mpostfig#1#2{%
```

Make sure that include files do not contain figures:

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

Process optional arguments:

```
742  \def\mpi@figfile{}%
743  \def\mpi@figlabel{}%
744  \def\mpi@figopt{}%
745  \mpi@figshowfalse%
746  \setkeys{mpi@fig}{#1}%
```

Display figure if no filename or label is provided:

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

Compose filename from label (if desired and specified):

```
748  \ifmpi@labelnames\ifx\mpi@figfile\mpi@empty\ifx\mpi@figlabel\mpi@empty\else%
749    \edef\mpi@figfile{\mpi@template{\mpi@figlabel}}%
750  \fi\fi\fi%
```

Compose filename from counter if no filename is provided:

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

Save filename to label, warn if label has already been defined, remember all labels for checking whether unused:

```
755  \ifx\mpi@figlabel\mpi@empty\else%
756    \expandafter\ifx\csname mpi@l@\mpi@figlabel\endcsname\relax\else%
757      \PackageWarning{mpostinl}{label '\mpi@figlabel' already defined; %
758        overwriting}{}%
759    \fi%
760    \expandafter\xdef\csname mpi@l@\mpi@figlabel\endcsname{\mpi@figfile}%
761    \ifmpi@warnunused\ifmpi@figshow\else%
762      \xdef\mpi@unusedlist{\mpi@unusedlist{\mpi@figlabel}}%
763    \fi\fi%
764  \fi%
```

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

```
765  \mpi@startcontent%
766  \mpi@clearbuf%
767  \mpi@putlineno%
768  \mpi@beginfig{\mpi@figfile}%
769  \mpi@verbatim#2%
770  }
```

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

```
771  \def\endmpostfig
772  {%
773    \mpi@endverbatim%
774    \mpi@addbufbuf{\mpi@verbbuf}%
775    \mpi@endfig%
776    \mpi@writebuf%
777    \xdef\mpi@filelist{\mpi@filelist{\mpi@figfile}}%
778    \ifmpi@now%
779      \mpi@processnow%
780    \fi%
```

Force compile if output file does not exist, is empty or equals dummy figure:

```
781    \ifmpi@filemod\else%
782      \mpi@getchecksum{}{\mpi@tmpa}{\mpi@figfile}%
783      \ifdefined\mpi@tmpa%
784        \edef\mpi@tmpa{\scantokens\expandafter{\mpi@tmpa\noexpand}}%
785        \def\mpi@tmpb{965DFFE32ABB43CC14C58E254B4D660E}%
786        \ifx\mpi@tmpa\mpi@tmpb\global\mpi@filemodtrue\fi%
787        \def\mpi@tmpb{D78A5F2EE4B244260ECD6A0E89ADC9CE}%
788        \ifx\mpi@tmpa\mpi@tmpb\global\mpi@filemodtrue\fi%
789        \def\mpi@tmpb{D41D8CD98F00B204E9800998ECF8427E}%
790        \ifx\mpi@tmpa\mpi@tmpb\global\mpi@filemodtrue\fi%
791      \else%
792        \global\mpi@filemodtrue%
793      \fi
794    \fi%
```

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

```
795    \ifmpi@figshow%
796      \expandafter\mpi@graphics\mpi@figopt{\mpi@figfile}%
797      \ifmpi@now\else\global\mpi@showinfiletrue\fi%
798    \fi%
799  }
```

\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 that the label has been used, remember if a file is currently open for writing:

```
800 \newcommand{\mpostuse}[2][]{%
801   \expandafter\ifx\csname mpi@l@#2\endcsname\relax%
802     \PackageWarning{mpostinl}{unknown label '#2'}{}%
803     \mpostplaceholder[label]{#2}%
804   \else%
805     \ifmpi@warnunused\expandafter\gdef\csname mpi@lu@#2\endcsname{}\fi%
806     \mpi@graphics[#1]{\csname mpi@l@#2\endcsname}%
807     \ifmpi@infile\global\mpi@showinfiletrue\fi%
808   \fi%
809 }
```

\mpostgetname \mpostgetname gets the filename of a figure declared earlier via its label and returns it in the macro \mpostfigurename. Remember that the label has been used:

```
810 \newcommand{\mpostgetname}[1]{%
811   \expandafter\ifx\csname mpi@l@#1\endcsname\relax%
812     \PackageWarning{mpostinl}{unknown label '#1'}{}%
813     \let\mpostfigurename\relax%
814   \else%
815     \ifmpi@warnunused\expandafter\gdef\csname mpi@lu@#1\endcsname{}\fi%
816     \edef\mpostfigurename{\csname mpi@l@#1\endcsname}%
817   \fi%
818 }
```

\mpostfile Declare options for \mpostfile:

```
819 \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:

```
820 \newcommand{\mpostfile}[2][]{%
821   \mpi@closefile%
822   \mpi@includefalse%
823   \setkeys{mpi@file}{#1}%
824   \xdef\mpostfilename{#2}%
825 }
```

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

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

\mpostuse<> This overwrites the definition of \mpostfile to accept the overlay option <...> of the beamer package:

```
827 \ifmpi@beamer\@ifclassloaded{beamer}{
828   \let\mpi@plain@mpostuse\mpostuse%
829   \newcommand<>{\mpostuse}[2][]{\begingroup%
830     \expandafter\def\expandafter\includegraphics\expandafter%
831       {\includegraphics#3}%
832     \mpi@plain@mpostuse[#1]{#2}\endgroup}%
833 }{}\fi
```