

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

4 January 2017, v1.0

Abstract

mpostinl is a $\text{\LaTeX} 2_{\epsilon}$ 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	2
1.1	Related CTAN Packages and Software	3
2	Usage	3
2.1	Figures and Definitions	4
2.2	Package Options	5
2.3	Writing and Compiling Options	7
2.4	Multiple Files	8
2.5	Immediate Processing	9
2.6	Interaction with Other Packages	10
2.7	Feature Suggestions	11
A	Files and Installation	11
B	Sample File	11
B.1	Preamble	12
B.2	Basic Functionality	12
B.3	Immediate Processing	14
B.4	Filename Composition	15
B.5	Multiple Files	16
C	Implementation	18
D	Copyright	28

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 and Software

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 2.7.

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.

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 L^AT_EX 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 L^AT_EX source, writes them to a metapost file, and compiles them at the end of the L^AT_EX document. This means that the figures (or their updates) will normally *not* be available in the first L^AT_EX 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 L^AT_EX 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 MiK_TE_X distribution the appropriate command line option is `-enable-write18`. In a L^AT_EX font 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 L^AT_EX 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 L^AT_EX 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 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` Figures which have been previously declared with a label *label* can be recycled any number of times with the command:

`\mpostgetname`

`\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.
- `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.
- `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. This will also set the `compiler` option appropriately.
- `compiler=compiler` – Set the compiler program for labels to *compiler*. The initial setting is `tex` (in \TeX mode) or `latex` (in \LaTeX mode).
- `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` (initially set to `mpost`) – Set the metapost command to *command*.
- `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.

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 meta-post 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` environment 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 L^AT_EX source is uploaded to a repository or passed on to a publisher.

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 L^AT_EX 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 L^AT_EX source. If the L^AT_EX 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 T_EX 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 `mpostnl` writes out a single file *source.mp* if the L^AT_EX 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 L^AT_EX 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 L^AT_EX 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 L^AT_EX 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 L^AT_EX 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}`.

2.7 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.
- Compare the generated metapost file to the previous instance and compile only in case of changes. This could be implemented by a checksum or by some external tool such as `diff`. This feature is probably too difficult to implement.

A Files and Installation

The package consists of the files:

<code>README</code>	readme file
<code>mpostinl.ins</code>	installation file
<code>mpostinl.dtx</code>	source file
<code>mpostinl.sty</code>	package file
<code>mpinlsmp.tex</code>	sample file
<code>mpostinl.pdf</code>	manual

The distribution consists of the files `README`, `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`.

B Sample File

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

B.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 an alignment switch `\align` (if `graphbox` is available):

```
6 \IfFileExists{currfile.sty}{\usepackage{currfile}}{}
7 \IfFileExists{graphbox.sty}
8   {\def\align{align}\usepackage{graphbox}}
9   {\def\align{}}
```

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

```
10 %% \mpostsetup{write=false}
11 %% \mpostsetup{compile=false}
12 \mpostsetup{now}
13 %% \mpostsetup{nowall}
14 \mpostsetup{lineno}
15 %% \mpostsetup{latex=false}
16 %% \mpostsetup{classopt={12pt}}
```

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

```
17 \begin{mpostdef}[tex,global]
18 \usepackage[utf8]{inputenc}
19 \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 is included in all of them:

```
20 \begin{mpostdef}[global]
21 newinternal unit;
22 unit:=1cm;
23 \end{mpostdef}
```

Begin document body:

```
24 \begin{document}
```

B.2 Basic Functionality

We start by demonstrating the basic functionality of the package:

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

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

```
26 a plain circle:\\
27 \begin{mpostfig}
28 draw fullcircle scaled unit;
29 label(btex 1 etex, (0,0));
30 \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):

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

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

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

Display the figure:

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

Display the figure with options for `\includegraphics`:

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

Display the figure within a box:

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

Display the figure at the centre of a line:

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

Display the figure within an equation:

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

Show the filename of the figure:

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

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

```
59 figure with filename:\\
```

```

60 \begin{mpostfig}[file={\jobname-name.mps},label={name},show]
61 draw fullcircle scaled unit;
62 label(btex 4 etex, (0,0));
63 \end{mpostfig}
64 \\\
65 filename: \mpostgetname{name}
66 \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:

```

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

```

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

```

69 label does not exist:\
70 \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:

```

71 utf-8 test:\
72 \begin{mpostfig}
73 interim unit:=1.5cm;
74 draw fullcircle scaled unit;
75 label(btex ââââââ etex, (0,0));
76 \end{mpostfig}

```

B.3 Immediate Processing

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

```

77 \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:

```

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

```

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

```

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

```

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

```

86 generate file \texttt{\jobname-now.mps}:
87 \begin{mpostfig}[file={\jobname-now.mps}]
88 draw fullcircle scaled unit;
89 label(btex 6 etex, (0,0));
90 \end{mpostfig}
91 \\\
92 display by \verb+\includegraphics+:\

```

```

93 \includegraphics{\jobname-now.mps}
94 \\\
95 source \texttt{\jobname-now.mp}
96 \IfFileExists{\jobname-now.mp}{exists}{does not exist}.

```

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

```

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

```

B.4 Filename Composition

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

```

99 \section{Filename Composition}

```

Use the figure label instead of a figure counter:

```

100 by label:\\
101 \mpostsetup{labelnames=true}
102 \begin{mpostfig}[label={circle},show]
103 draw fullcircle scaled unit;
104 label(btex 7 etex, (0,0));
105 \end{mpostfig}
106 \mpostsetup{labelnames=false}
107 \\\
108 filename: \mpostgetname{circle}
109 \texttt{\mpostfigurename}

```

Change extension to .eps:

```

110 change extension:
111 \mpostsetup{extension=eps}
112 \begin{mpostfig}[label={ext}]
113 draw fullcircle scaled unit;
114 label(btex 8 etex, (0,0));
115 \end{mpostfig}
116 \mpostsetup{extension=mps}
117 \\\
118 filename: \mpostgetname{ext}
119 \texttt{\mpostfigurename}

```

Change template altogether:

```

120 change template:\\
121 \mpostsetup{template=\mpostfilename-figure-#1.mps}
122 \begin{mpostfig}[label={template},show]
123 draw fullcircle scaled unit;
124 label(btex 9 etex, (0,0));
125 \end{mpostfig}
126 \mpostsetup{template=\mpostfilename-#1.mps}
127 \\\
128 filename: \mpostgetname{template}
129 \texttt{\mpostfigurename}

```

Demonstrate numbering within a section (here: subsection)

```

130 \mpostsetup{numberwithin=subsection}

```

One figure in the first section:

```

131 \subsection{Section 1}
132
133 \begin{mpostfig}[show,label={sec1}]
134 draw fullcircle scaled unit;
135 label(btex 10 etex, (0,0));
136 \end{mpostfig}
137 \\
138 filename: \mpostgetname{sec1}
139 \texttt{\mpostfigurename}

```

Two figures in the second section:

```

140 \subsection{Section 2}
141
142 \begin{mpostfig}[show,label={sec2}]
143 draw fullcircle scaled unit;
144 label(btex 11 etex, (0,0));
145 \end{mpostfig}
146 \\
147 filename: \mpostgetname{sec2}
148 \texttt{\mpostfigurename}
149
150 \begin{mpostfig}[show,label={sec3}]
151 draw fullcircle scaled unit;
152 label(btex 12 etex, (0,0));
153 \end{mpostfig}
154 \\
155 filename: \mpostgetname{sec3}
156 \texttt{\mpostfigurename}

```

One figure in the third section:

```

157 \subsection{Section 3}
158
159 \begin{mpostfig}[show,label={sec4}]
160 draw fullcircle scaled unit;
161 label(btex 13 etex, (0,0));
162 \end{mpostfig}
163 \\
164 filename: \mpostgetname{sec4}
165 \texttt{\mpostfigurename}

```

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

```

166 \makeatletter
167 \def\thempi@count{\arabic{mpi@count}}
168 \makeatother

```

B.5 Multiple Files

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

```

169 \section{Multiple Files}

```

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

```

170 \mpostdone

```

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

```
171 \mpostsetup{globaldef=true}
172
173 \begin{mpostdef}
174 unit:=0.8cm;
175 \end{mpostdef}
176
177 \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:

```
178 \mpostfile[include]{\jobname-inc}
179
180 \begin{mpostdef}
181 def drawcircle =
182 draw fullcircle scaled unit
183 enddef;
184 \end{mpostdef}
185
186 \mpostdone
```

The first file contains one figure:

```
187 \subsection*{Section 1}
188 \mpostfile{\jobname-sec1}
189
190 \begin{mpostfig}[show,label={sec5}]
191 drawcircle;
192 label(btex 14 etex, (0,0));
193 \end{mpostfig}
194 \\\
195 filename: \mpostgetname{sec5}
196 \texttt{\mpostfigurename}
197
198 \mpostdone
```

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

```
199 \subsection*{Section 2}
200 \mpostfile{\jobname-sec2}
201
202 \begin{mpostfig}[show,label={sec6}]
203 drawcircle;
204 label(btex 15 etex, (0,0));
205 \end{mpostfig}
206 \\\
207 filename: \mpostgetname{sec6}
208 \texttt{\mpostfigurename}
209
210 \begin{mpostdef}
211 unit:=1.5cm;
212 \end{mpostdef}
213
214 \begin{mpostfig}[show,label={sec7}]
215 drawcircle;
216 label(btex 16 etex, (0,0));
217 \end{mpostfig}
218 \\\
```

```

219 filename: \mpostgetname{sec7}
220 \texttt{\mpostfigurename}
221
222 \mpostdone

```

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

```

223 \subsection*{Section 3}
224 \mpostfile{\jobname-sec3}
225
226 \begin{mpostfig}[show,label={sec8}]
227 drawcircle;
228 label(btex 17 etex, (0,0));
229 \end{mpostfig}
230 \
231 filename: \mpostgetname{sec8}
232 \texttt{\mpostfigurename}
233
234 \mpostdone

```

End of document body:

```

235 \end{document}

```

C 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.

```

236 \RequirePackage{verbatim}
237 \RequirePackage{graphicx}
238 \RequirePackage{keyval}

```

Internal Definitions.

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

```

239 \newif\ifmpi@infile\mpi@infilefalse
240 \newif\ifmpi@inbody\mpi@inbodyfalse

```

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

```

\mpi@extension
\mpi@template
241 \def\mpostfilename{\jobname}
242 \def\mpi@nowname{\jobname-tmp}
243 \def\mpi@extension{mps}
244 \def\mpi@template#1{\mpostfilename-#1.\mpi@extension}

```

`mpi@count` Declare a counter for figure filenames:

```

245 \newcounter{mpi@count}
246 \def\thempi@count{\arabic{mpi@count}}

```

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

```

247 \newif\ifmpi@draft\mpi@draftfalse
248 \newif\ifmpi@latex\mpi@latextrue
249 \newif\ifmpi@fonts\mpi@fontsfalse
250 \newif\ifmpi@write\mpi@writetrue
251 \newif\ifmpi@compile\mpi@compiletrue
252 \newif\ifmpi@twice\mpi@twicefalse
253 \newif\ifmpi@lineno\mpi@linenofalse
254 \newif\ifmpi@labelnames\mpi@labelnamesfalse
255 \newif\ifmpi@nowactive\mpi@nowactivefalse
256 \newif\ifmpi@now\mpi@nowfalse
257 \newif\ifmpi@nowkeep\mpi@nowkeepfalse
258 \newif\ifmpi@include\mpi@includefalse
259 \newif\ifmpi@defglobal\mpi@defglobalfalse

```

`\mpi@postmem` These definitions store the options for processing labels via \TeX or \LaTeX :

```

\mpi@postcompiler 260 \def\mpi@postmem{}
\mpi@texcompiler   261 \def\mpi@postcompiler{mpost}
\mpi@latexclass    262 \def\mpi@texcompiler{latex}
\mpi@latexoptions  263 \def\mpi@latexclass{article}
\mpi@documentclass 264 \def\mpi@latexoptions{}
                  265 \def\mpi@documentclass{\@backslashchar documentclass%
                  266 \mpi@latexoptions{\mpi@latexclass}}

```

`\mpi@warncompile` Warn and disable compiling if `\write18` is unavailable:

```

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

```

Process package options:

```

271 \def\mpi@group{mpi@}
272 \DeclareOption{final}{\mpi@draftfalse}
273 \define@key{mpi@group}{draft}[true]{\csname mpi@draft#1\endcsname}
274 \define@key{mpi@group}{write}[true]{\csname mpi@write#1\endcsname}
275 \define@key{mpi@group}{latex}[true]{\csname mpi@latex#1\endcsname}%
276 \ifmpi@latex%
277 \def\mpi@texcompiler{latex}%
278 \else%
279 \def\mpi@texcompiler{tex}%
280 \fi}
281 \define@key{mpi@group}{compile}[true]{\csname mpi@compile#1\endcsname}
282 \define@key{mpi@group}{twice}[true]{\csname mpi@twice#1\endcsname}
283 \define@key{mpi@group}{fonts}[true]{\csname mpi@fonts#1\endcsname}
284 \define@key{mpi@group}{lineno}[true]{\csname mpi@lineno#1\endcsname}
285 \define@key{mpi@group}{labelnames}[true]{\csname mpi@labelnames#1\endcsname}
286 \define@key{mpi@group}{compiler}{\def\mpi@texcompiler{#1}}
287 \define@key{mpi@group}{mem}{\def\mpi@postmem{#1}}
288 \define@key{mpi@group}{command}{\def\mpi@postcompiler{#1}}
289 \define@key{mpi@group}{class}{\def\mpi@latexclass{#1}}
290 \define@key{mpi@group}{classopt}{\def\mpi@latexoptions{[#1]}}
291 \define@key{mpi@group}{now}[true]{\csname mpi@nowactive#1\endcsname}
292 \define@key{mpi@group}{nowall}[true]{\csname mpi@now#1\endcsname}
293 \define@key{mpi@group}{nowkeep}[true]{\csname mpi@nowkeep#1\endcsname}

```

```

294 \define@key{\mpi@group}{globaldef}[true]{\csname mpi@defglobal#1\endcsname}
295 \define@key{\mpi@group}{extension}{\def\mpi@extension{#1}}
296 \define@key{\mpi@group}{template}{\def\mpi@template##1{#1}}
297 \define@key{\mpi@group}{numberwithin}{%
298   \@addtoreset{\mpi@count}{#1}%
299   \def\thempi@count{\arabic{#1}-\arabic{\mpi@count}}}%
300 }

```

Pass undeclared options on to keyval processing:

```

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

```

Process package options and warn if \write18 mechanism is not available:

```

303 \ProcessOptions
304 \mpi@warncompile

```

Internal Commands and Definitions.

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

```

305 \def\mpi@empty{}

```

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

```

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

```

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

```

\mpi@outnow
307 \newwrite\mpi@out
308 \newwrite\mpi@outnow

```

`\mpi@write` Write to the file:

```

309 \def\mpi@write#1{\ifmpi@write\immediate\write\mpi@out{#1}\fi}

```

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

```

\mpi@nowbuf
310 \newtoks\mpi@buf
311 \newtoks\mpi@defbuf
312 \newtoks\mpi@nowbuf
313 \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:

```

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

```

`\mpi@clearbuf` `\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`:

```

316 \def\mpi@clearbuf{\global\mpi@buf={}}
317 \def\mpi@addbufexp#1{\mpi@addtoexp\mpi@buf{#1^^J}}
318 \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`:

```

319 \def\mpi@stripext#1{\edef\mpi@tmp{#1}\expandafter%
320 \mpi@stripstart\expandafter{\mpi@tmp}}
321 \def\mpi@ifeq#1#2#3#4{\def\mpi@tmpa{#1}\def\mpi@tmpb{#2}%
322 \ifx\mpi@tmpa\mpi@tmpb#3\else#4\fi}
323 \def\mpi@stripstart#1{\mpi@stripfor{\@gobble}#1.\@@.}
324 \def\mpi@stripfor#1#2.#3.{%
325 \begingroup%
326 \mpi@ifeq{#3}{\@@}{%
327 \def\mpi@tmp{\def\mpi@stripped{#1.#2}}%
328 \mpi@ifeq{#1}{\@gobble}{}%
329 \mpi@ifeq{#2}{eps}{\def\mpi@tmp{\def\mpi@stripped{#1}}{}}%
330 \mpi@ifeq{#2}{mps}{\def\mpi@tmp{\def\mpi@stripped{#1}}{}}%
331 \expandafter\mpi@ifeq\expandafter{\mpi@extension}{#2}%
332 {\def\mpi@tmp{\def\mpi@stripped{#1}}{}}%
333 }%
334 }\def\mpi@tmp{\mpi@stripfor{#1.#2}#3.}%
335 \expandafter\endgroup\mpi@tmp%
336 }

```

`\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’):

```

337 \newcommand{\mpostplaceholder}[2][\parbox[c]{1in}{%
338 \hrule\vrule\hfill%
339 \parbox[c]{0pt}{\rule{0cm}{0.6in}}\makebox[0pt][c]{\scriptsize\tt #2}%
340 \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`:

```

341 \newcommand{\mpi@graphics}[2][\%
342 \IfFileExists{#2}%
343 {\edef\mpi@tmp{#2}\includegraphics[#1]{\mpi@tmp}}%
344 {\typeout{graphics file ‘#2’ missing}\mpostplaceholder[file]{#2}}%
345 }

```

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

```

346 \newcommand{\mpi@verbatim}{%
347 \@bsphack%
348 \let\do\@makeother\dospecials%
349 \catcode'\^M\active%
350 \def\verbatim@processline{\mpi@addbufexp{\the\verbatim@line}}%
351 \verbatim@start%
352 }

```

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

```

353 \newcommand{\mpi@putlineno}{%
354 \ifmpi@lineno%
355 \mpi@addbuf{\@percentchar-----}%
356 \mpi@addbuf{\@percentchar%
357 \ifx\currfilename\undefined\else\currfilename\space\fi%
358 1.\the\inputlineno}%
359 \fi%
360 }

```

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

```
361 \newcommand{\mpi@beginfig}[1]{%
362   \mpi@addbuf{filename\template \mpi@dblquotchar#1\mpi@dblquotchar;}%
363   \mpi@addbuf{beginfig(\arabic{mpi@count})}%
364 }
```

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

```
365 \newcommand{\mpi@endfig}{%
366   \mpi@addbuf{endfig;}%
367 }
```

`\mpi@declaredoc` Write `\documentclass` statement in L^AT_EX mode to buffer:

```
368 \newcommand{\mpi@declaredoc}{%
369   \ifmpi@latex%
370     \mpi@putlineno%
371     \mpi@addbuf{verbatim}%
372     \mpi@addbuf{\mpi@documentclass}%
373     \mpi@addbuf{etex}%
374   \fi%
375 }
```

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

```
376 \newcommand{\mpi@begindoc}{%
377   \ifmpi@latex%
378     \mpi@putlineno%
379     \mpi@addbuf{verbatim}%
380     \mpi@addbuf{\@backslashchar begin{document}}%
381     \mpi@addbuf{etex}%
382   \fi%
383 }
```

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

```
384 \newcommand{\mpi@enddoc}{%
385   \ifmpi@latex%
386     \mpi@putlineno%
387     \mpi@addbuf{verbatim}%
388     \mpi@addbuf{\@backslashchar end{document}}%
389     \mpi@addbuf{etex}%
390   \fi%
391 }
```

`\mpi@writehead` Write file header to buffer; declare font inclusion and draft mode, write T_EX format specifier and L^AT_EX header:

```
392 \newcommand{\mpi@writehead}{%
393   \ifmpi@fonts\mpi@addbuf{prologues:=3;} \else \mpi@addbuf{prologues:=2;} \fi%
394   \ifmpi@draft\mpi@addbuf{draft:=1;} \fi%
395   \mpi@addbuf{}%
396   \mpi@putlineno%
397   \ifx\mpi@compiler\mpi@empty \else%
398     \mpi@addbuf{verbatim}%
399     \mpi@addbuf{\@percentchar &\mpi@texcompiler}%
400     \mpi@addbuf{etex}%

```

```

401 \mpi@addbuf{}%
402 \fi%
403 \mpi@declaredoc%
404 \mpi@addbuf{}%
405 }

```

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

```

406 \newcommand{\mpi@beginfile}{%
407 \ifx\mpi@mpostmem\mpi@empty\else%
408 \mpi@addbuf{\@percentchar &\mpi@mpostmem}%
409 \fi%
410 \mpi@addbuf{\@percentchar generated from file '\jobname' by mpostinl.sty}%
411 \ifmpi@include\else%
412 \mpi@writehead%
413 \mpi@addbufexp{\the\mpi@defbuf}%
414 \fi%
415 }

```

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

```

416 \newcommand{\mpi@endfile}{%
417 \mpi@putlineno%
418 \ifmpi@include\else%
419 \mpi@addbuf{end}%
420 \fi%
421 }

```

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

```

422 \newcommand{\mpi@startfile}{%
423 \ifmpi@infile\else%

```

Prevent reopening and overwriting the previous file:

```

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

```

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

```

427 \global\mpi@infiletrue%
428 \ifmpi@write\immediate\openout\mpi@out\mpostfilename.mp\fi%
429 \mpi@clearbuf%
430 \mpi@beginfile%
431 \mpi@write{\the\mpi@buf}%

```

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

```

432 \ifmpi@include%
433 \mpi@clearbuf%
434 \mpi@putlineno%
435 \mpi@addbuf{input \mpostfilename}%
436 \mpi@addtoexp\mpi@defbuf{\the\mpi@buf^^J}%

```

If immediate mode is available fill immediate buffer with header:

```

437 \else%
438 \ifmpi@nowactive%
439 \global\mpi@nowbuf={}%

```

```

440      \mpi@addtoexp\mpi@nowbuf{\the\mpi@buf^^J}%
441      \fi%
442  \fi%
443 \fi%
444 }

```

\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:

```

445 \newcommand{\mpi@startcontent}{%
446   \mpi@startfile%
447   \ifmpi@inbody\else%
448     \global\mpi@inbodytrue%
449     \mpi@clearbuf%
450     \mpi@begindoc%
451     \mpi@write{\the\mpi@buf}%
452     \ifmpi@nowactive\mpi@addtoexp\mpi@nowbuf{\the\mpi@buf^^J}\fi%
453   \fi%
454 }

```

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

```

455 \newcommand{\mpi@compile}[1]{%
456   \ifmpi@write\ifmpi@compile%

```

Pass on interactionmode setting to METAPOST:

```

457   \def\mpi@imode{}%
458   \ifcase\the\interactionmode%
459     \def\mpi@imode{-interaction=batchmode}\or%
460     \def\mpi@imode{-interaction=nonstopmode}\or%
461     \def\mpi@imode{-interaction=scrollmode}\or%
462     \def\mpi@imode{-interaction=errorstopmode}\fi%

```

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

```

463   \def\mpi@execute{\mpi@mpostcompiler\space%
464     \mpi@imode\space%
465     \ifx\mpi@mpostmem\mpi@empty\else -mem=\mpi@mpostmem\space\fi%
466     -tex=\mpi@texcompiler\space#1}%
467   \immediate\write18{\mpi@execute}%
468   \ifmpi@twice%
469     \immediate\write18{\mpi@execute}%
470   \fi%
471 \fi\fi%
472 }

```

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

```

473 \newcommand{\mpi@closefile}{%
474   \ifmpi@infile%

```

Write end of content section (if started):

```

475   \mpi@clearbuf%
476   \ifmpi@inbody%
477     \mpi@enddoc%
478     \mpi@addbuf{}%
479   \fi%

```

Write end of file and close:

```

480 \mpi@endfile%
481 \mpi@write{\the\mpi@buf}%
482 \ifmpi@write\immediate\closeout\mpi@out\fi%

```

Compile if file contains figures:

```

483 \ifmpi@inbody\mpi@compile{\mpostfilename.mp}\fi%

```

Reset variables:

```

484 \global\mpi@infilefalse%
485 \global\let\mpostfilename\mpi@empty%
486 \global\mpi@inbodyfalse%
487 \setcounter{mpi@count}{0}%
488 \fi%
489 }

```

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

```

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

```

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

```

492 \ifmpi@nowkeep%
493 \mpi@stripext{\mpi@figfile}%
494 \edef\mpi@nowname{\mpi@stripped}%
495 \fi%

```

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

```

496 \immediate\openout\mpi@outnow\mpi@nowname.mp%
497 \immediate\write\mpi@outnow{\the\mpi@nowbuf}%
498 \immediate\write\mpi@outnow{\the\mpi@buf}%
499 \mpi@clearbuf%
500 \mpi@enddoc%
501 \mpi@addbuf{}%
502 \mpi@endfile%
503 \immediate\write\mpi@outnow{\the\mpi@buf}%
504 \immediate\closeout\mpi@outnow%
505 \mpi@compile{\mpi@nowname.mp}%
506 \fi\fi\fi%
507 }

```

Make sure to close and process the file at the end:

```

508 \AtEndDocument{\mpi@closefile}

```

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

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

```

509 \newcommand{\mpostsetup}[1]{%
510 \setkeys\mpi@group{#1}%
511 \mpi@warncompile%
512 }

```

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

```

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

```

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

```

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

```

Process optional arguments:

```

517 \mpi@deftexfalse%
518 \setkeys{mpi@def}{#1}%

```

TeX definitions cannot be in an include file:

```

519 \ifmpi@defglobal\else\ifmpi@deftex\ifmpi@include%
520 \PackageWarning{mpostinl}{tex definitions within an include file %
521 will be ignored by mpost; switching to global definition}{}%
522 \mpi@defglobaltrue%
523 \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:

```

524 \ifmpi@defglobal\else%
525 \mpi@startfile%
526 \fi%
527 \mpi@clearbuf%
528 \mpi@putlineno%
529 \ifmpi@deftex%
530 \mpi@addbuf{verbatimtex}%
531 \fi%
532 \mpi@verbatim%
533 }%

```

Postprocessing; add ‘`etex`’ if in TeX mode, add to appropriate buffer(s).

```

534 {%
535 \ifmpi@deftex%
536 \mpi@addbuf{etex}%
537 \fi%
538 \ifmpi@defglobal%
539 \mpi@addtoexp\mpi@defbuf{\the\mpi@buf^^J}%
540 \ifmpi@include\else\ifmpi@infile%
541 \mpi@write{\the\mpi@buf}%
542 \ifmpi@nowactive%
543 \mpi@addtoexp\mpi@nowbuf{\the\mpi@buf^^J}%
544 \fi%
545 \fi\fi%
546 \else%
547 \mpi@write{\the\mpi@buf}%
548 \ifmpi@include\else\ifmpi@nowactive%
549 \mpi@addtoexp\mpi@nowbuf{\the\mpi@buf^^J}%
550 \fi\fi%
551 \fi%
552 \@esphack%
553 }

```

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

```

554 \newif\ifmpi@figshow
555 \define@key{mpi@fig}{show}[true]{\csname mpi@figshow#1\endcsname}

```

```

556 \define@key{mpi@fig}{twice}[true]{\csname mpi@twice#1\endcsname}
557 \define@key{mpi@fig}{file}{\def\mpi@figfile{#1}}
558 \define@key{mpi@fig}{label}{\def\mpi@figlabel{#1}}
559 \define@key{mpi@fig}{opt}{\edef\mpi@figopt{[#1]}}
560 \define@key{mpi@fig}{now}[true]{\csname mpi@now#1\endcsname}

```

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

```

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

```

Make sure that include files do not contain figures:

```

562 \ifmpi@include%
563 \PackageError{mpostinl}{cannot write figure to include file}{}%
564 \fi%

```

Process optional arguments:

```

565 \def\mpi@figfile{}%
566 \def\mpi@figlabel{}%
567 \def\mpi@figopt{}%
568 \mpi@figshowfalse%
569 \setkeys{mpi@fig}{#1}%

```

Display figure if no filename or label is provided:

```

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

```

Compose filename from label (if desired and specified):

```

571 \ifmpi@labelnames\ifx\mpi@figfile\mpi@empty\ifx\mpi@figlabel\mpi@empty\else%
572 \edef\mpi@figfile{\mpi@template{\mpi@figlabel}}%
573 \fi\fi\fi%

```

Compose filename from counter if no filename is provided:

```

574 \ifx\mpi@figfile\mpi@empty%
575 \addtocounter{mpi@count}{1}%
576 \edef\mpi@figfile{\mpi@template{\thempi@count}}%
577 \fi%

```

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

```

578 \ifx\mpi@figlabel\mpi@empty\else%
579 \expandafter\ifx\csname mpi@l@mpi@figlabel\endcsname\relax\else%
580 \PackageWarning{mpostinl}{label '\mpi@figlabel' already defined; %
581 overwriting}{}%
582 \fi%
583 \expandafter\xdef\csname mpi@l@mpi@figlabel\endcsname{\mpi@figfile}%
584 \fi%

```

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

```

585 \mpi@startcontent%
586 \mpi@clearbuf%
587 \mpi@putlineno%
588 \mpi@beginfig{\mpi@figfile}%
589 \mpi@verbatim%
590 }%

```

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

```

591 {%

```

```

592 \mpi@endfig%
593 \mpi@write{\the\mpi@buf}%
594 \ifmpi@now%
595 \mpi@processnow%
596 \fi%
597 \@esphack%

```

Display figure:

```

598 \ifmpi@figshow%
599 \expandafter\mpi@graphics\mpi@figopt{\mpi@figfile}%
600 \fi%
601 }

```

\mpostuse `\mpostuse` includes a metapost figure which was declared earlier via its label. The optional argument is passed as the optional argument for `\includegraphics`

```

602 \newcommand{\mpostuse}[2][{}]{%
603 \expandafter\ifx\csname mpi@l@#2\endcsname\relax%
604 \PackageWarning{mpostinl}{unknown label ‘#2’}{}%
605 \mpostplaceholder[label]{#2}%
606 \else%
607 \mpi@graphics[#1]{\csname mpi@l@#2\endcsname}%
608 \fi%
609 }

```

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

```

610 \newcommand{\mpostgetname}[1]{%
611 \expandafter\ifx\csname mpi@l@#1\endcsname\relax%
612 \PackageWarning{mpostinl}{unknown label ‘#1’}{}%
613 \let\mpostfigurename\relax%
614 \else%
615 \edef\mpostfigurename{\csname mpi@l@#1\endcsname}%
616 \fi%
617 }

```

\mpostfile Declare options for `\mpostfile`:

```

618 \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.

```

619 \newcommand{\mpostfile}[2][{}]{%
620 \mpi@closefile%
621 \mpi@includefalse%
622 \setkeys{mpi@file}{#1}%
623 \xdef\mpostfilename{#2}%
624 }

```

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

```

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

```

D Copyright

Copyright © 2010–2017 Niklas Beisert

This work may be distributed and/or modified under the conditions of the L^AT_EX 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 <http://www.latex-project.org/lppl.txt> and version 1.3 or later is part of all distributions of L^AT_EX version 2005/12/01 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 `mpostinl.dtx` and `mpostinl.ins` and the derived files `mpostinl.sty` and `mpinlsmp.tex`