glossary.sty v 2.4: $\LaTeX 2_\varepsilon$ Package to Assist Generating Glossaries

Nicola L.C. Talbot

20th July 2006

Contents

1 Introduction

The glossary package is provided to assist generating a glossary. It has a certain amount of flexibility, allowing the user to customize the format of the glossary, and define new glossary-style objects.

2 Installation

You need to make sure you have downloaded the following three files:

```
glossary.ins
glossary.dtx
README
```

To extract the code from the `glossary.dtx` file, you will need to run the installation file through LaTeX:

```
latex glossary.ins
```

This will create the following files:

```
glossary.sty
glossary.perl
makeglos.pl
makeglos.bat
```

along with several sample files. The file `glossary.sty` should be placed somewhere in the LaTeX path, e.g. `texmf/tex/latex/glossary/` or `localtexmf/tex/latex/glossary/`. Remember to update the TeX database if you are installing this package for the first time. The file `glossary.perl` is a LaTeX2HTML style file, and should be placed in the LaTeX2HTML style file directory (usually `latex2html/styles/`). The file `makeglos.pl` is a Perl script which calls `makeindex`. If you are using UNIX or Linux, you will need to set the permissions so that you can execute the file:

```
chmod a+x makeglos.pl
```

You should then place this file somewhere on your path. (You may also need to edit the first line of this file, if `perl` is located in a directory other than `/usr/bin/`[1].)

If you are not using UNIX or Linux etc, you may have to explicitly load the file into Perl, so you would need to do `perl makeglos.pl` instead of just `makeglos.pl`. If you are using Windows, a batch file, `makeglos.bat` is provided which will run Perl on `makeglos.pl`. Both `makeglos.pl` and `makeglos.bat` should be placed somewhere specified by the PATH environment variable. (For example, put them both in the same directory as `makeindex`, which will probably be in `\texmf\miktex\bin\`).

If you don't have Perl installed on your system, you can just use `makeindex`, only you will have to remember all the command line switches, and you won't be able to merge entries that have the same name, but different descriptions.

Note that if you are updating the glossary package, it is a good idea to update `makeglos.pl` as it may also have been modified.

3 Generating Glossary Information

`\makeglossary` The standard LaTeX command `\makeglossary` (analogous to `\makeindex`) should be placed in the document preamble. If this command is omitted, glossary information will be ignored. Glossary entries are generated using the command

`\glossary` `\glossary{`⟨*key-val list*⟩`}`. This command is a slightly modified version of the standard `\glossary` command, in order to separate out the information into ⟨*entry-name*⟩ and ⟨*entry-description*⟩. The argument to `\glossary` must be a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs. The following keys are available:

Key	**Value**
`name`	The entry name
`description`	A description about the entry
`sort`	How to sort the entry. (Entry name used if sort omitted)
`format`	How to format the page number
`number`	Override the page number with a different counter. The value should be the name of a counter (e.g. `number=section`).

For example:

```
\glossary{name={singular matrix},
          description={A matrix with zero determinant}}
```

The following example sorts on the text `U` instead of `$\mathcal{U}$`:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}
```

Note that you should always use the `sort` key if the `name` key contains commands, this is particularly important if you are using hyperlinks, as the target is constructed from the `name` key if the `sort` key is omitted.

In the glossary, each entry is followed by a list of page numbers that correspond to the pages where the relevant `\glossary` command is placed. By default the numbers are formatted in the current font, but the page number format for

[1] and you can also remove the `.pl` extension which isn't to everyone's liking.

individual entries can be changed using the `format` key. This should be the name of a LaTeX formatting command without the preceding `\` (as with the `|` operator in `\index`.) For example:

```
\glossary{name={$\mathbb{R}$},
          description={The set of real numbers},
          sort=R,
          format=textbf}
```

In addition, the following formats are also available:

`hyperrm`	The number is a hyper link in roman
`hypersf`	The number is a hyper link in sans-serif
`hypertt`	The number is a hyper link in typewriter font
`hyperbf`	The number is a hyper link in bold
`hypermd`	The number is a hyper link in medium weight
`hyperit`	The number is a hyper link in italic
`hypersl`	The number is a hyper link in slanted font
`hyperup`	The number is a hyper link in upright font
`hypersc`	The number is a hyper link in small caps
`hyperem`	The number is a hyper link using `\emph`

If the `hyper` option has not been set, `hyperem` is equivalent to `emph`, and the remaining `hyperrm` etc are equivalent to `textrm` etc. Note that it is important the you use `hyperrm` instead of `hyperpage`, as the `\hyperpage` command won't work on a list or range of numbers in the glossary [2]. If you want to define your own command that uses hyper links, it must be defined in an analogous manner to `\hyperrm`. For example, if you want to display a page number in a bold italic format, that contains a hyperlink to the appropriate page, you would need to define it as follows:

```
\newcommand{\hyperbfit}[2][\gls@number]{%
\textbf{\itshape\glshyper{#1}{#2}}}
```

As can be seen from the definition, all the `\hyper`⟨*xx*⟩ commands have an optional argument. This argument is the name of the counter being used. You do not need to worry about this argument if you only use these commands within the `\glossary` command. So the previous example can simply be rewritten as:

```
\glossary{name={$\mathbb{R}$},
          description={The set of real numbers},
          sort=R,
          format=hyperbf}
```

Note: although the numbers in the glossary are referred to as "page" numbers in this manual, they may in fact refer to some other counter, such as the `section` counter, depending on whether the `number` key has been used.

As with the `\index` command, care must be taken if you want to use the special characters: `@ | "` and `!`. These characters should be preceded by the double quote character. For example:

```
\glossary{name={$"|\mathcal{S}"|$,
description=The cardinality of the set \mathcal{S}}}
```

[2]This is because the list and number ranges are delimited using `\delimR` and `\delimN` instead of explicitly using a comma or en-dash.

There is no provision for sub-entries, as these are generally only applicable in an index, and not in a glossary.

`\xglossary` As from version 2.14, there is an additional command available:

`\xglossary{`⟨*gls-entry*⟩ `}{`⟨*text*⟩`}`

This is equivalent to⟨*text*⟩`\glossary{`⟨*gls-entry*⟩`}`, where ⟨*text*⟩ will be made a hyper link to the relevant entry in the glossary, if hyper links are supported.

3.1 Storing Glossary Information

It is very cumbersome having to use the `\glossary` command throughout your document, every time you use a term that you want in your glossary. This is particularly true for terms with a long description. The glossary package provides a means of storing the glossary information at the beginning of the document, and then using it whenever required. It is strongly recommended that you use this approach, rather than explicity using the `\glossary` command.

`\storeglosentry` The following command:

`\storeglosentry[`⟨*gls-type*⟩`]{`⟨*label*⟩`}{`⟨*gls-entry*⟩`}`

can be used to store glossary information, where ⟨*label*⟩ is a unique label assigned to this entry. The information can then be used later with any of the following commands:

`\useglosentry` `\useglosentry[`⟨*opt*⟩`]{`⟨*label*⟩`}`
`\useGlosentry` `\useGlosentry[`⟨*opt*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`
`\gls` `\gls[`⟨*opt*⟩`]{`⟨*label*⟩`}`

`\useglosentry` adds the glossary entry whose label is given by ⟨*label*⟩ to the appropriate glossary, `\useGlosentry` adds the glossary entry, and makes ⟨*text*⟩ a hyperlink to that entry (if hyperlinks are supported). The third command, `\gls`, is like `\useGlosentry`, but forms ⟨*text*⟩ from the name given in the glossary entry.

Returning to an earlier example, instead of typing:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}
```

every time you want to add this entry to the glossary, you can instead store the information:

```
\storeglosentry{glos:U}{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}
```

Now, instead of continually copying and pasting the glossary command for this entry (which can have quite a large `description` field), you can use either:

```
\useglosentry{glos:U}
```

which is equivalent to:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}
```

or you can use:

```
\useGlosentry{glos:U}{text}
```

which is equivalent to:

```
\xglossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}{text}
```

or you can use:

```
\gls{glos:U}
```

which is equivalent to:

```
\xglossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}{$\mathcal{U}$}
```

If you want to use glossary entries in an equation, it is better to use `\ensuremath` instead of `$...$`. For example:

```
\storeglosentry{Gamma}{name=\ensuremath{\Gamma(z)},
description=Gamma function,
sort=Gamma}
```

You can then use this entry in either text or math mode:

```
The \useGlosentry{Gamma}{Gamma function} is defined as
\begin{equation}
\gls{Gamma} = \int_{0}^{\infty}e^{-t}t^{z-1}\,dt
\end{equation}
```

If you are using hyper links, and you want to use `\useGlosentry` within math mode, you must use `\ensuremath`:

```
\begin{equation}
\useGlosentry{Gamma}{\ensuremath{\Gamma(x+1)}} = x\Gamma(x)
\end{equation}
```

The optional argument to `\storeglosentry` (⟨*gls-type*⟩) indicates the glossary type (see section 7 to find out how to define new glossary types). If omitted, the standard glossary is used.

The optional argument to `\useglosentry`, `\useGlosentry` and `\gls` (⟨*opt*⟩) allows you to add additional information to the glossary entry, for example:

```
\useglosentry[format=textbf]{glos:U}
```

is equivalent to:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U,
          format=textbf}
```

Since version 2.4, `\storeglosentry` is robust, and `\protect` should no longer be needed, however the identifying label, ⟨*label*⟩, should not contain any special characters.

As from version 2.36, if you want to use all glossary entries which have been defined using `\storeglosentry`, do: `\useglosentry{*}`. (Note that this option is not available for `\useGlosentry` and `\gls`.)

4 makeglos.pl

Whenever a glossary entry is used, either explicity using `\glossary` or `\xglossary` or implicitly using `\useglosentry`, `\useGlosentry` and `\gls`, the information is saved in a file with the extension `glo` (unless the `\makeglossary` command is omitted, in which case the glossary information is simply ignored.) A `makeindex` style file (`ist`) is also created, which is customized for the document, and can be passed to `makeindex`.

For example, suppose your document is called `mydoc.tex`, the glossary will be saved in the file `mydoc.glo`, and the `makeindex` style file `mydoc.ist` will be created. These files can then be passed to `makeindex` as follows:

```
makeindex -s mydoc.ist -t mydoc.glg -o mydoc.gls mydoc.glo
```

which generates the output file `mydoc.gls`, with transcript written to `mydoc.glg`.

The Perl script `makeglos.pl` provided with this package allows you to use `makeindex` without having to remember all the command line options. The command

```
makeglos.pl mydoc
```

will perform the command:

```
makeindex -s mydoc.ist -t mydoc.glg -o mydoc.gls mydoc.glo
```

In addition, `makeglos.pl` also takes the option `-m` which can be used to collate entries where the same name has multiple descriptions.

`makeglos.pl` has the following syntax:

```
makeglos.pl [-ilqrgm] [-s sty] [-o gls] [-t log] [-p num] <filename>
```

where all switches, apart from `-m` are the same as those for `makeindex`. If there are multiple glossary types (see section 7) and the file extension is omitted, `makeglos.pl` will iterate through each glossary type (it will pick up the relevant information from the auxiliary file).

`\istfilename` The name of the `ist` file can be changed by redefining the command `\istfilename`*before* `\makeglossary`. For example:

```
\renewcommand{\istfilename}{foo.ist}
\makeglossary
```

Only one `ist` file will be created per document, even if you have multiple glossaries with different styles. The only circumstance where you will need multiple `ist` files for a single document is when you have multiple glossaries that use different counters with different compositors, but this is rarely likely to occur.

`\noist` Creation of the `ist` file can be suppressed by issuing the command `\noist` before `\makeglossary`. It will also be suppressed when the command `\nofiles` is used, or if the command `\makeglossary` is omitted.

It should be noted that there are a few packages that can cause problems with the creation of the `ist` file, for example `ngerman`. If you encounter problems when LaTeX is processing the `\makeglossary` command, or if you get errors from `makeindex` complaining about the style file, this is the most probable cause. See section 12, item 16 for information on how to fix this.

5 Displaying the Glossary

Once the `gls` file has been created by `makeindex` (as described in the previous section) the glossary can then be included in the document with the command `\printglossary`. If chapters are defined, the glossary will start with

`\printglossary`

```
\chapter*{\glossaryname}
```

If not, it will start with

```
\section*{\glossaryname}
```

The format of the main body of the glossary depends on the options passed to the package.

6 Package Options

The package options must be specified as a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs. Available options are:

`style` The glossary style. Values:

- `list` use description environment in the glossary
- `altlist` modified version of `style=list`. The description starts on the line following the name of the term being defined.
- `super` use supertabular environment in the glossary
- `long` use longtable environment in the glossary (Default)

`header` Glossary header. Values:

- `none` The glossary doesn't have a heading (Default)
- `plain` The glossary has a heading

`border` Glossary border. Values:

- `none` The glossary doesn't have a border (Default)
- `plain` Border around the main body of the glossary

`cols` Number of columns. Values:

- `2` The entry name and description are in two separate columns with the associated page numbers in the same column as the description. (Default)
- `3` The entry name, description and associated page numbers are in three separate columns.

`number` Associated number corresponding to each entry. This may either be the keyword `none` indicating that the corresponding numbers should be suppressed, or it can be the name of a LaTeX counter. The default is `number=page`.

`toc` Boolean variable:

`true` Add glossary to table of contents

`false` Omit glossary from table of contents (Default)

Note that if you specify this option, you will need to run LaTeX twice after generating the glossary.

`hypertoc` Boolean variable. This is similar to the package option `toc`, but if you are using the hyperref package, `hypertoc` will generate a link to the point immediately before the glossary title, whereas `toc` will have a hyperlink to just after the glossary title. Note that you can not use both `toc=true` and `hypertoc=true`. Default value: `hypertoc=false`.

`hyper` Boolean variable:

`true` Make associated numbers in the glossary a hypertext link, and also make acronyms, and the text given by `\xglossary` have a hyperlink to their corresponding entries in the glossary.

`false` Don't make associated numbers a hypertext link

If the hyperref or html package has been loaded prior to loading `glossary.sty`, `hyper=true` is set, otherwise the default is `hyper=false`. Note that this package option now encompasses the old `hyperacronym` option.

`section` Boolean variable:

`true` Make the glossary an unnumbered section, even if chapters are defined

`false` Only make glossary an unnumbered section if chapters are not defined (default).

`acronym` Boolean variable:

`true` Make the list of acronyms separate from the main glossary.

`false` The acronyms will all be placed in the main glossary. (Default)

`global` Boolean variable:

`false` Acronym commands only have a local effect. (Default)

`true` Acronym commands have a global effect.

The `border`, `header` and `cols` options should not be used in conjunction with `style=list` or `style=altlist`, as they only make sense with one of the tabular-style options. The value for the boolean variables can be omitted if they are to be set. For example `toc` is equivalent to `toc=true`. Note that the `altlist` style is better suited to glossaries with long entry names.

You can set up your own preferred defaults in a configuration file . The file must be called `glossary.cfg` and should be placed somewhere on the TeX path. In this file you can use the command `\glossarypackageoptions{`⟨*option-list*⟩`}` where ⟨*option-list*⟩ is a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs, as passed to the glossary package. Note that this command may only be used in the configuration file.

`\glossarypackageoptions`

6.1 Examples

Suppose the document has the following `\glossary` commands:

Page	Command
1	`\glossary{name=diagonal matrix, description=Matrix whose only non-zero entries are along the leading diagonal}`
2	`\glossary{name=identity matrix, description=Diagonal matrix with 1s along the leading diagonal}`
4	`\glossary{name=singular matrix, description=Matrix with zero determinant}`

Variations:

1. If `style=list` is chosen, the glossary will look like:

   diagonal matrix Matrix whose only non-zero entries are along the leading diagonal, 1

   identity matrix Diagonal matrix with 1s along the leading diagonal, 2

   singular matrix Matrix with zero determinant, 4

2. If `style=altlist` is chosen, the glossary will look like:

   diagonal matrix
   Matrix whose only non-zero entries are along the leading diagonal, 1

   identity matrix
   Diagonal matrix with 1s along the leading diagonal, 2

   singular matrix
   Matrix with zero determinant, 4

3. If `style=list`,`number=none` is chosen, the glossary will look like:

   diagonal matrix Matrix whose only non-zero entries are along the leading diagonal

   identity matrix Diagonal matrix with 1s along the leading diagonal

   singular matrix Matrix with zero determinant

4. If `style=long`,`border=none`, `header=none`,`number=page` is chosen (default), the glossary will look like:

diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal, 1
identity matrix	Diagonal matrix with 1s along the leading diagonal, 2
singular matrix	Matrix with zero determinant, 4

5. If `style=long,border=plain, header=none` is chosen, the glossary will look like:

diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal, 1
identity matrix	Diagonal matrix with 1s along the leading diagonal, 2
singular matrix	Matrix with zero determinant, 4

6. If `style=long,border=plain, header=plain` is chosen, the glossary will look like:

Notation	**Description**
diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal, 1
identity matrix	Diagonal matrix with 1s along the leading diagonal, 2
singular matrix	Matrix with zero determinant, 4

7. If `style=long,border=none, header=plain,cols=3` is chosen, the glossary will look like:

Notation	**Description**	
diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal	1
identity matrix	Diagonal matrix with 1s along the leading diagonal	2
singular matrix	Matrix with zero determinant	4

7 Defining New Glossary Types

`\newglossarytype` A new type of glossary can be defined using the command:

`\newglossarytype[`⟨*log-ext*⟩`]{`⟨*name*⟩`}{`⟨*out-ext*⟩`}{`⟨*in-ext*⟩`}[`⟨*style list*⟩`]`

For example, suppose you want your document to have a separate index of terms and index of notation, you could use `\makeglossary`, `\glossary`, `\xglossary` and `\printglossary` for the first glossary, and define a new type of glossary called say, `notation`, using

```
\newglossarytype[nlg]{notation}{not}{ntn}
```

which will create the analogous commands: `\makenotation`, `\notation`, `\xnotation` and `\printnotation` which can be used for the second glossary.

As from version 2.3, `\newglossarytype` now has an additional optional argument ⟨*style list*⟩. This should be a comma separated list of ⟨*key*⟩=⟨*value*⟩ pairs that can be used to specify the style of the new glossary. If omitted, the new glossary will have the same format as the main glossary. The following options are available: `number`, `style`, `header`, `border` and `cols`. These can take the same values as those given in the package options (described in section 6).

The command `\newglossarytype` should only occur in the preamble. The new commands `\make`⟨*name*⟩, `\`⟨*name*⟩, `\x`⟨*name*⟩ and `\print`⟨*name*⟩ all have the same format as their "glossary" counter-parts.

The glossary information will be saved to a file with the extension given by ⟨*out-ext*⟩ (analogous to `glo`), which can then be passed to `makeindex` either directly or via `makeglos.pl`, and the file to be read in (i.e. the file created by `makeindex`) will have the extension ⟨*in-ext*⟩ (analogous to `gls`).

The optional argument ⟨*log-ext*⟩ indicates the extension for the `makeindex` log file, if omitted the extension `glg` is used. This is not used by LaTeX, however `makeglos.pl` reads in this information from the LaTeX auxiliary file and passes it to `makeindex`.

For the above `notation` example, if your document is called, say, `mydoc.tex`, you will need to do the following:

```
latex mydoc
makeglos.pl mydoc
latex mydoc
```

(You may need to do an extra `latex mydoc` to get cross-references up-to-date.) Note that if you don't specify the file extension when using `makeglos.pl`, it will check the transcript file from the LaTeX run to determine all the glossary types, so, for this example,

```
makeglos.pl mydoc
```

is equivalent to:

```
makeglos.pl mydoc.glo
makeglos.pl mydoc.not
```

since `makeglos.pl` has read in the information for the `notation` glossary type from the file `mydoc.log`.

If you don't have Perl installed on your system, or for any other reason are unable to use `makeglos.pl`, you can call `makeindex` explicitly:

```
latex mydoc
makeindex -s mydoc.ist -t mydoc.glg -o mydoc.gls mydoc.glo
makeindex -s mydoc.ist -t mydoc.nlg -o mydoc.ntn mydoc.not
latex mydoc
```

Note that you can use the command `\printglossary[`⟨*name*⟩`]` instead of `\print`⟨*name*⟩. These two commands have the same effect when using LaTeX, however, they have a slightly different effect when using LaTeX2HTML (see section 11).

If the command `\`⟨*glossary-type*⟩`name` is defined, (e.g. `\notationname` in the above example) this will be used as the title for the specified glossary. If this command is not defined, `\glossaryname` will be used instead. If the command `\short`⟨*glossary-type*⟩`name` is defined, (e.g. `\shortnotationname` in the above example) this will be used for the table of contents entry, otherwise `\`⟨*glossary-type*⟩`name` will be used instead. For example:

```
\newglossarytype[nlg]{notation}{not}{ntn}
\newcommand{\notationname}{Index of Notation}
\newcommand{\shortnotationname}{Notation}
```

8 Acronyms

`\newacronym` The glossary package provides the command:

`\newacronym[`⟨*cmd-name*⟩`]{`⟨*acronym*⟩`}{`⟨*long*⟩`}{`⟨*glossary entry*⟩`}`

which can be used to define acronyms. The argument ⟨*long*⟩ is the full name, the argument ⟨*acronym*⟩ is the acronym for ⟨*long*⟩ and ⟨*glossary entry*⟩ is the glossary information in the form used by the `\glossary` command. If the optional argument ⟨*cmd-name*⟩ is missing, `\newacronym` will create a command called `\`⟨*acronym*⟩, otherwise it will create a command called `\`⟨*cmd-name*⟩ (henceforth denoted `\`⟨*acr-name*⟩). This command can then be used throughout the text. The first instance of this command is equivalent to:

⟨*long*⟩ `(\xacronym{name=`⟨*long*⟩ `(`⟨*acronym*⟩`),`⟨*glossary entry*⟩`}{`⟨*acronym*⟩`})`

subsequent instances will be equivalent to:

`\xacronym{name=`⟨*long*⟩ `(`⟨*acronym*⟩`),`⟨*glossary entry*⟩`}{`⟨*acronym*⟩`}`

The command `\`⟨*acr-name*⟩ also has a starred version, which will make the first letter of ⟨*long*⟩ uppercase (for use at the start of a sentence).

Note that if you want to change the format of the acronym, for example, if you want the acronym to appear in small caps, you will need to not only use the optional argument, but you will also need to use the `sort` key, otherwise you will get an error. For example:

```
\newacronym[SVM]{\textsc{svm}}{Support Vector Machine}%
{description=Statistical pattern recognition
technique,sort=svm}
```

If the package option `acronym` is not set (default) `\xacronym`, is a synonym for `\xglossary`, and the acronyms will appear in the main glossary (remember to specify `\makeglossary` and `\printglossary`). If the package option `acronym=true` is specified, a new glossary type called `acronym` will be defined as:

```
\newglossarytype[alg]{acronym}{acr}{acn}
\providecommand{\acronymname}{List of Acronyms}
```

You will then need to use the commands `\makeacronym` and `\printacronym` to make the list of acronyms appear. You will also need to run the `acr` file through `makeindex` (or `makeglos.pl`). For example:

```
makeindex -s mydoc.ist -t mydoc.alg -o mydoc.acn mydoc.acr
```

alternatively:

```
makeglos.pl mydoc
```

Note that the package option `acronym=true` is only appropriate if you want both a glossary and a separate list of acronyms. If you do not write in English, you can set up your own language definition for `\acronymname` in the configuration file `glossarycfg`. For example:

```
\newcommand{\acronymname}{Akronyme}
```

(If `glossary.cfg` does not exist, create a new file, add the appropriate definition of `\acronymname`, and save it to the same directory as `glossary.sty`.)

The `name` key does not need to appear in ⟨*glossary entry*⟩, as it is constructed from ⟨*long*⟩ and ⟨*acronym*⟩. By default this will be in the form: ⟨*long*⟩ (⟨*acronym*⟩), however the format can be overridden using the command:

`\setacronymnamefmt`

`\setacronymnamefmt{`⟨*format*⟩`}`

Within ⟨*format*⟩ the following commands may be used to represent ⟨*long*⟩ and ⟨*acronym*⟩: `\glolong` and `\gloshort`. For example, suppose you just want the acronym to appear in the glossary entry, and not its full length name, then you would need to do:

`\glolong`
`\gloshort`

```
\setacronymnamefmt{\gloshort}
```

As from version 2.32, you can also modify the way the description key is formatted for acronyms using:

`\setacronymdescfmt`

`\setacronymdescfmt{`⟨*format*⟩`}`

Within ⟨*format*⟩ you may use the commands `\glolong` and `\gloshort` (as above), and you can also use the command `\glodesc` which is the description as specified by the `description` key in `\newacronym`. This means that if you are using a tabular style glossary, you can have the abbreviated form in one column and the long form in the second column with the description. For example, the following:

```
\setacronymnamefmt{\gloshort}
\setacronymdescfmt{\glolong: \glodesc}
\newacronym{svm}{support vector machine}{description=Statistical
pattern recognition technique}
```

will generate a glossary entry of the form:

```
\glossary{name=svm,description=support vector machine: Statistical
pattern recognition technique}
```

Note that if you omit `\glodesc` from `\setacronymdescfmt` the description specified in `\newacronym` will be ignored. So

```
\setacronymnamefmt{\gloshort}
\setacronymdescfmt{\glolong}
\newacronym{svm}{support vector machine}{description=Statistical
pattern recognition technique}
```

will generate a glossary entry of the form:

```
\glossary{name=svm,description=support vector machine}
```

You will need to specify the `name` key explicitly if the name contains a `makeindex` special character. For example:

```
\newacronym{RNA}{Ribonukleins\"aure}{name={Ribonukleins\""aure (RNA)}}
```

Note that this will override any formatting specified by `\setacronymnamefmt`.

Given an acronym named ⟨*acr-name*⟩ (the command name associated with the acronym as defined in `\newacronym` without the preceding backslash), the following commands are also available:

`\useacronym` `\useacronym[`⟨*insert*⟩`]{`⟨*acr-name*⟩`}`

This command can be used instead of `\`⟨*acr-name*⟩. `\useacronym` also has a starred version equivalent to `\`⟨*acr-name*⟩`*`. The optional argument ⟨*insert*⟩ allows you to insert text after ⟨*long*⟩, if this is the first occurrence of the acronym, or after the acronym on subsequent occurrences.

`\resetacronym` `\resetacronym{`⟨*acr-name*⟩`}`

`\resetallacronyms` This command will cause the next use of `\`⟨*acr-name*⟩ to produce the long version. To reset all acronyms do `\resetallacronyms`.

`\unsetacronym` `\unsetacronym{`⟨*acr-name*⟩`}`

`\unsetallacronyms` This command will cause all subsequent uses of `\`⟨*acr-name*⟩ to produce the short version. To unset all acronyms do `\unsetallacronyms`.

`\ifacronymfirstuse` `\ifacronymfirstuse {`⟨*acr-name*⟩ `}{`⟨*true text*⟩`}{`⟨*false text*⟩`}`

This will test if the acronym has been used yet. If it has been used, ⟨*true text*⟩ will be implemented, otherwise ⟨*false text*⟩ will be implemented.

The long and short forms of an acronym can be produced explicitly without a corresponding glossary entry, using the commands:

`\acrln` `\acrln{`⟨*acr-name*⟩`}`
`\acrsh` `\acrsh{`⟨*acr-name*⟩`}`

Or, alternatively:

`\`⟨*acr-name*⟩`long`
`\`⟨*acr-name*⟩`short`

The first two commands (`\acrln` and `\acrsh`) have a starred form that makes the first letter uppercase. The other two commands, simply contain ⟨*long*⟩ and ⟨*acronym*⟩.

Note that since these four commands do not generate glossary entries they will therefore not contain any hyperlinks, even if you have specified the `hyper` package option. They are provided for use in situations where the associated glossary command may cause problems (e.g. in a sectioning command.)

Note that, as with all LaTeX commands, spaces following command names are ignored so if, for example, you defined a new acronym called, say, SVM, then the command `\SVM` will ignore any spaces following it. To force a space, you can either place an empty set of braces after the command name (e.g. `\SVM{}`) or use `\␣` i.e.

a backslash followed by a space (e.g. `\SVM\ `). Alternatively, as from version 2.22, if you load the xspace package before loading the glossary package, spaces will be put in automatically using `\xspace`.

`\acronymfont` If you want the acronym to appear in a particular font, for example, small caps, you can redefine the command `\acronymfont`. For example:

```
\renewcommand{\acronymfont}[1]{\textsc{#1}}
```

The default definition of `\acronymfont` is:

```
\newcommand{\acronymfont}[1]{#1}
```

8.1 Examples

```
\newacronym{SVM}{Support Vector Machine}{description=Statistical
pattern recognition technique}
```

This will define the command `\SVM`. The first time this command is used will display the text: Support Vector Machine (SVM). Subsequent use will simply display: SVM. The next example uses the optional argument ⟨*cmd-name*⟩ since the acronym contains a non-alphabetical character:

```
\newacronym[KSVM]{K-SVM}{Kernel Support Vector
Machine}{description=Statistical pattern recognition
technique using the ‘‘kernel trick’’}
```

This will define the command `\KSVM`. The first time this command is used will display the text: Kernel Support Vector Machine (K-SVM). Subsequent use will simply display: K-SVM.

To test whether or not an acronym has been used:

```
\ifacronymfirstuse{SVM}{a}{an} \SVM\ is \ldots
```

If the acronym has not been used, the following text will be produced:

a Support Vector Machine is . . .

otherwise it will produce:

an SVM is . . .

To expand the acronym a second time:

```
\chapter{An overview of the \protect\SVM}
\resetacronym{SVM}
The \SVM\ \ldots
```

Note the use of `\protect` (see note 15 on page 26.) In fact, in this situation it would be better to do:

```
\chapter[An overview of the \SVMlong]{An overview of the \protect\SVM}
\resetacronym{SVM}
The \SVM\ \ldots
```

Now suppose you want the text: support vector machine, instead of Support Vector Machine (i.e. you don't like the uppercase letters). You can define the acronym as follows:

```
\newacronym{SVM}{support vector machine}{description=Statistical
pattern recognition technique}
```

however, if the command `\SVM` occurs at the start of the sentence, you would clearly want the first letter as an uppercase letter. This can be done using `\SVM*` instead of `\SVM`. For example:

```
\SVM*\ techniques are widely used \ldots
```

This will then come out as: Support vector machine (SVM) techniques are widely used . . . (Assuming this is the first use of either `\SVM` or `\SVM*`.)

Alternatively, `\useacronym{SVM}` can be used instead of `\SVM`. For example:

```
\useacronym*[s]{SVM} are widely used in the area of pattern
recognition.
```

If this is the first use of the acronym SVM, it will produce the following text:

Support vector machines (SVM) are widely used in the area of pattern recognition.

If this is not the first use of this acronym, it will produce the following text:

SVMs are widely used in the area of pattern recognition.

9 Customizing the Glossary

The glossary package provides commands which can be redefined to customize the glossary. The following name commands are defined by this package:

`\glossaryname`
`\entryname`
`\descriptionname`

Command	**Default Value**
`\glossaryname`	Glossary
`\shortglossaryname`	`\glossaryname`
`\entryname`	Notation
`\descriptionname`	Description

The commands `\entryname` and `\descriptionname` are put in the first two columns of the header row if you are using one of the tabular glossary styles together with a header row (as specified by the `header=true` package option). `\glspageheader` If you are using `cols=3`, the command `\glspageheader` will be put in the third column of the header row. By default, this command does nothing.

`\shortglossaryname` The command `\shortglossaryname` is used for the page headers and table of contents entry. Any text required before or after the glossary can be added `\glossarypreamble` `\glossarypostamble` by redefining the commands `\glossarypreamble` and `\glossarypostamble`. For example.

```
\renewcommand{\glossarypreamble}{Page numbers in
italic indicate the main definition\par}
```

By default, `\glossarypreamble` and `\glossarypostamble` do nothing.

Any text required before or after the list of page numbers are specified by the commands `\glsbeforenum` and `\glsafternum`. By default, these commands do nothing, any redefinition of these commands should come somewhere before the relevant `\printglossary` command. For example:

`\glsbeforenum` `\glsafternum`

```
\printglossary
\renewcommand{\glsbeforenum}{(}
\renewcommand{\glsafternum}{)}
\printnotation
```

This will put the page number list in brackets for the second glossary, but not the first.

Individual glossaries can have their styles changed either by setting the style in the final optional argument to `\newglossarystyle` (see section 7) or using the `setglossarystyle` command:

`setglossarystyle`

`\setglossarystyle[`⟨*type*⟩`]{`⟨*style list*⟩`}`

For example:

```
\setglossarystyle[acronym]{style=long,border=true,cols=2}
```

If ⟨*type*⟩ is omitted, the change is applied to the main glossary.

`\setglossary`

The command `\setglossary{`⟨*key-val list*⟩`}` can be used to modify some of the glossary settings. The argument ⟨*key-val list*⟩ is a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs. Available keys are:

`type` This is the glossary type. If it is omitted, the standard glossary is assumed.

`glsnumformat` This is the name of the command, *without* the preceding backslash[3], to format the entry numbers. For example, to make all the entry numbers italic, do:

```
\setglossary{glsnumformat=textit}
```

To suppress numbering altogether, you can do:

```
\setglossary{glsnumformat=ignore}
```

`glodelim` This specifies what to do after the entry description and before the page numbers. The default value is a comma, unless the `cols=3` option is specified, in which case it has the value `&`, or if `style=altlist`, in which case it is simply a space[4]. If the package option `number=none` is specified, `glodelim` will have an empty value (unless `cols=3` is specified, where, again, it will have the value `&`.) This setting corresponds to the `delim_0` key in the `makeindex` style file.

Note that if you want a new line between the description and the list of page numbers you will need to use `\noexpand`. For example:

```
\setglossary{glodelim={\noexpand\newline}}
```

[3]Note, you should no longer try redefining the command `\glsnumformat`, as this now takes an optional argument, allowing for different glossary types

[4]This is because the `altlist` style is intended for use with long descriptions that will look better ending with a full stop which the user can add if desired.

`delimN` The delimiter to be inserted between two page numbers for the same entry. (This corresponds to the `delim_n` key in the `makeindex` style file.) By default, this has the value `,␣` (comma followed by a space). If the package option `number=none` is chosen, the value is set to empty.

`delimR` The delimiter to be inserted between the starting and ending page number range for the same entry. (This corresponds to the `delim_r` key in the `makeindex` style file.) By default, this has the value `--`. If the package option `number=none` is chosen, the value is set to empty.

`gloskip` This specifies what to do between groups. If `style=list` or `style=altlist` this has the value `\indexspace`, otherwise it creates a blank row in the longtable or supertabular environment. This command corresponds to the `group_skip` key in the `makeindex` style file. Note that as from version 2.3, you should no longer redefine the command `\gloskip`.

`delimT` The text to be inserted after the list of page numbers for an entry. (This corresponds to the `delim_t` key in the `makeindex` style file.) The default value depends on the glossary style. It does nothing for the list-type styles, and has the value `\\` for the tabular-type styles. Note that `delimT` is separate from `\glsafternum`.

For example, if you are using a 2 column tabular style, and you want a blank line after every entry (not just after every group) you can do the following:

```
\setglossary{delimT={\cr & \cr},gloskip={}}
```

Note the use of `\cr` instead of `\\` and `gloskip` is set to nothing otherwise there would be a double space between groups.

Note that:

```
\setglossary{glsnumformat=ignore}
```

is equivalent to

```
\setglossary{glsnumformat=ignore,delimN={},delimR={}}
```

As from version 2.4, you can insert text between groups by redefining the commands `\glogroupSymbols`, `\glogroupNumbers`, `\glogroupA` ... `\glogroupZ`. For example, if you are using one of the list styles, the following will print the appropriate heading in bold, followed by a gap:

```
\renewcommand{\glogroupSymbols}{\textbf{Symbols}\indexspace}
\renewcommand{\glogroupNumbers}{\textbf{Numbers}\indexspace}
\renewcommand{\glogroupA}{\textbf{A}\indexspace}
....% similar lines omitted
\renewcommand{\glosgroupZ}{\textbf{Z}\indexspace}
```

`\beforeglossary` `\afterglossary` `\glossaryalignment`

The start and end of the main body of the glossary is given by the commands: `\beforeglossary` and `\afterglossary`. If the `style=list` or `style=altlist` package options are chosen these commands simply begin and end the description environment, otherwise these commands begin and end the longtable or supertabular environment with argument specified by `\glossaryalignment`[5].

[5]This isn't quite true anymore, see the documented code for clarification

The glossary package no longer conflicts with the array package. Changes can now be made to `\glossaryalignment` regardless of whether or not the array package has been used.

`\gloitem` The command `\gloitem` indicates what to do at the start of each glossary entry. This command takes one argument, which will be the text specified by the `name` key in the `\glossary` command. In the case of the `style=list` option, `\gloitem{`⟨*text*⟩`}` will do

`\item[`⟨*text*⟩`]`

or if `style=altlist`:

`\item[`⟨*text*⟩`]\mbox{}\par`

otherwise it will do

⟨*text*⟩ `&`

This command corresponds to the `item_0` key in the `makeindex` style file.

If the glossary has a tabular style with a header row (`header=true` and either `style=long` or `style=super`), then the header row for `cols=2` will be given by:

```
\bfseries\entryname & \bfseries \descriptionname\\
```

and the header row for `cols=3` will be given by:

```
\bfseries\entryname & \bfseries\descriptionname &
\bfseries\glspageheader\\
```

(It may also contain `\hline\hline` if the `border` key is set.)

`\glossaryheader` If you want to override this, you need to define the command `\glossaryheader`[6]
.

For example, if you are using a tabular style with `cols=2`, and you want the `\descriptionname` to be centred, you could do:

```
\newcommand{\glossaryheader}{\bfseries\entryname &
 \hfil\bfseries\descriptionname\\}
```

`\glossarysubheader` If you want an extra row below the header row, you can define the command`\glossarysubheader` For example, if you are using `cols=3`, and you want an extra row after the header row, you can do:

```
\newcommand{\glossarysubheader}{ & & \\}
```

`\glosstail` The command `\glosstail` indicates what to do at the end of the longtable or supertabular environment.

`\descriptionwidth` The width of the second column for the tabular-type styles is given by the length `\descriptionwidth`. This value can be changed using the `\setlength` command (the default value is `0.6\linewidth`).

10 Sample Documents

This package comes with the following sample documents:

[6]Note that as from version 2.4, you must use `\newcommand` *not* `\renewcommand`

- `sampleSec.tex` — This document uses the options: `style=altlist`, `toc` and `number=section`. It also loads the hyperref package before loading the glossary package, so the glossary has hyperlinks to the section numbers. Experimenting with different package options, will illustrate the different glossary styles. You will need to do:

  ```
  pdflatex sampleSec
  makeglos.pl sampleSec
  pdflatex sampleSec
  pdflatex sampleSec
  ```

  If you don't want to use `makeglos.pl`, you will need to do

  ```
  makeindex -s sampleSec.ist -t sampleSec.glg -o sampleSec.gls sampleSec.glo
  ```

- `sampleNtn.tex` — This has a glossary and defines a new glossary type called `notation`. The glossary has associated page numbers, but the new glossary type doesn't. The two glossaries have different styles. You will need to do:

  ```
  latex sampleNtn
  makeglos.pl sampleNtn
  latex sampleNtn
  latex sampleNtn
  ```

  If you don't want to use `makeglos.pl`, you will need to do

  ```
  makeindex -s sampleNtn.ist -t sampleNtn.glg -o sampleNtn.gls sampleNtn.glo
  makeindex -s sampleNtn.ist -t sampleNtn.nlg -o sampleNtn.ntn sampleNtn.not
  ```

- `sampleNtn2.tex` —This is similar to `sampleNtn.tex`, but uses `\storeglosentry`.

- `sampleEq.tex` — This has a glossary where the numbers in the glossary refer to the equation number rather than the page number (achieved with the package option `number=equation`). The `\entryname`, `\descriptionname`, `\glossaryname` and `\glspageheader` are all redefined to customize the glossary. You will need to do:

  ```
  latex sampleEq
  makeglos.pl sampleEq
  latex sampleEq
  ```

  If you don't want to use `makeglos.pl`, you will need to do

  ```
  makeindex -s sampleEq.ist -t sampleEq.glg -o sampleEq.gls sampleEq.glo
  ```

- `sampleEqPg.tex` — This is a modified version of `sampleEq.tex`. This example has one glossary, where some of the entry numbers refer to the corresponding page number, and some of the entry numbers refer to the corresponding equation number. You will need to do:

  ```
  latex sampleEqPg
  makeglos.pl sampleEqPg
  latex sampleEqPg
  ```

If you don't want to use makeglos.pl, you will need to do

```
makeindex -s sampleEqPg.ist -t sampleEqPg.glg -o sampleEqPg.gls sampleEqPg.glo
```

- sampleAcr.tex — This has a glossary containing acronyms. It uses the style altlist as this is better suited to glossaries with long names. It also uses the hyperref package, so the page numbers in the glossary will automatically be hyperlinks, and the acronyms within the text will have hyperlinks to their corresponding entry in the glossary. You will need to do:

  ```
  pdflatex sampleAcr
  makeglos.pl sampleAcr
  pdflatex sampleAcr
  pdflatex sampleAcr
  ```

  If you don't want to use makeglos.pl, you will need to do

  ```
  makeindex -s sampleAcr.ist -t sampleAcr.glg -o sampleAcr.gls sampleAcr.glo
  ```

- sample.tex — This has a glossary entry with two different definitions of the same name. If you just use makeindex, the two entries will be treated separately, however, if you want them concatenated, you can use makeglos.pl with the -m switch. You will need to do:

  ```
  pdflatex sample
  makeglos.pl -m sample
  pdflatex sample
  pdflatex sample
  ```

  (Depending on the configuration of your system, you may have to do perl makeglos.pl instead of just makeglos.pl)
  If you don't want to use makeglos.pl, you will need to do

  ```
  makeindex -s sample.ist -t sample.glg -o sample.gls sample.glo
  ```

  however, the entries with the same name but multiple descriptions will not be merged. You will also have to given them different sort keys otherwise you will get duplicate hyper targets.

- sample4col.tex—This illustrates how to modify the glossary style so that it has 4 columns. You will need to do:

  ```
  latex sample4col
  makeglos.pl sample4col
  latex sample4col
  ```

11 LaTeX2HTML Style File

A LaTeX2HTML Perl script, glossary.perl, is provided with this package for those wishing to use the glossary package with the LaTeX2HTML translator. The file glossary.perl should be extracted along with glossary.sty when you run the installation script (glossary.ins) through LaTeX.

11.1 Limitations

- The only package options supported are: `style=altlist`, `hyper=true`, `toc=true`, `acronym=true` and `acronym=false`.
- If you have more than one glossary type, the secondary glossaries will occur in the same segment as the primary glossary if you use the command `\print`⟨*name*⟩ instead of `\printglossary[`⟨*name*⟩`]`, where ⟨*name*⟩ is the name of the glossary type.
- The command `\setglossary` must be placed in the preamble to have an effect.
- The `\storeglosentry` commands must be in the document environment to have an effect. (They don't seem to work in the preamble, I don't know why.)
- If you place a `\glossary` command inside an environment not translated by LaTeX2HTML (for example, inside a mathematics environment), it will not be entered into the glossary.
- The combinations `""`, `"|`, `"!` and `"@` will be correctly translated, unless they occur within a maths environment. This is because the maths environment is translated before being passed to `\glossary`. You can overcome this by doing, e.g.:

  ```
  \begin{latexonly}
  \glossary{name=$"|\mathcal{S}"|$,description=cardinality of set
  $\mathcal{S}$,sort=cardinality}
  \end{latexonly}
  \begin{htmlonly}
  \glossary{name=$|\mathcal{S}|$,description=cardinality of set
  $\mathcal{S}$,sort=cardinality}
  \end{htmlonly}
  ```

  Alternative, you can use `\mid` instead:

  ```
  \glossary{name=$\mid\mathcal{S}\mid$,description=cardinality of
  set $\mathcal{S}$,sort=cardinality}
  ```

- Glossary items with the same names but different definitions will not be merged.
- The configuration file `glossary.cfg` is ignored.

12 Troubleshooting

This is a list of common problems, for a more up-to-date FAQ, see `http://theoval.cmp.uea.ac.uk/~nlct/packages/faq/`.

1. My glossary hasn't appeared.

   Check the following:

   - Have you included the command `\makeglossary` in the preamble?

- Have you put the command `\printglossary` where you want the glossary to appear?
- Have you used `makeglos.pl` or `makeindex`, and if you did, did it successfully create the `gls` file? (Check the transcript `glg` file.)
  - If you used `makeindex` directly, did you specify the `ist` file created by `\makeglossary`, and did you remember to specify the output file with the extension `gls`?
  - When `makeindex` scans the `ist` file, it should generate the message:

    ```
    9 attributes redefined, 0 ignored
    ```

    If you have a number other than 0 ignored, then there is something wrong with the `ist` file. Some packages can cause problems with the creation of this file, see item 16 below.
- Have you remembered to LaTeX your document again after using `makeglos.pl` or `makeindex`?
- Have you used `\glossary` or `\xglossary`?
- If you have used `\storeglosentry`, have you also used `\useglosentry`, `\useGlosentry` or `\gls`?

If you have defined a new glossary type, have you checked all the analogous commands to the above?

2. My list of acronyms hasn't appeared.

   Have you used the `acronym=true` package option? If no, check the answers to the previous item, if yes, make sure you have used `\makeacronym` and `\printacronym`. Have you used any of the acronyms you have defined? Remember that `\acrsh`, `\acrln`, \⟨*acr-name*⟩`short` and \⟨*acr-name*⟩`long` don't generate entries in the list of acronyms, where \⟨*acr-name*⟩ is the name of an acronym command.

3. My acronym has been expanded twice.

   By default, if any of your acronym commands occur within a group (this includes environments which form implicit grouping) the effect will be local to that group. You can either unset the acronym outside the group, or use the `global` package option.

4. I get an error when using the command `\saveglosentry`.

   Don't use this command it's obsolete, use `\storeglosentry` instead.

5. One of more of my glossary entries hasn't appeared.

   Check the following

   - If you defined the entry using `\storeglosentry` have you used either `\useglosentry`, `\useGlosentry` or `\gls`?
   - Have you remembered to `\protect` commands such as `\mathcal` within `\storeglosentry`?
   - Have you used the characters `@ ! | "`? If so, have you preceded them with a double quote character?

Check the `makeindex` log file to see if there are any error messages.

6. My glossary has duplicate entries on separate lines.

   LaTeX treats multiple spaces equivalent to a single space, but `makeindex` takes spaces into account when determining whether two entries are identical. For example:

   ```
   \glossary{name=Identity␣matrix,
   description=diagonal␣matrix␣with␣1's␣along␣the␣diagonal}
   ```

   and

   ```
   \glossary{name=Identity␣␣matrix,
   description=diagonal␣matrix␣with␣1's␣along␣the␣diagonal}
   ```

   will be treated as different entries by `makeindex`, because the first has only one space between 'Identity' and 'matrix' and the second has two. The easiest way to ensure consistency is to use `\storeglosentry` together with `\useglosentry`, `\useGlosentry` or `\gls`.

7. I had an error, fixed it, but I keep getting the same error message.

   Suppose you've made an error in the `\glossary` command. For example:

   ```
   \glossary{name=Java,description=A programming language,format=texbf}
   ```

   In this case `textbf` has been mis-spelt. This error will be copied to the `glo` file, which in turn will be copied to the `gls` file by `makeindex`. A subsequent run of LaTeX will read this error in. If you fix the error in your main document, the error will still be read in from the `gls` file. The best thing to do is to delete the `gls` file, and try again.

8. My glossary has ended up wider than my page.

   This may occur if you have long entry names, and you are using either the `style=long` or `style=super` options. The width of the description column is proportional to the line width (in fact, it's `0.6\linewidth`) but the first column is as wide as the widest entry name. You can either redefine `\glossaryalignment` to change the column specifications, or use one of the list-type styles.

9. The page numbers in my glossary don't match up with the actual page numbers where the entry was defined.

   You may need to LaTeX your document again (just as you have to do with `\tableofcontents`, `\listoffigures` etc).

10. I'm getting a `keyval` error.

    The glossary package uses the keyval package to extract the information from ⟨*key*⟩=⟨*value*⟩ comma separated lists. You need to make sure the syntax is correct. If your ⟨*value*⟩ contains a comma, you will need to enclose ⟨*value*⟩ in curly braces. See the keyval documentation for further information[7].

[7]This should be in the directory `texmf/doc/latex/graphics/`

11. I've used the `hyper` option, but nothing happens when I click on the numbers in the glossary.

    Check the following:

    (a) Have you remembered to use PDFLaTeX instead of LaTeX, or used a driver that understands hyperlinks?

    (b) Have you remembered to use the hyperref or html package?

    (c) Have you remembered to use a formatting command which uses `\hyperlink`? (E.g. using `hyperbf` instead of `textbf`)? Remember to check the `format` key in your `\glossary` commands, and the `glsnumformat` key in the `\setglossary` command.

    (d) What application are you using to view the PDF file? Ghostview can display a PDF file, but ignores the links. If you are using Windows, try using Adobe's Acrobat Reader, or if you are using UNIX or Linux, try using `xpdf` or `acroread`.

12. The glossary package conflicts with the datetime package.

    This has been fixed in version 2.01.

13. I get an error when using certain commands, such as `\cite` or `~` in `\newacronym`.

    This has been fixed in version 2.1.

14. I get the following error:

    ```
    ! Package array Error: Illegal pream-token (\glossaryalignment): 'c' used.
    ```

    The glossary package used to conflict with the array package. This was fixed in version 2.1. As from version 2.3, it doesn't matter whether you load the glossary package before or after the array package.

15. I get the following error:

    ```
    Use of \@chapter doesn't match its definition
    ```

    or

    ```
    ! Argument of \@sect has an extra }
    ```

    If you want to use an acronym command in a moving argument (such as a chapter heading) you need to `\protect` it first. Note that if you do put an acronym in a chapter etc heading, it will be expanded for the first time in the table of contents, not in the chapter heading. The best way to get around this is to use the optional argument, e.g.

    ```
    \chapter[Introduction to Kernel Support Vector Machines]{Introduction
    to \protect\KSVM}
    ```

    You will also need to do this if you are using bookmarks in a PDF document.

    Alternatively, you can do:

    ```
    \resetacronym{KSVM}
    \chapter{Introduction to \protect\KSVM}
    ```

or if you are using PDFLaTeX:

```
\resetacronym{KSVM}
\chapter{Introduction to \texorpdfstring{\protect\KSVM}{KSVM}}
```

16. The glossary package conflicts with ngerman.

    This problem is caused by the fact that ngerman redefines the effect of the double quote character, but this character is used in the creation of the `ist` `makeindex` style file. Try one of the following methods:

    (a) Include the ngerman package after the `\makeglossary` command:

    ```
    \usepackage{glossary}
    \makeglossary
    \usepackage{ngerman}
    ```

    (b) First omit the ngerman package and include `\makeglossary` then LaTeX your document. This will create the `ist` file. Then include the ngerman package, and insert `\noist` before the `\makeglossary` command, this will prevent further attempts to generate the `ist` file.

    ```
    \usepackage{ngerman}
    \usepackage{glossary}
    \noist\makeglossary
    ```

    (c) Use `\noist`, as above, and create the `ist` file in an ordinary text editor. The file should contain the following lines:

    ```
    keyword "\\glossaryentry"
    preamble "\\begin{theglossary}"
    postamble "\n\\end{theglossary}\n"
    group_skip "\\gloskip "
    item_0 "\n\\gloitem "
    delim_0 "\n\\glodelim "
    page_compositor "-"
    delim_n "\\delimN "
    delim_r "\\delimR "
    ```

    It is possible that there may be other packages which will also cause a problem, if so, try any of the above.

17. `makeglos.pl` gives the following error message:

    ```
    unable to extract name from glossary item:
    ```

    You are using an old version of `makeglos.pl` with a new version of the glossary package. You will need to update your version `makeglos.pl`.

Let me know if you encounter any other problems or if you have any comments regarding this package.

13 Obsolete Commands

The commands described in this section are now obsolete, but are currently still provided for backwards compatibility. Their use is deprecated.

`\saveglosentry` `\saveglosentry{⟨name⟩}{⟨description⟩}`

This command has now been replaced by `\storeglosentry`.

`\glsprimaryfmt` The command `\glsprimaryfmt` has now been removed.

The package option `hyperacronym` is now superseded by the package option `hyper`. This option was implemented prior to the introduction of the command `\xglossary`. Since the acronyms now use `\xglossary`, there is no difference between the `hyperacronym` and `hyper` options. This option has a boolean value:

`true` Make acronyms link to their corresponding entry in the glossary

`false` Acronyms don't have a hyperlink.

If the hyperref package has been loaded prior to loading `glossary.sty` or if `hyper=true` is set, `hyperacronym=true` otherwise `hyperacronym=false`.

14 Contact Details

Dr Nicola Talbot
School of Computing Sciences
University of East Anglia
Norwich. Norfolk
NR4 7TJ. United Kingdom.
http://theoval.cmp.uea.ac.uk/~nlct/

15 Acknowledgements

I would like to thank all the many people who have made suggestions and pointed out bugs.

16 The Code

⟨∗glossary.sty⟩

16.1 Package Definition

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossary}[2006/07/20 2.4 (NLCT)]
```

Load packages needed by glossary.sty:

```
\RequirePackage{ifthen}
\RequirePackage{keyval}
```

The package options are in the form of a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs. First need to set up the keys.

The `style` key. This may be one of `list`, `altlist` (use description environment), `super` (use supertabular environment) or `long` (use longtable environment).

```
\define@key{gloss}
{style}
```

```
{\ifthenelse{\equal{#1}{list} \or \equal{#1}{altlist}
\or \equal{#1}{super} \or \equal{#1}{long}}
{\def\gls@style{#1}}
{\PackageError{glossary}
{Unknown glossary style '#1'}
{Available styles are: list, altlist, super and long}}}
```

The `header` key. This can either be `none` or `plain`. Should only be used in conjunction with `super=style` or `style=long`.

```
\define@key{gloss}
{header}[plain]{\ifthenelse{\equal{#1}{none} \or \equal{#1}{plain}}
{\def\gls@header{#1}}
{\PackageError{glossary}
{Unknown glossary style '#1'}
{Available styles are: none and plain}}}
```

The `border` key. This can either be `none` or `plain`. Should only be used in conjunction with `style=super` or `style=long`.

```
\define@key{gloss}
{border}[plain]{\ifthenelse{\equal{#1}{none} \or \equal{#1}{plain}}
{\def\gls@border{#1}}
{\PackageError{glossary}
{Unknown glossary border '#1'}
{Available styles are: none and plain}}}
```

Number of columns (either 2 or 3). Should only be used in conjunction with `style=super` or `style=long`.

```
\newcount\gls@cols
\define@key{gloss}{cols}{\gls@cols=#1\relax
\ifthenelse{\gls@cols<2 \or \gls@cols>3}
{\PackageError{glossary}
{invalid number of columns}
{The cols option can only be 2 or 3}}
{}}
```

The number key may either be `none` or the name of a counter.

```
\define@key{gloss}
{number}
{\ifthenelse{\equal{#1}{none}}
{\def\gls@glossary@number{#1}}
{\@ifundefined{c@#1}{
\PackageError{glossary}
{Unknown glossary number style '#1'}
{You may either specify "none" or the name of a counter,
e.g. "section"}\def\gls@glossary@number{page}}{\def\gls@glossary@number{#1}}}}
```

The `toc` key. If set, adds the glossary to the table of contents

```
\newif\ifgls@toc
\define@key{gloss}{toc}[true]{\ifthenelse{\equal{#1}{true}
\or \equal{#1}{false}}
{\csname gls@toc#1\endcsname}
{\PackageError{glossary}{Glossary option 'toc' is boolean}
{The value of 'toc' can only be set to 'true' or 'false'}}}
```

The `hypertoc` key. Like `toc`, but puts the anchor before the section heading. Should only be used if the hyperref package is used (because it uses `\phantomsection`).

```
\newif\ifgls@hypertoc
```

```
\define@key{gloss}{hypertoc}[true]{%
\ifthenelse{\equal{#1}{true} \or \equal{#1}{false}}
{\csname gls@hypertoc#1\endcsname}
{\PackageError{glossary}{Glossary option 'hypertoc' is boolean}
{The value of 'hypertoc' can only be set to 'true' or 'false'}}}
```

The `section` key. This will put the glossary in an unnumbered section, even if chapters are defined.

```
\newif\ifgls@section
\define@key{gloss}{section}[true]{%
\ifthenelse{\equal{#1}{true} \or \equal{#1}{false}}
{\csname gls@section#1\endcsname}
{\PackageError{glossary}{Glossary option 'section' is boolean}
{The value of 'section' can only be set to 'true' or 'false'}}}
\gls@sectionfalse
```

Enable hyperlinks. If hyperref or html packages loaded, `hyper=true` is the default.

```
\newif\ifglshyper
\newif\ifglshyperacronym
\define@key{gloss}{hyper}[true]{%
\ifthenelse{\equal{#1}{true} \or \equal{#1}{false}}
{\csname glshyper#1\endcsname\glshyperacronymtrue}
{\PackageError{glossary}{Glossary option 'hyper' is boolean}
{The value of 'hyper' can only be set to 'true' or 'false'}}}
```

Enable hyperlinks for acronyms. Deprecated: use `hyper` instead.

```
\define@key{gloss}{hyperacronym}[true]{%
\ifthenelse{\equal{#1}{true} \or \equal{#1}{false}}
{\csname glshyperacronym#1\endcsname}
{\PackageError{glossary}{Glossary option 'hyperacronym' is boolean}
{The value of 'hyperacronym' can only be set to 'true' or 'false'}}}
```

The `acronym` key. If set, the acronyms will be separate from main glossary entries. Remember to use `\makeacronym` and `\printacronym` if `true`.

```
\newif\ifglsacronym
\define@key{gloss}{acronym}[true]{%
\ifthenelse{\equal{#1}{true} \or \equal{#1}{false}}
{\setboolean{glsacronym}{#1}}{%
\PackageError{glossary}{Glossary option 'acronym' is boolean}{The
value of 'acronym' can only be set to 'true' or 'false'}}}
```

The `global` key. If not set, any acronyms expanded in a group will be treated as unused once outside of the group. Set `global=true` to prevent this.

```
\newif\ifglsglobal
\define@key{gloss}{global}[true]{\ifthenelse{\equal{#1}{true}\or
\equal{#1}{false}}{\setboolean{glsglobal}{#1}}{%
\PackageError{glossary}{Glossary option 'global' is boolean}{The
value of 'global' can only be set to 'true' or 'false'}}}
```

Set up defaults

```
\def\gls@style{long}
\def\gls@header{none}
\def\gls@border{none}
\def\gls@glossary@number{page}
\gls@cols=2\relax
\gls@tocfalse
```

If `\hyperpage` is defined, then assume hyperlinks required

```
\@ifundefined{hyperpage}{\glshyperfalse\glshyperacronymfalse}{%
\glshypertrue\glshyperacronymtrue}
```

If `\hypertarget` defined, then `\glosslabel` will make a target (`#1`) and `\glossref` will make a hyperlink (to `#1`). Otherwise will simply print the second argument.

```
\@ifundefined{hypertarget}{
% no hyperlinks
\newcommand{\glosslabel}[2]{#2}%
\newcommand{\glossref}[2]{#2}%
}{%
\newcommand{\glosslabel}[2]{\hypertarget{#1}{#2}}%
\newcommand{\glossref}[2]{\hyperlink{#1}{#2}}
}
```

If the xspace package has been loaded, use `\xspace` in acronyms.

```
\@ifundefined{xspace}{%
\let\glsxspace\relax}{%
\let\glsxspace\xspace}
```

Set `\glossaryalignment` to `\relax` before loading configuration file.

```
\let\glossaryalignment\relax
```

Load configuation file if it exists

```
\newcommand{\glossarypackageoptions}[1]{\setkeys{gloss}{#1}}
\InputIfFileExists{glossary.cfg}{%
\typeout{Glossary configuration file loaded}}{%
\typeout{No configuration file glossary.cfg found}}
\renewcommand{\glossarypackageoptions}[1]{%
\PackageError{glossary}{Command \string\glossarypackageoptions
^^Jcan only be used in configuration file}{}}
```

Set up the options so that they are treated as a ⟨*key*⟩=⟨*value*⟩ list.

```
\DeclareOption*{\edef\@pkg@ptions{\noexpand
\setkeys{gloss}{\CurrentOption}}
\ifthenelse{\equal{\CurrentOption}{}}{}{\@pkg@ptions}}
```

Process options

```
\ProcessOptions
```

Check to make sure that the options don't conflict.

```
\ifthenelse{\(\equal{\gls@style}{list} \or
\equal{\gls@style}{altlist}\) \and
\(\not\equal{\gls@header}{none} \or \not\equal{\gls@border}{none}
\or \gls@cols=3\)}
{\PackageError{glossary}{You can't have option 'style=list' or
'style=altlist' in combination with any of the other style
options}{The 'list' and 'altlist' options don't have a header,
border or number of columns option.}}
{}
```

Can't have both toc and hypertoc. Make it a warning rather than an error.

```
\ifthenelse{\boolean{gls@hypertoc} \and \boolean{gls@toc}}{%
\PackageWarning{glossary}{Can't have both 'toc' and
'hypertoc', ignoring 'toc' option}
\ifgls@hypertoc\gls@tocfalse\fi}{}
```

16.2 Redefining `\glossary` format

The glossary is going to be redefined so that it accepts ⟨*key*⟩=⟨*value*⟩ information, so need to define the keys (see keyval documentation for further details on how to do this.) Added `\@onelevel@sanitize` at the recommendation of Dan Luecking and Ulrich Diez.

```
\define@key{wrgloss}{name}{%
\def\@glo@n@me{#1}%
\@onelevel@sanitize\@glo@n@me%
\global\let\@glo@n@me\@glo@n@me}
\define@key{wrgloss}{description}{%
\def\@descr{#1}%
\@onelevel@sanitize\@descr}
\define@key{wrgloss}{sort}{%
\def\@s@rt{#1}%
\@onelevel@sanitize\@s@rt
\global\let\@s@rt\@s@rt}
\define@key{wrgloss}{format}{\def\@f@rm@t{#1}}
\define@key{wrgloss}{number}{\def\@glo@num{#1}}
```

Redefine `\@wrglossary` so that it separates out the entry name and entry description. This was rewritten in version 2.4. It is now used for both the main glossary, and user-defined glossaries. The command `\@@wrglossary` is called at the end of `\@wrglossary`, by default this does nothing, but some commands temporarily redefine it.

```
\newcommand{\@@wrglossary}{}
```

The label for each entry is usually made up of the glossary prefix followed by the sort value, this can be over-ridden by redefining `\@glo@l@bel`. (This is done if the optional argument to `\glossary` is used.) By default this does nothing.

```
\newcommand{\@glo@l@bel}{}
```

Define the prefix for the principle glossary. (Added to version 2.4.)

```
\newcommand{\@gls@glossary@type}{glo}
```

The optional first argument was added in version 2.4. This is the name of the glossary type.

```
\renewcommand{\@wrglossary}[2][glossary]{\relax
\gdef\@glo@n@me{}\def\@descr{}\def\@s@rt{}\def\@f@rm@t{}%
\edef\@glo@num{\csname gls@#1@number\endcsname}\relax
\xdef\@pr@fix{\csname @gls@#1@type\endcsname}%
 \setkeys{wrgloss}{#2}\relax
\ifthenelse{\equal{\@glo@num}{none}}{\def\@@glo@num{\thepage}}{%
\@ifundefined{c@\@glo@num}{\PackageError{glossary}{%
Not such counter '\@glo@num'}{The value of the 'number' key
must be the name of a counter or the word "none"}%
\def\@@glo@num{\thepage}}{%
\edef\@@glo@num{\csname the\@glo@num\endcsname}}}%
\ifthenelse{\equal{\@s@rt}{}}{\gdef\@s@rt{\@glo@n@me}}{}%
\ifthenelse{\equal{\@glo@l@bel}{}}{%
\gdef\@glo@l@bel{\@pr@fix:\@s@rt}}{}%
```

User has not specified a format, so use default

```
\ifthenelse{\equal{\@f@rm@t}{}}
{\expandafter\protected@write\csname @#1file\endcsname{}%
{\string\glossaryentry{\@s@rt @{%
\string\glosslabel{\@glo@l@bel}{\@glo@n@me}}\@descr
\string\relax|glsnumformat}{\@@glo@num}}}
```

User has specified a format. If it is one of the `\hyper`$\langle xx \rangle$ types, append the required counter. This is needed if the glossary contains a mixture of counters used (as in `sampleEqPg.tex`).

```
{\ifthenelse{\equal{\@f@rm@t}{hyperrm} \or
\equal{\@f@rm@t}{hypersf} \or \equal{\@f@rm@t}{hypertt}
\or \equal{\@f@rm@t}{hypermd} \or \equal{\@f@rm@t}{hyperbf}
\or \equal{\@f@rm@t}{hyperit} \or \equal{\@f@rm@t}{hyperem}
\or \equal{\@f@rm@t}{hypersl} \or \equal{\@f@rm@t}{hyperup}
\or \equal{\@f@rm@t}{hypersc}}
{\expandafter\protected@write\csname @#1file\endcsname{}%
    {\string\glossaryentry{\@s@rt @{%
     \string\glosslabel{\@glo@l@bel}{\@glo@n@me}}\@descr
     \string\relax|\@f@rm@t[\@glo@num]}{\@@glo@num}}}
{\expandafter\protected@write\csname @#1file\endcsname{}%
    {\string\glossaryentry{\@s@rt @{%
     \string\glosslabel{\@glo@l@bel}{\@glo@n@me}}\@descr
     \string\relax|\@f@rm@t}{\@@glo@num}}}}\relax
 \endgroup\@esphack
\@@wrglossary
}
```

Command to extract name key from glossary entry. This shouldn't be sanitized, so define a new key for this

```
\define@key{wrnsgloss}{name}{\def\@glo@n@me{#1}}
\define@key{wrnsgloss}{description}{\def\@descr{#1}}
\define@key{wrnsgloss}{sort}{\def\@s@rt{#1}}
\define@key{wrnsgloss}{format}{\def\@f@rm@t{#1}}
\define@key{wrnsgloss}{number}{\def\@glo@num{#1}}
```

Extract name from key-value list. Name stored in `\@glo@n@me`.

```
\newcommand{\@gls@getn@me}[1]{%
\def\@glo@n@me{}\setkeys{wrnsgloss}{#1}%
}
```

Command to extract description key from glossary entry.

```
\newcommand{\@gls@getdescr}[1]{%
\@bsphack\begingroup
\def\@descr{}%
\setkeys{wrgloss}{#1}%
\global\let\@glo@desc\@descr
\endgroup\@esphack
}
```

Now define `\xglossary` so you can have a hyperlink that takes you to the entry in the glossary

```
\newcommand{\xglossary}{\renewcommand{\@@wrglossary}[1]{%
\glossref{\@glo@l@bel}{##1}\renewcommand{\@@wrglossary}{}}%
\glossary}
```

16.3 Storing Glossary Entries

Provide a means to store glossary information to save typing and ensure consistency (new to v2.17).

Store label in list (new to version 2.36) so that all entries can be added to the glossary with a single command.

```
\newcommand*{\@glo@label@list}{}
\toksdef\gls@ta=0 \toksdef\gls@tb=2
\newcommand{\@glo@label@addtolist}[1]{%
\gls@ta={{#1}}\gls@tb=\expandafter{\@glo@label@list}%
\xdef\@glo@label@list{\the\gls@ta,\the\gls@tb}}
```

First define command to store details (don't allow a label consisting solely of a `*` as this represents all entries when passed to `\useglosentry`.)

```
\newcommand*{\storeglosentry}[3][glossary]{%
\ifthenelse{\equal{#2}{*}}{%
\PackageError{glossary}{Glossary label '*' invalid}{You can't have
a glossary entry with a * as the label}}{%
\@ifundefined{glo@#2@entry}{%
\@glo@label@addtolist{#2}%
\expandafter\def\csname glo@#2@type\endcsname{#1}%
\expandafter\def\csname glo@#2@entry\endcsname{#3}%
\@gls@getn@me{#3}%
\expandafter\protected@edef\csname glo@#2@name\endcsname{\@glo@n@me}%
}{%
\PackageError{glossary}{Glossary entry '#2' already
defined}{There already exists a glossary entry with the label '#2'}}}%
}
```

This command will not produce text in the document, but will produce the relevant glossary entry.

```
\providecommand{\useglosentry}[2][\relax]{%
\ifthenelse{\equal{#2}{*}}{\@for\@glolab:=\@glo@label@list\do{%
\ifthenelse{\equal{\@glolab}{}}{}{\useglosentry[#1]{\@glolab}}}}{%
\@ifundefined{glo@#2@type}{%
\PackageError{glossary}{Glossary entry '#2' undefined}{You need
to define the entry using \string\storeglosentry\space before
using it.}}{{%
\edef\@glostype{\csname glo@#2@type\endcsname}%
\@glo@tb=\expandafter\expandafter\expandafter
{\csname glo@#2@entry\endcsname}%
\ifx#1\relax
\edef\@glo@cmd{\expandafter\noexpand
\csname\@glostype\endcsname{\the\@glo@tb}}%
\else
\edef\@glo@cmd{\expandafter\noexpand
\csname\@glostype\endcsname{\the\@glo@tb,#1}}%
\fi
\@glo@cmd
}}}}
```

This command will produce the specified text in the document (with a hyperlink if enabled), and will produce the relevant glossary entry.

```
\providecommand{\useGlosentry}[3][\relax]{%
\@ifundefined{glo@#2@type}{%
\PackageError{glossary}{Glossary entry '#2' undefined}{You need
to define the entry using \string\storeglosentry\space before
using it.}}{{%
\edef\@glostype{x\csname glo@#2@type\endcsname}%
\@glo@tb=\expandafter\expandafter\expandafter
{\csname glo@#2@entry\endcsname}%
\ifx#1\relax
\edef\@glo@cmd{\expandafter\noexpand
\csname\@glostype\endcsname{\the\@glo@tb}}%
\else
\edef\@glo@cmd{\expandafter\noexpand
\csname\@glostype\endcsname{\the\@glo@tb,#1}}%
\fi
\@glo@cmd{#3}%
}}}
```

As above, but the text displayed in the document is constructed from the `name` key.

```
\newcommand{\gls}[2][\relax]{%
\useGlosentry[#1]{#2}{%
\csname glo@#2@name\endcsname}}
```

This command was defined in earlier verions, but doesn't work very well, currently retained for backwards compatibility, but may well be removed at a later date.

```
\providecommand{\saveglosentry}[3][glossary]{%
\PackageWarning{glossary}{\string\saveglosentry\space is obsolete,
please use \string\storeglosentry\space instead}%
\expandafter\def\csname glo@#2@type\endcsname{#1}%
\expandafter\def\csname glo@#2@entry\endcsname{%
name={#2},description={#3}}}
```

Set up default number formats, dependent on the package `number` option. Define default page compositor. Any redefinition of the page compositor will need to come before the `.ist` file is written. The other commands can be redefined at any point before `\printglossary`.

Define a command to set up the glossary counter. The optional argument specifies the glossary type (defaults to the main glossary). The mandatory command is the name of the counter, or `none`.

```
\newcommand*{\@gls@setnumbering}[2][glossary]{%
```

If no numbering (`number=none`):

```
\ifthenelse{\equal{#2}{none}}{%
\def\pagecompositor{-}
\expandafter\def\csname @#1@delimN\endcsname{}
\expandafter\def\csname @#1@delimR\endcsname{}
\expandafter\def\csname glsX#1Xnumformat\endcsname##1{}}{%
```

If `number=page`, set the page compositor to `-` (dash) otherwise set it to `.` (dot).

```
\ifthenelse{\equal{#2}{page}}{%
\def\pagecompositor{-}}{%
\def\pagecompositor{.}}
```

Set up delimiters and formats

```
\expandafter\def\csname @#1@delimN\endcsname{, }
\expandafter\def\csname @#1@delimR\endcsname{--}
\ifglshyper
```

```
\expandafter\def\csname glsX#1Xnumformat\endcsname##1{%
\hyperrm[#2]{##1}}%
\else
\expandafter\def\csname glsX#1Xnumformat\endcsname##1{##1}\fi
}
```

End of `\@gls@setnumbering` definition:

```
}
```

Now call it to set up current numbering:

```
\@gls@setnumbering{\gls@glossary@number}
```

Provide a means of changing the page number format for a given glossary type.

```
\newcommand{\glsnumformat}[1]{%
\@ifundefined{\@glostype}{\def\@glostype{glossary}}{}%
\@ifundefined{glsX\@glostype Xnumformat}{%
\PackageError{glossary}{Glossary type '\@glostype' undefined}{}}{%
\csname glsX\@glostype Xnumformat\endcsname{#1}}}
```

Set the default glossary type

```
\def\@glostype{glossary}
```

Make the delimiters etc depend on the glossary type. `\@glostype` should be set to the appropriate glossary type before using any of these commands.

```
\newcommand{\delimN}{\csname @\@glostype @delimN\endcsname}
\newcommand{\delimR}{\csname @\@glostype @delimR\endcsname}
\newcommand{\gloitem}{\csname @\@glostype @gloitem\endcsname}
\newcommand{\gloskip}{\csname @\@glostype @gloskip\endcsname}
\newcommand{\delimT}{\glsafternum
\csname @\@glostype @delimT\endcsname}
\newcommand{\glodelim}{\csname @\@glostype @glodelim\endcsname
\glsbeforenum}
```

Add facility to insert text between groups. By default these do nothing.

```
\newcommand{\glogroupSymbols}{}
\newcommand{\glogroupNumbers}{}
\newcommand{\glogroupA}{}
\newcommand{\glogroupB}{}
\newcommand{\glogroupC}{}
\newcommand{\glogroupD}{}
\newcommand{\glogroupE}{}
\newcommand{\glogroupF}{}
\newcommand{\glogroupG}{}
\newcommand{\glogroupH}{}
\newcommand{\glogroupI}{}
\newcommand{\glogroupJ}{}
\newcommand{\glogroupK}{}
\newcommand{\glogroupL}{}
\newcommand{\glogroupM}{}
\newcommand{\glogroupN}{}
\newcommand{\glogroupO}{}
\newcommand{\glogroupP}{}
\newcommand{\glogroupQ}{}
\newcommand{\glogroupR}{}
\newcommand{\glogroupS}{}
\newcommand{\glogroupT}{}
```

```
\newcommand{\glogroupU}{}
\newcommand{\glogroupV}{}
\newcommand{\glogroupW}{}
\newcommand{\glogroupX}{}
\newcommand{\glogroupY}{}
\newcommand{\glogroupZ}{}
```

Allow user to change number format for different glossary types.

```
\define@key{glossnum}{glsnumformat}{\def\@glsnumformat{#1}}
\define@key{glossnum}{type}{\def\@glsnumtype{#1}}
\define@key{glossnum}{delimN}{\def\@delimN{#1}}
\define@key{glossnum}{delimR}{\def\@delimR{#1}}
\define@key{glossnum}{delimT}{\def\@delimT{#1}}
\define@key{glossnum}{gloskip}{\def\@gloskip{#1}}
\define@key{glossnum}{glodelim}{\def\@glodelim{#1}}
```

Define a command that will ignore its argument. This is used when suppressing the page numbers.

```
\providecommand{\ignore}[1]{}
```

Define command that allows the user to modify the style for a given glossary type.

```
\newcommand{\setglossary}[1]{%
\def\@glsnumformat{}%
\def\@glsnumtype{glossary}%
\def\@delimN{@dontchange@}%
\def\@delimR{@dontchange@}%
\def\@delimT{@dontchange@}%
\def\@gloskip{@dontchange@}%
\def\@glodelim{@dontchange@}%
\setkeys{glossnum}{#1}\relax
\@ifundefined{print\@glsnumtype}{%
\PackageError{glossary}{Invalid glossary type '\@glsnumtype'}{%
Glossary type '\@glsnumtype' has not been defined}
}{%
\ifthenelse{\equal{\@glsnumformat}{}}{}{%
\expandafter\xdef\csname glsX\@glsnumtype Xnumformat\endcsname{%
\noexpand\csname\@glsnumformat\noexpand\endcsname}%
\ifthenelse{\equal{\@glsnumformat}{ignore}}{%
\expandafter\xdef\csname @\@glsnumtype @delimN\endcsname{}%
\expandafter\xdef\csname @\@glsnumtype @delimR\endcsname{}%
}{}%
}%
%
\ifthenelse{\equal{\@delimN}{@dontchange@}}{}{%
\expandafter\xdef\csname @\@glsnumtype @delimN\endcsname{%
\@delimN}}%
%
\ifthenelse{\equal{\@delimR}{@dontchange@}}{}{%
\expandafter\xdef\csname @\@glsnumtype @delimR\endcsname{%
\@delimR}}%
%
\ifthenelse{\equal{\@delimT}{@dontchange@}}{}{%
\expandafter\xdef\csname @\@glsnumtype @delimT\endcsname{%
\@delimT}}%
%
\ifthenelse{\equal{\@gloskip}{@dontchange@}}{}{%
\expandafter\xdef\csname @\@glsnumtype @gloskip\endcsname{%
\@gloskip}}%
%
\ifthenelse{\equal{\@glodelim}{@dontchange@}}{}{%
\expandafter\xdef\csname @\@glsnumtype @glodelim\endcsname{%
\@glodelim}%
}%
}}
```

Now define the command `\printglossary` which will print the contents of the glossary file. Define the file extension for the main glossary:

```
\newcommand{\@gls@glossary@inext}{gls}
```

The optional argument is the glossary type, the default is the main glossary. This sets `\gls@number` to `\gls@#1@number` before reading in the file. This ensures that `\hyperrm` etc use the correct counter in the target name.

```
\newcommand\printglossary[1][glossary]{%
\def\@glostype{#1}%
\@ifundefined{#1name}{%
\renewcommand{\@glossaryname}{\glossaryname}}{%
\renewcommand{\@glossaryname}{\csname #1name\endcsname}}%
\@ifundefined{short#1name}{%
\renewcommand{\@shortglossaryname}{\@glossaryname}}{%
\renewcommand{\@shortglossaryname}{\csname short#1name\endcsname}}%
\expandafter\let\expandafter\gls@number\csname gls@#1@number\endcsname
\@input@{\jobname.\csname @gls@#1@inext\endcsname}}
```

Define contextual names. Changed `\newcommand` to `\providecommand` in version 2.2.

```
\providecommand{\glossaryname}{Glossary}
\newcommand{\shortglossaryname}{\glossaryname}
\newcommand{\entryname}{Notation}
\newcommand{\descriptionname}{Description}
\newcommand{\istfilename}{\jobname.ist}
\def\@glossaryname{\glossaryname}
\def\@shortglossaryname{\shortglossaryname}
```

Version 2.4 also writes ist filename to aux file. This is only used by makeglos.pl, so ignore.

```
\newcommand{\@istfilename}[1]{}
```

Define command to generate glossary title (new to version 2.24)

```
\providecommand{\glossarytitle}{%
\@ifundefined{chapter}%
```

`\chapter` not defined, use `\section*`

```
{%
\ifgls@hypertoc
```

`hypertoc` option used, so use `\phantomsection` to add anchor *before* `\section*`

```
\phantomsection
\@glosaddtoc{section}%
\section*{\@glossaryname}\relax
\else
```

`hypertoc=false`: add to toc after `\section*`

```
\section*{\@glossaryname}\relax
```

only add contentsline if toc=true

```
\ifgls@toc\@glosaddtoc{section}\fi
\fi}%
```

`\chapter` defined, but has user requested `\section` instead?

```
{%
\ifthenelse{\boolean{gls@section}}%
{%
```

user requested `\section`

```
\ifgls@hypertoc
```

User request `hypertoc=true`, so add anchor *before* `\section`:

```
\phantomsection
\@glosaddtoc{section}%
\section*{\@glossaryname}\relax
\else
```

`hypertoc=false` so add contentsline (if applicable) after `\section`

```
\section*{\@glossaryname}\relax
\ifgls@toc\@glosaddtoc{section}\fi
\fi}%
{%
```

User has not requested `\section`, so use `\chapter`

```
\ifgls@hypertoc
```

User has requested `hypertoc=true`. Chapters usually start a new page, so to ensure anchor is at the top of the correct page, issue a `\clearpage` (or `\cleardoublepage`) to place the anchor at the correct place.

```
\@ifundefined{if@twoside}{%
```

Document class doesn't support twosided documents so just do `\clearpage`

```
\clearpage}{%
\if@twoside
```

Document is two-sided If `\cleardoublepage` is defined, use that otherwise just do `\clearpage`

```
\@ifundefined{cleardoublepage}{\clearpage}{\cleardoublepage}%
\else
```

One-sided document, just do `\clearpage`

```
\clearpage
\fi}%
```

add anchor before `\chapter`

```
\phantomsection
\@glosaddtoc{chapter}%
\fi
\chapter*{\@glossaryname}\relax
```

both `hypertoc=true` and `toc=true`, so won't get toc entry twice.)

```
\ifgls@toc\@glosaddtoc{chapter}\fi}}
\markboth{\@shortglossaryname}{\@shortglossaryname}%
}
```

Now define theglossary environment. Version 2.2: check to see if defined already

```
\@ifundefined{theglossary}{%
\newenvironment{theglossary}{}{}}{%
\PackageWarning{glossary}{Redefining 'theglossary' environment}}
\renewenvironment{theglossary}{%
\glossarytitle
\glossarypreamble\@bef@reglos}{\@ftergl@s\glossarypostamble}
```

Provide a means to add text to the beginning or end of the glossary.

```
\newcommand{\glossarypreamble}{}
\newcommand{\glossarypostamble}{}
```

By default, add the short title to the table of contents.

```
\newcommand{\@glosaddtoc}[1]{%
\addcontentsline{toc}{#1}{\@shortglossaryname}
}
```

Set up switch to determine whether the item is the first item in the glossary (in the event that a special case is needed for the first item)

```
\newif\ifgloitemfirst
\newcommand{\@bef@reglos}{\global\gloitemfirsttrue\beforeglossary}
\newcommand{\@ftergl@s}{\afterglossary\global\gloitemfirstfalse}
```

Set up defaults.

```
\newcommand{\glossaryalignment}{\relax}
\newcommand{\@gls@align@glossary}{}
\newcommand{\glosstail}{%
\@ifundefined{@gls@tail@\@glostype}{%
\PackageError{glossary}{No glossary tail defined for glossary
type '\@glostype'}{}}{%
\csname @gls@tail@\@glostype\endcsname}}
\newcommand{\@gls@tail@glossary}{}
\newcommand{\afterglossary}{%
\@ifundefined{@gls@afterglos@\@glostype}{%
\PackageError{glossary}{No after glossary defined for glossary
type '\@glostype'}{}}{%
\csname @gls@afterglos@\@glostype\endcsname}}
\newcommand{\beforeglossary}{%
\@ifundefined{@gls@beforeglos@\@glostype}{%
\PackageError{glossary}{No before glossary defined for glossary
type '\@glostype'}{}}{%
\csname @gls@beforeglos@\@glostype\endcsname}}
\newcommand{\@gls@beforeglos@glossary}{}
\newcommand{\@gls@afterglos@glossary}{}
\newcommand{\@glossary@glodelim}{}
\newcommand{\@glossary@delimT}{}
\newcommand{\glsafternum}{}
\newcommand{\glsbeforenum}{}
\newcommand{\@glossary@gloskip}{}
\newcommand{\@glossary@gloitem}[1]{#1}
```

Now define what to do depending on which style has been selected. First define command to switch to `list` style:

```
\newcommand{\gls@setlist}[1][glossary]{%
\expandafter\def\csname @gls@beforeglos@#1\endcsname{%
\begin{description}}%
\expandafter\def\csname @gls@afterglos@#1\endcsname{%
\end{description}}%
\expandafter\def\csname @#1@gloskip\endcsname{\indexspace}%
\ifthenelse{\equal{\csname gls@#1@number\endcsname}{none}}{%
\expandafter\def\csname @#1@glodelim\endcsname{}}{%
\expandafter\def\csname @#1@glodelim\endcsname{, }}%
\expandafter\def\csname @#1@gloitem\endcsname##1{\item[##1]}%
\expandafter\def\csname @#1@delimT\endcsname{}
}
```

Next define command to switch to altlist style:

```
\newcommand{\gls@setaltlist}[1][glossary]{%
\expandafter\def\csname @gls@beforeglos@#1\endcsname{%
\begin{description}}%
\expandafter\def\csname @gls@afterglos@#1\endcsname{%
\end{description}}%
\expandafter\def\csname @#1@gloskip\endcsname{\indexspace}%
\expandafter\def\csname @#1@gloitem\endcsname##1{%
\item[##1]\mbox{}\nopagebreak\par\nopagebreak}%
\expandafter\def\csname @#1@glodelim\endcsname{ }%
\expandafter\def\csname @#1@delimT\endcsname{}
}
```

Now deal with the other styles. I orginally used a tabular environment, but obviously this doesn't work for a glossary longer than one page (this package started out as a simple example accompanying one of my tutorials). Nick van Foreest recommended the supertabular environment. The longtable environment also works, so have both options, and leave it to the user.

```
\ifthenelse{\equal{\gls@style}{super}}{
\IfFileExists{supertab.sty}{\RequirePackage{supertab}}
{\IfFileExists{supertabular.sty}{\RequirePackage{supertabular}}
{\PackageError{glossary}{Option "super" chosen, but can't find
"supertab" package}{If you want the "super" option, you have to have
the "supertab" package installed.}}}}
{\RequirePackage{longtable}}
```

Define new length specifying the width of the description field.

```
\newlength{\descriptionwidth}
\setlength{\descriptionwidth}{0.6\linewidth}
```

If user has defined the command `\glossaryheader`, use it otherwise use header as specified by glossary style. Added `\glossarysubheader` in version 2.4. This is provided to add a sub heading, or to add a bit of space between the header row and the table.

```
\newcommand{\@glossaryheader}{%
\@ifundefined{glossaryheader}{\csname @\@glostype @header\endcsname}
{\glossaryheader}%
\@ifundefined{glossarysubheader}{}{\glossarysubheader}%
}
```

Define command to set header style. Added `\glspageheader` in version 2.4. (Third column header)

```
\newcommand{\gls@setheader}[1][glossary]{%
\ifthenelse{\equal{\gls@header}{none}}%
{%
\ifthenelse{\equal{\gls@border}{none}}
{\expandafter\def\csname @#1@header\endcsname{}%
}{\expandafter\def\csname @#1@header\endcsname{\hline}}%
}{%
\ifnum\gls@cols=2\relax
\ifthenelse{\equal{\gls@border}{none}}
{%
\expandafter\def\csname @#1@header\endcsname{%
\bfseries\entryname & \bfseries \descriptionname\\}}%
{%
\expandafter\def\csname @#1@header\endcsname{%
\hline\bfseries\entryname & \bfseries\descriptionname
\\\hline\hline}}%
\else
\ifthenelse{\equal{\gls@border}{none}}
{%
\expandafter\def\csname @#1@header\endcsname{%
\bfseries\entryname & \bfseries \descriptionname &
\bfseries \glspageheader \\}}%
{%
\expandafter\def\csname @#1@header\endcsname{%
\hline\bfseries\entryname &\bfseries\descriptionname &
\bfseries \glspageheader \\\hline\hline}}%
\fi
}}
```

Define `\glspageheader` to do nothing, to keep it compatible with earlier versions:

```
\newcommand*{\glspageheader}{}
```

Define command to set glossary alignment and borders

```
\newcommand{\gls@setalignment}[1][glossary]{%
\ifthenelse{\equal{\gls@border}{none}}
{
\ifnum\gls@cols=2\relax
\expandafter\def\csname @gls@align@#1\endcsname{%
@{\hspace{\tabcolsep}\bfseries}lp{\descriptionwidth}}
\else
\expandafter\def\csname @gls@align@#1\endcsname{%
@{\hspace{\tabcolsep}\bfseries}lp{\descriptionwidth}l}
\fi
%
\expandafter\def\csname @gls@tail@#1\endcsname{}%
}{%
\ifnum\gls@cols=2\relax
\expandafter\def\csname @gls@align@#1\endcsname{%
|@{\hspace{\tabcolsep}\bfseries
}lp{\descriptionwidth}|}
\else
\expandafter\def\csname @gls@align@#1\endcsname{%
|@{\hspace{\tabcolsep}\bfseries
}lp{\descriptionwidth}l|}
\fi
%
\expandafter\def\csname @gls@tail@#1\endcsname{\hline}%
}%
%
\expandafter\def\csname @#1@delimT\endcsname{\\}
%
\ifnum\gls@cols=2\relax
\expandafter\def\csname @#1@gloskip\endcsname{& \\}%
\ifthenelse{\equal{\csname gls@#1@number\endcsname}{none}}{%
\expandafter\def\csname @#1@glodelim\endcsname{}}{%
\expandafter\def\csname @#1@glodelim\endcsname{, }}%
\else
\expandafter\def\csname @#1@gloskip\endcsname{& & \\}%
\expandafter\def\csname @#1@glodelim\endcsname{& }%
\fi
\expandafter\def\csname @#1@gloitem\endcsname##1{##1 &}%
}
```

Need a way to avoid conflict with the array package. In an earlier version I defined a new column type if the array package was being used, however this restricts the ability to have multiple glossaries with different column alignments.

```
\newcommand{\@st@rtglostable}[2]{%
\gls@ta={\begin{#1}}\gls@tb=\expandafter{#2}%
\edef\@st@rtglost@ble{\the\gls@ta{\the\gls@tb}}
\@st@rtglost@ble}
```

Define command to switch to `super` style:

```
\newcommand{\gls@setsuper}[1][glossary]{%
\gls@setalignment[#1]%
\gls@setheader[#1]%
%
\expandafter\def\csname @gls@beforeglos@#1\endcsname{%
\tablehead{\@glossaryheader}\tabletail{\glosstail}%
\if\glossaryalignment\relax
\expandafter\let\expandafter\@glossaryalignment
\csname @gls@align@#1\endcsname
\else
\let\@glossaryalignment\glossaryalignment
\fi
\@st@rtglostable{supertabular}\@glossaryalignment}
%
\expandafter\def\csname @gls@afterglos@#1\endcsname{%
\end{supertabular}}%
}
```

Define command to switch to `long` style:

```
\newcommand{\gls@setlong}[1][glossary]{%
\gls@setalignment[#1]%
\gls@setheader[#1]%
%
\expandafter\def\csname @gls@beforeglos@#1\endcsname{%
\if\relax\glossaryalignment
\expandafter\let\expandafter\@glossaryalignment
\csname @gls@align@#1\endcsname
\else
\let\@glossaryalignment\glossaryalignment
\fi
\@st@rtglostable{longtable}{\@glossaryalignment}
\@glossaryheader\endhead\glosstail\endfoot}
%
\expandafter\def\csname @gls@afterglos@#1\endcsname{%
\end{longtable}}%
}
```

Define command to set the glossary style.

```
\newcommand{\@setglossarystyle}[1][glossary]{%
\@ifundefined{gls@set\gls@style}{%
\PackageError{glossary}{Glossary style '\gls@style' undefined}{}}{%
\ifthenelse{\equal{\gls@number}{}}{}{%
\expandafter\edef\csname gls@#1@number\endcsname{\gls@number}%
\@gls@setnumbering[#1]{\gls@number}%
}%
\csname gls@set\gls@style\endcsname[#1]}}
```

Set main glossary style as per package options

```
\let\gls@number\gls@glossary@number
\@setglossarystyle
```

Define keys to change glossary style. The `style` key sets the basic style.

```
\define@key{glosstyle}
{style}
{\ifthenelse{\equal{#1}{list} \or \equal{#1}{altlist}
\or \equal{#1}{super} \or \equal{#1}{long}}
{\def\gls@style{#1}}
{\PackageError{glossary}
{Unknown glossary style '#1'}
{Available styles are: list, altlist, super and long}}}
```

The `header` key should only be used in conjunction with one of the tabular-type styles. If set to `plain`, a header row will be used.

```
\define@key{glosstyle}
{header}[plain]{\ifthenelse{\equal{#1}{none} \or \equal{#1}{plain}}
{\def\gls@header{#1}}
{\PackageError{glossary}
{Unknown glossary style '#1'}
{Available styles are: none and plain}}}
```

The `border` key should only be used in conjunction with one of the tabular-type styles. If set to `plain`, a border will be placed around the glossary.

```
\define@key{glosstyle}
{border}[plain]{\ifthenelse{\equal{#1}{none} \or \equal{#1}{plain}}
{\def\gls@border{#1}}
{\PackageError{glossary}
{Unknown glossary border '#1'}
{Available styles are: none and plain}}}
```

The `cols` key should only be used in conjunction with one of the tabular-type styles. If set to `2`, the description and page list will both be placed in the second column, if set to `3`, the description will go in the second column, and the page list will go in the third column.

```
\define@key{glosstyle}{cols}{\gls@cols=#1\relax
\ifthenelse{\gls@cols<2 \or \gls@cols>3}
{\PackageError{glossary}
{invalid number of columns}
{The cols option can only be 2 or 3}}
{}}
```

The number key may either be none or the name of a counter.

```
\define@key{glosstyle}
{number}
{\ifthenelse{\equal{#1}{none}}
{\def\gls@number{#1}}
{\@ifundefined{c@#1}{
\PackageError{glossary}
{Unknown glossary number style '#1'}
{You may either specify "none" or the name of a counter,
e.g. "section"}\def\gls@number{page}}{\def\gls@number{#1}}}}
```

Provide a means of setting the style for a given glossary type.

```
\newcommand{\setglossarystyle}[2][glossary]{%
\def\gls@number{}%
\setkeys{glosstyle}{#2}%
\@setglossarystyle[#1]%
}
```

Set the delimiter for the case where there is no numbering and there aren't 3 columns.

```
\ifthenelse{\equal{\gls@glossary@number}{none} \and \gls@cols<3}{%
\renewcommand{\@glossary@glodelim}{}}{}
```

16.4 Makeindex style file

This is the code to generate the .ist file. First define a switch that governs whether or not to write the ist file.

```
\newif\ifist
\let\noist=\istfalse
\if@filesw\isttrue\else\istfalse\fi
```

Provide a command to write the ist file. This will cause a problem with ngerman because the behaviour of the double quote character changes. Any packages that modify this character should be loaded after the .ist file is written.

```
\newwrite\istfile
\catcode`\%11\relax
\newcommand{\writeist}{
\protected@write\@auxout{}{\protect\@istfilename{\istfilename}}
\openout\istfile=\istfilename
\write\istfile{% makeindex style file created by LaTeX for document "\jobname" on \the\year-\t
\write\istfile{keyword "\string\\glossaryentry"}
\write\istfile{preamble "\string\\begin{theglossary}"}
\write\istfile{postamble "\string\n\string\\end{theglossary}\string\n"}
\write\istfile{group_skip "\string\\gloskip "}
\write\istfile{item_0 "\string\n\string\n\string\\gloitem "}
\write\istfile{delim_0 "\string\n\string\\glodelim "}
\write\istfile{page_compositor "\pagecompositor"}
\write\istfile{delim_n "\string\\delimN "}
\write\istfile{delim_r "\string\\delimR "}
\write\istfile{delim_t "\string\\delimT "}
\write\istfile{headings_flag 1}
\write\istfile{heading_prefix "\string\\glogroup"}
\write\istfile{symhead_positive "Symbols"}
\write\istfile{numhead_positive "Numbers"}
\closeout\istfile
}
\catcode`\%14\relax
```

Redefine `\makeglossary` so that it creates the `.ist` file. Once it is created, the `\ifist` flag is set to false to prevent repeated creation of the file in the event that another glossary-style type is created. If a different `.ist` file is desired for each glossary type, you will need to precede each `\make`⟨*type*⟩ with `\isttrue` and changed the definition of `\istfilename`. (This is unlikely to occur unless more than one type of page compositor is required.) If you do this, remember to pass the correct ist file to makeindex. I have removed `\@sanitize` at the recommendation of Ulrich Diez.

```
\renewcommand{\makeglossary}{
\newwrite\@glossaryfile
\immediate\openout\@glossaryfile=\jobname.glo
\renewcommand{\glossary}[1][]{\gdef\@glo@l@bel{##1}%
\@bsphack \begingroup \@wrglossary }
\typeout {Writing glossary file \jobname .glo }
\let \makeglossary \@empty
\ifist\writeist\fi
\noist}
```

The `\glossary` command has been modified to allow for an optional argument to modify the label. This is the default definition of `\glossary`, it doesn't write anything to the `.glo` file. It doesn't use `\setkeys`, so `\@sanitize` is used here. Use `\makeglossary` to redefine it so that entries are written to the `.glo` file.

```
\renewcommand{\glossary}[1][]{%
\@bsphack\begingroup\@sanitize\@index}
```

16.5 Defining a new glossary type

First parameter (optional) is the extension of the log file (information used by `makeglos.pl` but not LaTeX). Second parameter is the name of new glossary type e.g. `notation`. Third parameter is the extension of output file (equivalent to `ind` or `glo`. Fourth parameter is the extension of input file (equivalent to `idx` or `gls`). The fifth parameter (optional) is the format.

```
\newcommand{\newglossarytype}[4][glg]{
\@ifundefined{#2}{%
\protected@write\@auxout{}{\@newglossarytype[#1]{#2}{#3}{#4}}%
\def\@glstype{#2}\def\@glsout{#3}\def\@glsin{#4}%
\expandafter\edef\csname gls@\@glstype @number\endcsname{%
\gls@glossary@number}%
\expandafter\gdef\csname glsX\@glstype Xnumformat\endcsname{%
\glsXglossaryXnumformat}%
\expandafter\gdef\csname @\@glstype @delimN\endcsname{%
\@glossary@delimN}%
\expandafter\gdef\csname @\@glstype @delimR\endcsname{%
\@glossary@delimR}%
\expandafter\gdef\csname @gls@\@glstype @inext\endcsname{#4}%
\expandafter\def\csname @gls@#2@type\endcsname{#4}%
\expandafter\edef\csname make\@glstype\endcsname{%
\noexpand\@m@kegl@ss{\@glstype}{\@glsout}}
\expandafter\edef\csname \@glstype\endcsname{%
\noexpand\@gl@ss@ary{\@glstype}}
\expandafter\edef\csname x\@glstype\endcsname{%
\noexpand\@Gl@ss@ary{\@glstype}}
\@namedef{print\@glstype}{%
\printglossary[#2]}%
}{\PackageError{glossary}{Command
\expandafter\string\csname #2\endcsname \space already defined}{%
You can't call your new glossary type '#2' because there already
exists a command with this name}}%
\@@n@wglostype}
\newcommand{\@@n@wglostype}[1][]{%
\setglossarystyle[\@glstype]{#1}}
```

The command `\@newglossarytype` is written to the auxiliary file and is only used by makeglos.pl. LaTeX should ignore it.

```
\newcommand{\@newglossarytype}[4][glg]{}
```

Define equivalent of `\makeglossary`:

```
\newcommand\@m@kegl@ss[2]{%
\expandafter\newwrite\csname @#1file\endcsname
\expandafter\immediate\expandafter
\openout\csname @#1file\endcsname=\jobname.#2
\typeout {Writing #1 file \jobname .#2 }
\expandafter\let \csname make#1\endcsname \@empty
\ifist\writeist\fi
\expandafter\def\csname the#1num\endcsname{\thepage}
\noist
}
```

Define the equivalent of `\glossary`.

```
\newcommand\@gl@ss@ary[2][]{\@ifundefined{@#2file}{%
\@bsphack\begingroup\@sanitize \@index}{%
\gdef\@glo@l@bel{#1}%
\@bsphack \begingroup \@wrglossary[#2]}}
```

Define the equivalent of `\xglossary`.

```
\newcommand{\@Gl@ss@ary}{%
\renewcommand{\@@wrglossary}[1]{%
\glossref{\@glo@l@bel}{##1}\renewcommand{\@@wrglossary}{}}%
\@gl@ss@ary}
```

The command `\newglossarytype` should only be used in the preamble.

```
\@onlypreamble{\newglossarytype}
```

16.6 Acronyms

Define `\newacronym`[⟨*cmd-name*⟩]{⟨*abbrv*⟩}{⟨*long name*⟩}{⟨*glos entry*⟩}

```
\newcommand\@acrnmsh{}
\newcommand\@sacrnmsh{}
\newcommand\@acrnmln{}
\newcommand\@acrnmcmd{}
\newcommand\@acrnmgls{}
\newcommand\@acrnmins{}
```

List of all defined acronyms.

```
\toksdef\@glo@tb=2
\newcommand{\@acr@list}{}
```

append acronym to list

```
\newcommand{\@acr@addtolist}[1]{\edef\@glo@ta{#1}%
\ifthenelse{\equal{\@acr@list}{}}{%
\edef\@acr@list{\@glo@ta}}{%
\@glo@tb=\expandafter{\@acr@list}%
\edef\@acr@list{\the\@glo@tb,\@glo@ta}}}
```

Specify how to control the way the name key is set for acronyms.

```
\newcommand{\@acronymnamefmt}{\glolong\ (\gloshort)}
\newcommand{\setacronymnamefmt}[1]{\def\@acronymnamefmt{#1}}
```

Specify how to control the way the description key is set for acronyms.

```
\newcommand{\@acronymdescfmt}{\glodesc}
\newcommand{\setacronymdescfmt}[1]{\def\@acronymdescfmt{#1}}
```

Format the acronym abbreviation in the format specified by `\acronymfont`. This simply prints its argument by default.

```
\newcommand{\acronymfont}[1]{#1}
```

This command has been restructured as from v2.17

```
\newcommand{\newacronym}[4][]{%
\ifthenelse{\equal{#1}{}}{\renewcommand\@acrnmcmd{#2}}{%
\renewcommand\@acrnmcmd{#1}}
\@ifundefined{\@acrnmcmd}{%
\expandafter\newcommand\csname\@acrnmcmd short\endcsname{%
#2\protect\glsxspace}
\expandafter\newcommand\csname\@acrnmcmd @nx@short\endcsname{#2}
\expandafter\newcommand\csname\@acrnmcmd long\endcsname{%
#3\protect\glsxspace}
\expandafter\newcommand\csname\@acrnmcmd @nx@long\endcsname{#3}
\def\@acrn@entry{#4}%
{%
% extract description
\expandafter\@gls@getdescr\expandafter{\@acrn@entry}%
\let\glodesc\@glo@desc%
\def\glolong{#3}%
\@onelevel@sanitize\glolong
\def\gloshort{\noexpand\acronymfont{#2}}%
\@onelevel@sanitize\gloshort
\expandafter\protected@xdef\expandafter\@acrnamefmt{\@acronymnamefmt}
\expandafter\protected@xdef\expandafter\@acrdesc{\@acronymdescfmt}
}%
\@acr@addtolist{\@acrnmcmd}
\@glo@tb=\expandafter{\@acrn@entry}%
\protected@edef\@acr@glsentry{name={\@acrnamefmt},%
format=glsnumformat,sort={\@acrnmcmd},\the\@glo@tb,%
description={\@acrdesc}}%
\@glo@tb=\expandafter{\@acr@glsentry}%
\newboolean{\@acrnmcmd first}\setboolean{\@acrnmcmd first}{true}
\expandafter\protected@edef\csname \@acrnmcmd\endcsname{%
\noexpand\@ifstar{\csname @s@\@acrnmcmd\endcsname}{%
\csname @\@acrnmcmd\endcsname}}
```

```
\ifglshyperacronym % hyperlinks
% unstarred version
\expandafter\protected@edef\csname @\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})\noexpand\unsetacronym{\@acrnmcmd}%
}{\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}\noexpand\glsxspace}
% starred version
\expandafter\protected@edef\csname @s@\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\noexpand\expandafter\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})%
\noexpand\unsetacronym{\@acrnmcmd}}{%
\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\noexpand\expandafter\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}\noexpand\glsxspace}
\else % no hyperlinks
% unstarred version
\expandafter\protected@edef\csname @\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})\noexpand\unsetacronym{\@acrnmcmd}%
}{\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}%
\noexpand\glsxspace}
% starred version
\expandafter\protected@edef\csname @s@\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\noexpand\expandafter
\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})%
\noexpand\unsetacronym{\@acrnmcmd}}{%
\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\noexpand\expandafter\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}\noexpand\glsxspace}
\fi
}{%
\PackageError{glossary}{Command '\expandafter\string
\csname\@acrnmcmd\endcsname' already defined}{%
The command name specified by \string\newacronym already exists.}}}
```

Define a command to use a given acronym.

```
\newcommand{\useacronym}{\@ifstar\@suseacronym\@useacronym}
\newcommand{\@suseacronym}[2][]{{\let\glsxspace\relax
\def\@acrnmins{#1}\csname @s@#2\endcsname}%
\setboolean{#2first}{false}}
\newcommand{\@useacronym}[2][]{{\let\glsxspace\relax
\def\@acrnmins{#1}\csname @#2\endcsname}%
\setboolean{#2first}{false}}
```

Define a command to use the long form of an acronym without generating a glossary entry. The starred form makes the first character uppercase.

```
\newcommand{\acrln}{\@ifstar\@sacrln\@acrln}
```

Unstarred form:

```
\newcommand{\@acrln}[1]{\@ifundefined{#1long}{%
\PackageError{glossary}{Acronym '#1' has not been defined}{}}{%
\csname#1@nx@long\endcsname}}
```

Starred form:

```
\newcommand{\@sacrln}[1]{\@ifundefined{#1long}{%
\PackageError{glossary}{Acronym '#1' has not been defined}{}}{%
\expandafter\expandafter\expandafter
\MakeUppercase\csname#1@nx@long\endcsname}}
```

As above, but for the short form.

```
\newcommand{\acrsh}{\@ifstar\@sacrsh\@acrsh}
```

Unstarred form:

```
\newcommand{\@acrsh}[1]{\@ifundefined{#1short}{%
\PackageError{glossary}{Acronym '#1' has not been defined}{}}{%
\acronymfont{\csname#1@nx@short\endcsname}}}
```

Starred form:

```
\newcommand{\@sacrsh}[1]{\@ifundefined{#1short}{%
\PackageError{glossary}{Acronym '#1' has not been defined}{}}{%
\acronymfont{\expandafter\expandafter\expandafter
\MakeUppercase\csname#1@nx@short\endcsname}}}
```

Define a means of determining whether an acronym has been used or not. This was mainly included for use with LaTeX2HTML which currently has no ifthen style.

```
\newcommand{\ifacronymfirstuse}[3]{%
\@ifundefined{if#1first}{%
\PackageError{glossary}{Acronym '#1' not defined}{}}{%
\ifthenelse{\boolean{#1first}}{#2}{#3}}}
```

Provide a means of resetting an acronym so that it is expanded next time it is used.

```
\newcommand{\resetacronym}[1]{%
\@ifundefined{if#1first}{%
\PackageError{glossary}{Acronym '#1' not defined}{}}{%
\ifglsglobal
\expandafter\global\csname #1firsttrue\endcsname
\else
\setboolean{#1first}{true}%
\fi}}
```

Reverse of the above.

```
\newcommand{\unsetacronym}[1]{%
\@ifundefined{if#1first}{%
\PackageError{glossary}{Acronym '#1' not defined}{}}{%
\ifglsglobal
\expandafter\global\csname #1firstfalse\endcsname
\else
\setboolean{#1first}{false}%
\fi}}
```

Reset all acronyms so that they will all be expanded when next used.

```
\newcommand{\resetallacronyms}{%
\@for\@acr:=\@acr@list\do{\resetacronym{\@acr}}}
```

Ensure that all acronyms are not expanded, even if they haven't yet been used.

```
\newcommand{\unsetallacronyms}{%
\@for\@acr:=\@acr@list\do{\unsetacronym{\@acr}}}
```

Check to see if acronyms should be separate from glossary

```
\ifglsacronym
\newglossarytype[alg]{acronym}{acr}{acn}
\providecommand{\acronymname}{List of Acronyms}
\else
\let\acronym=\glossary
\let\xacronym=\xglossary
\fi
```

16.7 Glossary Hyperlinks

This section deals with commands that are used to make the numbers in the glossary have hyperlinks, if hyperlinks are supported.

The command `\glshyper` is a modification of hyperref's `\hyperpage` command, but it uses `\delimR` instead of a dash, and `\delimN` instead of a comma. The command was originally called `\glshyperpage` but was modified in version 2.4 to enable page to be substituted with some arbitrary counter (which should be specified as the first argument).

```
\ifglshyper
\def\glshyper#1#2{\@glshyper{#1}#2\delimR \delimR \\}
\def\@glshyper#1#2\delimR #3\delimR #4\\{%
\ifx\\#3\\%
\@delimNhyper{#1}{#2}%
\else
\@ifundefined{hyperlink}{#2\delimR #3}{%
\hyperlink{#1.#2}{#2}\delimR \hyperlink{#1.#3}{#3}}%
\fi
}
```

For a list of individual pages instead of a range:

```
\def\@delimNhyper#1#2{\@@delimNhyper{#1}#2\delimN \delimN\\}
\def\@@delimNhyper#1#2\delimN #3\delimN #4\\{%
  \ifx\\#3\\%
    \@ifundefined{hyperlink}{#2}{\hyperlink{#1.#2}{#2}}%
  \else
    \@ifundefined{hyperlink}{#2\delimN #3}{%
\hyperlink{#1.#2}{#2}\delimN \hyperlink{#1.#3}{#3}}%
  \fi
}
```

To maintain backwards compatibility, define `\glshyperpage` and `\glshypersection`. These commands may be removed at a later date, so don't use them.

```
\newcommand\glshyperpage[1]{\glshyper{page}{#1}}
\newcommand\glshypersection[1]{\glshyper{section}{#1}}
```

If chapters are defined, modify `\@chapter` so that is adds a `section.`⟨*n*⟩`.0` target (otherwise it gets too complicated if you have to work out whether to use the chapter or section counter—there's more than enough conditional code in this package already!)

```
\@ifundefined{chapter}
{}
{\let\@gls@old@chapter\@chapter
\def\@chapter[#1]#2{\@gls@old@chapter[{#1}]{#2}%
\@ifundefined{hyperdef}{}{\hyperdef{section}{\thesection}{}}}}
```

Provide `\hyper`⟨*xx*⟩ to make it easier to change the page number format to `bf`, `sf`, `tt` and `it` if you are using hyperlinks. The optional first argument (new to version 2.4) specifies the counter being used.

```
\providecommand\hyperrm[2][\gls@number]{%
\textrm{\glshyper{#1}{#2}}}
\providecommand\hypersf[2][\gls@number]{%
\textsf{\glshyper{#1}{#2}}}
\providecommand\hypertt[2][\gls@number]{%
\texttt{\glshyper{#1}{#2}}}
\providecommand\hyperbf[2][\gls@number]{%
\textbf{\glshyper{#1}{#2}}}
\providecommand\hyperit[2][\gls@number]{%
\textit{\glshyper{#1}{#2}}}
```

The following were added in version 2.4:

```
\providecommand\hyperem[2][\gls@number]{%
\emph{\glshyper{#1}{#2}}}
\providecommand\hyperup[2][\gls@number]{%
\textup{\glshyper{#1}{#2}}}
\providecommand\hypersl[2][\gls@number]{%
\textsl{\glshyper{#1}{#2}}}
\providecommand\hypersc[2][\gls@number]{%
\textsc{\glshyper{#1}{#2}}}
\providecommand\hypermd[2][\gls@number]{%
\textmd{\glshyper{#1}{#2}}}
```

Hyperlinks not enabled.

```
\else
\providecommand\hyperrm[2][]{\textrm{#2}}
\providecommand\hypersf[2][]{\textsf{#2}}
\providecommand\hypertt[2][]{\texttt{#2}}
\providecommand\hypermd[2][]{\textmd{#2}}
\providecommand\hyperbf[2][]{\textbf{#2}}
\providecommand\hyperit[2][]{\textit{#2}}
\providecommand\hypersl[2][]{\textsl{#2}}
\providecommand\hyperup[2][]{\textup{#2}}
\providecommand\hypersc[2][]{\textsc{#2}}
```

```
\providecommand\hyperem[2][]{\emph{#2}}
\fi
```

Change History

1.0
- General: Initial version 1

1.1
- General: 'delimR 18
  - 'glossarypostamble 16
  - 'glossarypreamble 16
  - Increased User Flexibility 17
  - `'delimN` 18
  - `'glsnumformat` 17
  - Package option `number` 7

2.0
- General: Acronyms 12
  - Hyper page formats: `'hypersf`, `'hypertt`, `'hyperbf` and `'hyperbf` 3
  - Package option `altlist` style . . 7
  - Package option `hyper` 8
  - Package option `toc` 7
  - primary acronym number format `'glsprimaryfmt` 27

2.01
- General: Fixed conflict with datetime package 25

2.1
- General: made glossary compatible with array package 18
  - name field can be omitted in `'newacronym` 13

2.11
- General: `'useacronym` 14

2.12
- General: Hyper page format: `'hyperrm` 3
  - Package option `section` 8
  - primary acronym number format `'glsprimaryfmt` no longer used 27

2.13
- General: Package option `hyperacronym` 27

2.14
- General: 'ifacronymfirstuse added 14
  - 'resetacronym added 14
  - 'saveglosentry added 27
  - 'setglossary added 17
  - 'useGlosentry added 4
  - 'useglosentry added 4
  - 'xglossary added 4
  - makeglos -m switch added 6

2.15
- General: 'shortglossaryname 16

2.16
- General: fixed bug preventing changes to 'glossaryname and 'shortglossaryname 16

2.17
- General: 'storeglosentry added . . . 4
  - Package option `acronym` 8

2.18
- General: 'gls added 4
  - Fixed bug in 'useacronym 14

2.19
- General: 'acrln added 14
  - 'acrsh added 14
  - fixed bug in 'storeglosentry 4

2.2
- General: 'glossaryname now defined using 'providecommand instead of 'newcommand 16

2.21
- General: 'resetallacronyms added 14

2.22
- General: Added 'acronymfont . . . 15
  - added makeglos.bat file 2
  - Added provision for 'xspace . . 15
  - changed makeglos to read information in from .aux instead of .log file 11

2.23
- General: Fixed minor bug with hyperlinks and 'glsxspace 15

2.24
- General: Package option `hypertoc` 8

2.26
- General: Fixed bug in 'useacronym 14

2.28
- General: Fixed erroneous spaces occuring while using xspace . . . 15

2.3
- General: 'delimT 18
  - 'gloskip 18
  - 'glsafternum 17
  - 'glsbeforenum 17
  - Added extra optional argument to 'newglosarytype 11

made glossary compatible with array package 18

2.31

General: 'unsetacronym added . . 14

'unsetallacronyms added 14

Package option **global** 8

2.32

General: added **'setacronymdescfmt** 13

2.4

General: 'glspageheader added . . 16

Added facility to insert text between groups 18

Hyper page formats updated to take two arguments. Additional formats also provided 3

Package option **number** modified so that any counter can be used 7

primary acronym number format **'glsprimaryfmt** has been removed 27

provision for 'glossarysubheader added 19

Index