

glossaries.sty v 1.12: L^AT_EX 2_ε Package to Assist Generating Glossaries

Nicola L.C. Talbot

School of Computing Sciences
University of East Anglia
Norwich, Norfolk
NR4 7TJ, United Kingdom.

<http://theoval.cmp.uea.ac.uk/~nlct/>

8th March 2008

Contents

1	Introduction	2
1.1	Generating the associated glossary files	3
1.2	Troubleshooting	4
2	A Quick Guide For The Impatient	4
3	Overview	10
3.1	Package Options	10
3.2	Defining Glossary Entries	12
3.3	Number lists	14
3.4	Links to Glossary Entries	14
3.4.1	Changing the format of the link text	18
3.4.2	Enabling and disabling hyperlinks to glossary entries	20
3.5	Adding an entry to the glossary without generating text	20
3.6	Displaying a glossary	20
3.6.1	Changing the way the entry name appears in the glossary . .	22
3.7	Defining New Glossaries	22
3.8	Acronyms	22
3.9	Unsetting and resetting entry flags	26
3.10	Glossary Styles	27
3.11	Defining your own glossary style	29
3.11.1	Example: creating a completely new style	30
3.11.2	Example: creating a new glossary style based on an existing style	31
4	Mfirstuc Package	31

5	Documented Code	32
5.1	Package Definition	32
5.2	Package Options	32
5.3	Default values	36
5.4	Loops and conditionals	40
5.5	Defining new glossaries	42
5.6	Defining new entries	43
5.7	Resetting and unsetting entry flags	47
5.8	Loading files containing glossary entries	48
5.9	Using glossary entries in the text	48
5.9.1	Links to glossary entries	49
5.9.2	Displaying entry details without adding information to the glossary	74
5.10	Adding an entry to the glossary without generating text	76
5.11	Creating associated files	77
5.12	Writing information to associated files	79
5.13	Displaying the glossary	80
5.14	Acronyms	85
5.15	Additional predefined acronym styles	87
5.16	Predefined Glossary Styles	92
6	Mfirstuc package	92
6.0.1	Glossary hyper-navigation definitions (glossary-hypernav package)	93
6.0.2	List Style (glossary-list package)	95
6.0.3	Glossary Styles using longtable (the glossary-long package)	96
6.0.4	Glossary Styles using supertabular environment (glossary-super package)	98
6.1	Multi-Lingual Support	101
6.1.1	Babel Captions	101
6.1.2	Danish Dictionary	103
6.1.3	Dutch Dictionary	103
6.1.4	English Dictionary	104
6.1.5	French Dictionary	104
6.1.6	German Dictionary	104
6.1.7	Irish Dictionary	105
6.1.8	Italian Dictionary	105
6.1.9	Magyar Dictionary	105
6.1.10	Spanish Dictionary	105

1 Introduction

The `glossaries` package is provided to assist generating glossaries. It has a certain amount of flexibility, allowing the user to customize the format of the glossary and define multiple glossaries. It also supports acronyms and glossary styles that include symbols (in addition to a name and description) for glossary entries. There is provision for loading a database of glossary terms where only those terms used in the text are added to the glossary. This package replaces the `glossary` package which is now obsolete.

As from version 1.08, the `glossaries` package now has limited multi-lingual support, thanks to all the people who have sent me the relevant translations either via email or via `comp.text.tex`. However you must load `babel` *before* `glossaries` to enable this. (See [subsection 5.3](#) for the predefined names used by the `glossaries` package.) As from version 1.1, the `glossaries` package also supports Till Tantau's `translator` package provided it is loaded before the `glossaries` package. This makes it easier to change the default translations or add new dictionaries. If you don't want to use the predefined translations, you can use the package option `translate=false` and supply your own translations.

This documentation is structured as follows: [section 2](#) is for people who want a few quick pointers of how to get started, without having to read through lengthy descriptions, [section 3](#) gives an overview of available commands and their syntax, [section 4](#) describes the associated `mfirstuc` package, [section 5](#) contains the documented source code for those who want to know more about how the package works and how to do more complicated things, such as changing the way glossary entries appear.

1.1 Generating the associated glossary files

The `glossaries` package comes with the Perl script `makeglossaries` which will run `makeindex` on all the glossary files using a customized `makeindex .ist` style file (which is created by `\makeglossaries`). The relevant extensions are obtained from the auxiliary file, so you should only pass the basename as the argument. For example, if your document is called `myfile.tex`, do:

```
latex myfile
makeglossaries myfile
latex myfile
```

You may need to explicitly load `makeglossaries` into Perl:

```
perl makeglossaries myfile
```

There is a batch file called `makeglossaries.bat` which does this for Windows users.

If you don't have Perl installed, you will have to run `makeindex` for each glossary type you have defined. For example, if you have used the `acronym` package option then you will have both a main glossary as well as a list of acronyms, so you will need to do (assuming your document is called `myfile.tex`):

```
makeindex -s myfile.ist -t myfile.glg -o myfile.gls myfile.glo
makeindex -s myfile.ist -t myfile.alg -o myfile.acr myfile.acn
```

This requires remembering all extensions for each of the glossaries defined in your document, so where possible you should use `makeglossaries` instead to reduce the possibility of error. Don't pass all the glossary files in a single call to `makeindex` or it will merge all your glossaries into a single glossary.

If any problems occur, remember to check the transcript files (e.g. `.glg` or `.alg`) for messages.

1.2 Troubleshooting

The glossaries package comes with a minimal file called `minimalgls.tex` which can be used for testing. This should be located in `texmf/doc/latex/glossaries/samples/`. Further information on debugging L^AT_EX code is available at <http://theoval.cmp.uea.ac.uk/~nlct/latex/minexample/>.

There is a list of frequently asked questions for the glossaries package available at <http://theoval.cmp.uea.ac.uk/~nlct/latex/packages/faq/glossariesfaq.html>.

2 A Quick Guide For The Impatient

This section is for people who want a few quick pointers of how to get started, without having to read through lengthy descriptions.

1. Load glossaries *after* hyperref:

```
\usepackage{hyperref}
\usepackage{glossaries}
```

Similarly for the html package:

```
\usepackage{html}
\usepackage{glossaries}
```

2. Always use `\makeglossaries` if you want the glossary entries to be written to the glossary file:

```
\documentclass{article}
\usepackage{glossaries}
\makeglossaries
```

If you don't use `\makeglossaries`, your glossaries will not appear in the document!

3. Use `\printglossaries` to make your glossaries appear in the document at that point. For example:

```
\maketitle
\printglossaries
\section{Introduction}
```

Note that only the glossary entries that have been used in the document text will appear in the glossary.

4. When you have created your document, run L^AT_EX on it, then the Perl script `makeglossaries`, then run L^AT_EX on it again:

```
latex myfile
makeglossaries myfile
latex myfile
```

(You may need to run L^AT_EX again if you have used the `toc` package option.)
 If you use Windows, there is a batch file called `makeglossaries.bat` which you can use, but you will still need Perl installed.

5. New glossaries can be defined using:

```
\newglossary[⟨log-ext⟩]{⟨label⟩}{⟨in-ext⟩}{⟨out-ext⟩}{⟨title⟩}
```

where `⟨label⟩` is an identifying label, `⟨in-ext⟩` is the extension of the file to be created by `makeindex` (called by `makeglossaries`), `⟨out-ext⟩` is the extension of the file to be read by `makeindex` and `⟨title⟩` is the title for this new glossary. The first optional argument `⟨log-ext⟩` specifies the extension of the `makeindex` transcript file. Example:

```
\newglossary[nlg]{notation}{not}{ntn}{Notation}
```

This glossary's label is `notation` and its title will be `Notation`. If you use `makeglossaries`, the `makeindex` transcript will be written to a file with the extension `.nlg`. If `⟨log-ext⟩` is omitted, the extension `.glg` will be used.

6. Any new glossaries must be defined before `\makeglossaries`

```
\documentclass{article}
\usepackage{glossaries}
\newglossary{notation}{not}{ntn}{Notation}
\makeglossaries
```

7. If you use the `acronym` package option, the `glossaries` package will automatically create a new glossary type labelled `acronym`:

```
\usepackage[acronym]{glossaries}
```

8. If your pages have a hyphen compositor (i.e. your page numbers appear in the form 2-1), redefine `\glscompositor` *before* `\makeglossaries`:

```
\documentclass{article}
\usepackage{glossaries}
\renewcommand{\glscompositor}{-}
\makeglossaries
```

9. To add the glossaries to the table of contents use the `toc` package option:

```
\usepackage[toc]{glossaries}
```

10. Define a new entry with:

```
\newglossaryentry{⟨label⟩}{⟨key-val list⟩}
```

The `⟨key-val list⟩` must at least contain a `name` key and a `description` key. For example:

```
\newglossaryentry{perl}{name=Perl,
description=A scripting language}
```

In this example, I have given the entry the label `perl`. Whenever I want to use this entry, that is the label I need to use to identify it.

11. If the entry name starts with an accented letter, you will need to group the first letter (otherwise it will cause a problem for `\Gls` and `\Glspl`):

```
\newglossaryentry{elite}{name={{\'}lite},
description={select group or class}}
```

Likewise with commands such as `\ae` and `\oe`:

```
\newglossaryentry{oesophagus}{%
name={{\oe}sophagus},
description={canal from mouth to stomach}}
```

12. If you have multiple glossaries, use the `type` key to specify in which glossary the entry belongs. For example:

```
\newglossary{languages}{lan}{lng}{Index of Languages}

\makeglossaries

\newglossaryentry{perl}{name=Perl,
description=A scripting language,
type=languages}
```

If `type` is omitted, the default glossary is used.

13. Remember to group values that have a comma or equal sign. For example:

```
\newglossaryentry{pagelist}{name=page list,
description={A list of individual pages or page ranges
(e.g.\ 1,2,4,7--9)}}
```

14. Plural forms are assumed to be the singular form with an “s” appended, unless otherwise specified. To specify an irregular plural, use the `plural` key. For example:

```
\newglossaryentry{matrix}{name=matrix,
description=rectangular array of quantities,
plural=matrices}
```

15. The way the term appears in the main text can be different from the way the term appears in the glossary:

```
\newglossaryentry{matrix}{name=Matrix,
description=rectangular array of quantities,
text=matrix,
plural=matrices}
```

In this example, the entry name appears as “Matrix” in the glossary, and either “matrix” or “matrices” in the text.

16. The way the term appears on first use can be different to the way it appears subsequently:

```
\newglossaryentry{singmtx}{name=Singular Matrix,
description=A matrix with a zero determinant,
first=singular matrix (SM),
text=SM,
firstplural=singular matrices (SMs)}
```

In this example, the entry name appears as “Singular Matrix” in the glossary, and in the text it appears as “singular matrix (SM)” or “singular matrices (SMs)” the first time the entry is used, and subsequently appears as “SM” or “SMs”.

17. The quick and easy way to define an acronym is to use:

```
\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrev⟩}{⟨long⟩}
```

For example:

```
\newacronym{svm}{SVM}{support vector machine}
```

This is equivalent to:

```
\newglossaryentry{svm}{type=\acronymtype,
name={SVM},
description={support vector machine},
text={SVM},
first={support vector machine (SVM)},
plural={SVMs},
firstplural={support vector machines (SVMs)}}
```

(The value of `\acronymtype` varies depending on whether the `acronym` package option is used or not. The optional argument `⟨key-val list⟩` can be used to override any of the `\newglossaryentry` keys; for example, if the acronym has an irregular plural.)

18. The font used to display the entry name in the glossary is governed by `\glsnamefont`. This can be redefined as required. For example, to make the entry names appear in a medium sans-serif font do:

```
\renewcommand{\glsnamefont}[1]{\textsf{\mdseries #1}}
```

Note that the list-like glossary styles defined in the `glossary-list` package place the entry name in the optional argument to `\item`, so they will appear in bold, unless you redefine `\glsnamefont` to counteract the bold font.

19. In the document use `\gls{⟨label⟩}` to use a predefined term (this will also enter the term into the associated glossary output file). For example:

```
A \gls{singmtx} is a matrix with a zero determinant.
```

20. Other variations:

- `\Gls{⟨label⟩}` : like `\gls`, but with first letter in upper case
- `\GLS{⟨label⟩}` : like `\gls`, but all upper case.
- `\glspl{⟨label⟩}` : use plural
- `\Glspl{⟨label⟩}` : use plural with first letter in upper case
- `\GLSpl{⟨label⟩}` : use plural but all upper case
- `\glslink{⟨label⟩}{⟨link text⟩}` : use `⟨link text⟩` to link to the given entry in the glossary.

For example, the following will produce the plural form with the first letter in uppercase:

`\Glspl{singmtx}` are matrices with a zero determinant.

21. Additional text can be appended to the link using the end optional argument. For example, to form the possessive:

The `\gls{singmtx}[’s]` dimensions `\ldots`

22. The format of the associated entry number can be changed using the `format` key in the optional argument. Note that the value of the `format` key should be the name of a command *without* the initial backslash. For example:

The primary definition of `\glspl[format=textbf]{singmtx}`.

In this example the relevant glossary entry will have the page number in bold (since it uses `\textbf`) but it will no longer have a hyperlink (if hyperlinks are enabled.)

23. The `glossaries` package provides commands to change the font whilst ensuring that the number remains a hyperlink. These are of the form `\hyper⟨xx⟩` and are equivalent to the standard font changing commands of the form `\text⟨xx⟩`, as well as `\hyperemph` (which uses `\emph`.) For example:

The primary definition of `\glspl[format=hyperbf]{singmtx}`.

24. Don’t use declarations in `format` as this can cause unpredictable results, as there is no guarantee that the effect will be localised to the required text.
25. Entries can be added to the glossary without producing any text using `\glsadd{⟨label⟩}` or `\glsaddall`. These commands also taken an optional argument where you can specify the format. For example

`\glsadd[format=hyperbf]{singmtx}`

will add a line to the glossary file for the specified term, but will not produce any text where the command occurs.

26. A number range can be entered using `format=(` and `format=)` to mark the beginning and ending of the range¹. For example:

```
\glsadd[format={}]{singmtx}
This is a very long section all about \glspl{singmtx}.

% lots of text omitted

\glsadd[format=)]{singmtx}
```

This is equivalent to `makeindex`'s `| (` and `|)` formats.

27. You can combine the range markers with a formatting command (again without the preceding backslash). For example:

```
This is the start of a very long section all
about \glspl[format=hyperbf]{singmtx}.

% lots of text omitted

This is the end a very long section all about
\glspl[format=)hyperbf]{singmtx}.
```

28. Only those terms that have actually been used in the document will be placed in the glossary. If you have defined a term that doesn't appear in the document, then it means you haven't used it in the text (either via `\glslink` or `\gls` and related commands, or via `\glsadd` or `\glsaddall`.)
29. To change the sorting order, use the `sort` key. For example:

```
\newglossaryentry{universal}{name={\ensuremath{\mathcal{U}}}},
description=The universal set,
sort=U}
```

30. You don't need to escape `makeindex`'s special characters:

```
\newglossaryentry{quote}{name={"},
description={Double quote character}}

\newglossaryentry{exclam}{name={!},
description={Exclamation mark}}

\newacronym{rna}{RNA}{ribonukleins"aure}
```

31. Associated symbols can also be specified, but whether the symbol appears in the glossary depends on the glossary style. For example:

```
\newglossaryentry{metre}{name={metre},
description={A metric measurement of length},
symbol={m}}
```

¹This is new to version 1.01

The predefined glossary styles that display the entry symbol are: `long4col`, `long4colheader`, `long4colborder`, `long4colheaderborder`, `super4col`, `super4colheader`, `super4colborder` and `super4colheaderborder`. All the other styles supplied by this package ignore the associated symbol.

32. Glossary styles can be set using the `style` package option. For example:

```
\usepackage[style=long3col]{glossaries}
```

or using `\glossarystyle{<style>}`. For example:

```
\glossarystyle{altlist}
```

The predefined glossary styles provided by the `glossaries` bundle are listed in [subsection 3.10](#).

33. The list of numbers associated with each glossary entry can be suppressed using the package option `nonumberlist`:

```
\usepackage[nonumberlist]{glossaries}
```

34. By default, the glossaries will appear in an unnumbered chapter if chapters are defined, otherwise in an unnumbered section. This can be changed using the `section` package option. For example, to make the glossaries appear in an unnumbered section, even if chapters are defined, do:

```
\usepackage[section]{glossaries}
```

Other sectional units can also be specified as `section=<value>`. For example, to make the glossaries appear in unnumbered subsections:

```
\usepackage[section=subsection]{glossaries}
```

3 Overview

3.1 Package Options

The `glossaries` package options are as follows:

toc Add the glossaries to the table of contents.

numberline When used with `toc`, this will add `\numberline{}` in the final argument of `\addcontentsline`. This will align the table of contents entry with the numbered section titles. Note that this option has no effect if the `toc` option is omitted. If `toc` is used without `numberline`, the title will be aligned with the section numbers rather than the section titles.

acronym Make a separate glossary for acronyms.

section This is a `<key>=<value>` option. Its value should be the name of a sectional unit (e.g. chapter). This will make the glossaries appear in the named sectional unit, otherwise each glossary will appear in a chapter, if chapters exists, otherwise in a section. Unnumbered sectional units will be used by default. Example:

```
\usepackage[section=subsection]{glossaries}
```

You can omit the value if you want to use sections, i.e.

```
\usepackage[section]{glossaries}
```

is equivalent to

```
\usepackage[section=section]{glossaries}
```

You can change this value later in the document using `\setglossarysection{<type>}`.

numberedsection The glossaries are placed in unnumbered sectional units by default, but this can be changed using `numberedsection`. This option can take three possible values: `false` (no number, i.e. use starred form), `nolabel` (numbered, i.e. unstarred form, but not labelled) and `autolabel` (numbered with automatic labelling). If `numberedsection=autolabel` is used, each glossary is given a label that matches the glossary type, so the main (default) glossary is labelled `main`, the list of acronyms is labelled `acronym`² and additional glossaries are labelled using the value specified in the first mandatory argument to `\newglossary`. For example, if you load `glossaries` using:

```
\usepackage[section,numberedsection=autolabel]{glossaries}
```

then each glossary will appear in a numbered section, and can be referenced using something like:

The main glossary is in section~\ref{main} and the list of acronyms is in section~\ref{acronym}.

If you can't decide whether to have the acronyms in the main glossary or a separate list of acronyms, you can use `\acronymtype` which is set to `main` if the `acronym` option is not used and is set to `acronym` if the `acronym` option is used. For example:

The list of acronyms is in section~\ref{\acronymtype}.

style This is a `<key>=<value>` option. Its value should be the name of the glossary style to use. Predefined glossary styles are listed in [subsection 3.10](#).

nonumberlist This option will suppress the associated number lists in the glossaries (see also [subsection 3.3](#).)

counter This is a `<key>=<value>` option. The value should be the name of the default counter to use in the number lists.

sanitize This is a `<key>=<value>` option whose value is also a `<key>=<value>` list. By default, the `glossaries` package sanitizes the values of the `name`, `description` and `symbol` keys used when defining a new glossary entry. This may lead to unexpected results if you try to display these values within the document text. This sanitization can be switched off using the `sanitize` package option. (See [subsection 5.2](#) and [subsection 5.6](#) for further details.) For example, to

²if the `acronym` option is used, otherwise the list of acronyms is the main glossary

switch off the sanitization for the `description` and `name` keys, but not for the `symbol` key, do:

```
\usepackage[sanitize={name=false,description=false,%
symbol=true}]{glossaries}
```

Note: this sanitization only applies to the `name`, `description` and `symbol` keys. It doesn't apply to any of the other keys (except the `sort` key which is always sanitized) so fragile commands contained in the value of the other keys must always be protected using `\protect`. Since the value of the `text` key is obtained from the `name` key, you will still need to protect fragile commands in the `name` key if you don't use the `text` key.

babel This is a boolean key and is for use when the `glossaries` package is used in conjunction with the `babel` package. The default is `babel=true`, which adds the glossary translations to `babel`'s captions. If you don't like the translations provided by the `glossaries` package, you can set `babel=false` and provide your own translations.

description This option changes the definition of `\newacronym` to allow a description. See [subsection 3.8](#) for further details.

footnote This option changes the definition of `\newacronym` and the way that acronyms are displayed. See [subsection 3.8](#) for further details.

smallcaps This option changes the definition of `\newacronym` and the way that acronyms are displayed. See [subsection 3.8](#) for further details.

dua This option changes the definition of `\newacronym` so that acronyms are always expanded. See [subsection 3.8](#) for further details.

3.2 Defining Glossary Entries

All glossary entries that are used in a document must be defined in the preamble. Only those entries that occur in the document (using any of the commands described in [subsection 3.4](#) and [subsection 3.5](#)) will appear in the glossary. Each time an entry is used in this way, a line is added to an associated glossary (`.glo`) file, which then needs to be converted into a corresponding `.gls` file which contains the typeset glossary which is input by `\printglossary` or `\printglossaries`. The Perl script `makeglossaries` can be used to call `makeindex`, using a customised `.ist` style file, for each of the glossaries that are defined in the document. Note that there should be no need for you to explicitly edit or input any of these external files.

\makeglossaries The command `\makeglossaries` must be placed in the preamble in order to create the customised `makeindex .ist` style file and to ensure that glossary entries are written to the appropriate output file. If you omit `\makeglossaries` none of the glossaries will be created. Note that if your page numbers use a hyphen compositor, you must set this by redefining `\glscompositor` *before* using `\makeglossaries`:

```
\renewcommand*{\glscompositor}{-}
```

(The default value of `\glscompositor` is a full stop.)

`\newglossaryentry` New glossary entries are defined using the command:

```
\newglossaryentry{<label>}{<key-val list>}
```

The first argument, `<label>`, must be a unique label with which to identify this entry. The second argument, `<key-val list>`, is a `<key>=<value>` list that supplies the relevant information about this entry. There are two required fields: **name** and **description**. Available fields are listed below:

name The name of the entry (as it will appear in the glossary).

description A brief description of this term (to appear in the glossary).

descriptionplural The plural form of the description (as passed to `\glsdisplay` and `\glsdisplayfirst` by `\glspl`, `\Glspl` and `\GLSp1`). If omitted, the value is set to the same as the **description** key.

text How this entry will appear in the document text when using `\gls` (or one of its uppercase variants). If this field is omitted, the value of the **name** key is used.

first How the entry will appear in the document text the first time it is used with `\gls` (or one of its uppercase variants). If this field is omitted, the value of the **text** key is used.

plural How the entry will appear in the document text when using `\glspl` (or one of its uppercase variants). If this field is omitted, the value is obtained by appending an “s” to the value of the **text** field.

firstplural How the entry will appear in the document text the first time it is used with `\glspl` (or one of its uppercase variants). If this field is omitted, the value is obtained by appending an “s” to the value of the **first** field.

symbol This field is provided to allow the user to specify an associated symbol, but most glossary styles ignore this value. If omitted, the value is set to `\relax`.

symbolplural This is the plural form of the symbol (as passed to `\glsdisplay` and `\glsdisplayfirst` by `\glspl`, `\Glspl` and `\GLSp1`). If omitted, the value is set to the same as the **symbol** key.

sort This value indicates how `makeindex` should sort this entry. If omitted, the value is given by the **name** field. This value is equivalent to `makeindex`’s “actual” character (which is usually the at-sign @ although the `glossaries` package uses a different symbol).

type This is the glossary type to which this entry belongs. If omitted, the default glossary is assumed (**type=main**). The list of acronyms type is given by `\acronymtype` which will either be **main** or **acronym**, depending on whether the `acronym` package option was used.

Note that if the key starts with an accented letter, you must group the accented letter, otherwise it will cause a problem for `\Gls` and `\Glspl`. For example:

```
\newglossaryentry{elite}{name={{\'}e}lite},
description={select group or class}}
```

Similarly, commands such as `\ae` and `\oe` should either be placed inside a group or followed by a space. For example:

```
\newglossaryentry{oesophagus}{%
name={{\oe}sophagus},
description={canal from mouth to stomach}}
```

The following is also fine:

```
\newglossaryentry{oesophagus}{%
name={\oe sophagus},
description={canal from mouth to stomach}}
```

but don't do `\oe{sophagus}` as it will confuse `\makefirstuc` (described in [section 4](#)) and will produce `œSophagus` instead of `Œsophagus`.

`\loadglsentries` You can store all your glossary entry definitions in another file, and use:

```
\loadglsentries[⟨type⟩]{⟨filename⟩}
```

where `⟨filename⟩` is the name of the file containing all the `\newglossaryentry` commands. The optional argument `⟨type⟩` is the name of the glossary to which those entries should belong, for those entries where the `type` key has been omitted. Note that only those entries that have been used in the text will appear in the relevant glossaries.

3.3 Number lists

Each entry in the glossary has an associated *number list*. By default, these numbers refer to the pages on which that entry has been used (using any of the commands described in [subsection 3.4](#) and [subsection 3.5](#)). The number list can be suppressed using the `nonumberlist` package option, or an alternative counter can be set as the default using the `counter` package option.

3.4 Links to Glossary Entries

Once you have defined a glossary entry using `\newglossaryentry`, you can refer to that entry in the document using one of the commands listed in this section. The text which appears at that point in the document when using one of these commands is referred to as the *link text* (even if there are no hyperlinks).

`\glstextformat` The way the link text is displayed depends on `\glstextformat{⟨text⟩}`. For example, to make all link text appear in a sans-serif font, do:

```
\renewcommand*{\glstextformat}[1]{\textsf{#1}}
```

`\glslink` The command:

```
\glslink[⟨options⟩]{⟨label⟩}{⟨text⟩}
```

will place `⟨text⟩` in the document at that point and add a line into the associated

glossary file for the glossary entry given by $\langle label \rangle$. If hyperlinks are supported, $\langle text \rangle$ will be a hyperlink to the relevant line in the glossary. The optional argument $\langle options \rangle$ must be a $\langle key \rangle = \langle value \rangle$ list which can take any of the following keys:

format This specifies how to format the associated number for this entry in the glossary. This value is equivalent to the `makeindex` `encap` value, and (as with `\index`) the value needs to be the name of a command *without* the initial backslash. As with `\index`, the characters (and) can also be used to specify the beginning and ending of a number range. Again as with `\index`, the command should be the name of a command which takes an argument (which will be the associated number). Be careful not to use a declaration (such as `\bfseries`) instead of a text block command (such as `\textbf`) as the effect is not guaranteed to be localised. If you want to apply more than one style to a given entry (e.g. **bold** and *italic*) you will need to create a command that applies both formats, e.g.

```
\newcommand*{\textbfem}[1]{\textbf{\emph{#1}}}
```

and use that command.

If you are using hyperlinks and you want to change the font of the hyperlink, don't use `\hyperpage` (provided by the `hyperref` package) as the numbers may not refer to a page number. Instead, the `glossaries` package provides the following number formats:

<code>hyperrrm</code>	The number is a serif hyperlink to the relevant part of the document
<code>hypersf</code>	The number is a sans-serif hyperlink to the relevant part of the document
<code>hypertt</code>	The number is a monospaced hyperlink to the relevant part of the document
<code>hyperbf</code>	The number is a bold hyperlink to the relevant part of the document
<code>hypermd</code>	The number is a medium weight hyperlink to the relevant part of the document
<code>hyperit</code>	The number is an italic hyperlink to the relevant part of the document
<code>hypersl</code>	The number is a slanted hyperlink to the relevant part of the document
<code>hyperup</code>	The number is an upright hyperlink to the relevant part of the document
<code>hypersc</code>	The number is a small caps hyperlink to the relevant part of the document
<code>hyperemph</code>	The number is an emphasized hyperlink to the relevant part of the document

Note that if the `\hyperlink` command hasn't been defined, the `hyper $\langle xx \rangle$` formats are equivalent to the analogous `\text $\langle xx \rangle$` font commands. If you want to make a new format, you will need to define a command which takes one argument and use that; for example, if you want the associated number

in the glossary to be in a bold sans-serif font, you can define a command called, say, `\hyperbsf`:

```
\newcommand{\hyperbsf}[1]{\textbf{\hypersf{#1}}}
```

and then use `hyperbsf` as the value for the `format` key. (See also [subsection 5.13](#).)

counter This specifies which counter to use for the associated number for this glossary entry. (See also [subsection 3.3](#).)

hyper This is a boolean key which can be used to enable/disable the hyperlink to the relevant entry in the glossary. (Note that setting `hyper=true` will have no effect if `\hyperlink` has not been defined.) The default value is `hyper=true`.

`\glslink*` There is also a starred version:

```
\glslink*[\<options>]{\<label>}{\<text>}
```

which is equivalent to `\glslink`, except it sets `hyper=false`.

`\gls` The command:

```
\gls[\<options>]{\<label>}[{\<insert>}]
```

is the same as `\glslink`, except that the link text is determined from the values of the `text` and `first` keys supplied when the entry was defined using `\newglossaryentry`. There are two uppercase variants:

```
\Gls \Gls[\<options>]{\<label>}[{\<insert>}]
```

and

```
\GLS \GLS[\<options>]{\<label>}[{\<insert>}]
```

which make the first letter of the link or all the link text uppercase, respectively.

The final optional argument `\<insert>`, allows you to insert some additional text into the link text. By default, this will append `\<insert>` at the end of the link text. The first optional argument `\<options>` is the same as the optional argument to `\glslink`. As with `\glslink`, these commands also have a starred version that disable the hyperlink.

There are also analogous plural forms:

```
\glspl \glspl[\<options>]{\<label>}[{\<insert>}]
```

```
\Glspl \Glspl[\<options>]{\<label>}[{\<insert>}]
```

```
\GLSpl \GLSpl[\<options>]{\<label>}[{\<insert>}]
```

These determine the link text from the `plural` and `firstplural` keys supplied when the entry was first defined. As before, these commands also have a starred version that disable the hyperlink.

`\gls{text}` The command:

`\glstext[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

is similar to `\gls` except that it always uses the value of the `text` key and does not mark the entry as having been used.

There are also analogous commands:

`\Glstext` `\Glstext[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

`\GLStext` `\GLStext[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

As before, these commands also have a starred version that disable the hyperlink.

`\glsfirst` The command:

`\glsfirst[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

is similar to `\gls` except that it always uses the value of the `first` key and does not mark the entry as having been used.

There are also analogous commands:

`\Glsfirst` `\Glsfirst[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

`\GLSfirst` `\GLSfirst[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

As before, these commands also have a starred version that disable the hyperlink.

`\glsplural` The command:

`\glsplural[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

is similar to `\gls` except that it always uses the value of the `plural` key and does not mark the entry as having been used.

There are also analogous commands:

`\Glsplural` `\Glsplural[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

`\GLSplural` `\GLSplural[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

As before, these commands also have a starred version that disable the hyperlink.

`\glsfirstplural` The command:

`\glsfirstplural[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

is similar to `\gls` except that it always uses the value of the `firstplural` key and does not mark the entry as having been used.

There are also analogous commands:

`\Glsfirstplural` `\Glsfirstplural[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

`\GLSfirstplural` `\GLSfirstplural[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

As before, these commands also have a starred version that disable the hyperlink.

`\glsname` The command:

`\glsname[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

is similar to `\gls` except that it always uses the value of the `name` key and does not mark the entry as having been used. Note: if you want to use this command and the `name` key contains commands, you will have to disable the **sanitization** of the `name` key and protect fragile commands.

There are also analogous commands:

`\Glsname` `\Glsname[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

`\GLSname` `\GLSname[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

As before, these commands also have a starred version that disable the hyperlink.

`\glssymbol` The command:

`\glssymbol[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

is similar to `\gls` except that it always uses the value of the `symbol` key and does not mark the entry as having been used. Note: if you want to use this command and the `symbol` key contains commands, you will have to disable the **sanitization** of the `symbol` key and protect fragile commands.

There are also analogous commands:

`\Glssymbol` `\Glssymbol[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

`\GLSsymbol` `\GLSsymbol[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

As before, these commands also have a starred version that disable the hyperlink.

`\glsdesc` The command:

`\glsdesc[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

is similar to `\gls` except that it always uses the value of the `description` key and does not mark the entry as having been used. Note: if you want to use this command and the `description` key contains commands, you will have to disable the **sanitization** of the `description` key and protect fragile commands.

There are also analogous commands:

`\Glsdesc` `\Glsdesc[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

`\GLSdesc` `\GLSdesc[⟨options⟩]{⟨text⟩}[⟨insert⟩]`

As before, these commands also have a starred version that disable the hyperlink.

3.4.1 Changing the format of the link text

The format of the link text for `\gls`, `\glsp1` and their uppercase variants is governed by two commands: `\glsdisplayfirst`, which is used the first time a glossary entry is used in the text and `\glsdisplay`, which is used subsequently. Both commands take four arguments: the first is either the singular or plural form given by the `text`, `plural`, `first` or `firstplural` keys (used when the term was defined) depending on context; the second argument is the term's description (as supplied by the `description` key); the third argument is the symbol associated with the term (as supplied by the `symbol` key) and the fourth argument is the additional text

supplied in the final optional argument to `\gls` or `\glspl` (or their uppercase variants). The default definitions of `\glsdisplay` and `\glsdisplayfirst` simply print the first argument immediately followed by the fourth argument. The remaining arguments are ignored.

For example, suppose you want a glossary of measurements and units, you can use the `symbol` key to store the unit:

```
\newglossaryentry{distance}{name=distance,
description={The length between two points},
symbol={km}}
```

and now suppose you want `\gls{distance}` to produce “distance (km)” on first use, then you can redefine `\glsdisplayfirst` as follows:

```
\renewcommand{\glsdisplayfirst}[4]{#1#4 (#3)}
```

Note that the additional text is placed after `#1`, so `\gls{distance}[s]` will produce “distance’s (km)” rather than “distance (km)’s” which looks a bit odd (even though it may be in the context of “the distance (km) is measured between the two points” — but in this instance it may be better not to use a contraction).

Note also that all of the link text will be formatted according to `\glstextformat` (described earlier). So if you do, say:

```
\renewcommand{\glstextformat}[1]{\textbf{#1}}
\renewcommand{\glsdisplayfirst}[4]{#1#4 (#3)}
```

then `\gls{distance}` will produce “**distance (km)**”.

If you have multiple glossaries, changing `\glsdisplayfirst` and `\glsdisplay` will change the way entries for all of the glossaries appear when using commands `\gls`, `\glspl` and their uppercase variants. If you only want the change to affect entries for a given glossary, then you need to use `\defglsdisplay` and `\defglsdisplayfirst` instead of redefining `\glsdisplay` and `\glsdisplayfirst`.

`\defglsdisplay`
`\defglsdisplayfirst`

Both `\defglsdisplay` and `\defglsdisplayfirst` take two arguments: the first (which is optional) is the glossary name³ and the second is how the term should be displayed when it is invoked using commands `\gls`, `\glspl` and their uppercase variants. This is similar to the way `\glsdisplayfirst` was redefined above.

For example, suppose you have created a new glossary called `notation` and you want to change the way the entry is displayed on first use so that it includes the symbol, you can do:

```
\defglsdisplayfirst[notation]{#1#4 (denoted #3)}
```

Now suppose you have defined an entry as follows:

```
\newglossaryentry{set}{type=notation,
name=set,
description={A collection of objects},
symbol={$$$},
}
```

³main for the main (default) glossary, `\acronymtype` for the list of acronyms, or the name supplied in the first mandatory argument to `\newglossary` for additional glossaries.

The first time you reference this entry using `\gls` it will be displayed as: “set (denoted S)” (similarly for `\glspl` and the uppercase variants).

Remember that if you use the `symbol` key, you need to use a glossary style that displays the symbol, as many of the styles ignore it. In addition, if you want either the description or symbol to appear in the link text, you will have to disable the [sanitization](#) of these keys and protect fragile commands.

3.4.2 Enabling and disabling hyperlinks to glossary entries

If you load the `hyperref` or `html` packages prior to loading the `glossaries` package, commands such as `\glslink` and `\gls`, described above, will automatically have hyperlinks to the relevant glossary entry, unless the `hyper` option has been set to `false`. You can disable or enable links using:

`\glsdisablehyper` `\glsdisablehyper`

and

`\glsenablehyper` `\glsenablehyper`

respectively. The effect can be localised by placing the commands within a group. Note that you should only use `\glsenablehyper` if the commands `\hyperlink` and `\hypertarget` have been defined (for example, by the `hyperref` package).

3.5 Adding an entry to the glossary without generating text

`\glsadd` It is possible to add a line in the glossary file without generating any text at that point in the document using:

`\glsadd[<options>]{<label>}`

This is similar to `\glslink`, only it doesn’t produce any text (so therefore, there is no `hyper` key available in *<options>* but all the other options that can be used with `\glslink` can be passed to `\glsadd`).

`\glsaddall` To add all entries that have been defined, use:

`\glsaddall[<glossary list>]`

If there are multiple glossaries, you can specify to add only those entries which belong to the glossaries listed in *<glossary list>* (which must be a comma separated list of glossary names). For example:

`\glsaddall[notation]`

will add all the entries that have been defined for the glossary labelled “notation”.

3.6 Displaying a glossary

`\printglossaries` The command `\printglossaries` will display all the defined glossaries. Note that no glossaries will appear until you have either used the Perl script `makeglossaries` or have directly used `makeindex` (as described in [subsection 1.1](#)). If the glossary still does not appear after you re- \LaTeX your document, check the `makeindex` log

files to see if there is a problem. Remember that you also need to use the command `\makeglossaries` in the preamble to enable the glossaries.

`\printglossary` An individual glossary is displayed using:

```
\printglossary[\langle options \rangle]
```

where *\langle options \rangle* is a *\langle key \rangle*=*\langle value \rangle* list of options. The following keys are available:

type The value of this key specifies which glossary to print. If omitted, the default glossary is assumed. For example, to print the list of acronyms:

```
\printglossary[type=\acronymtype]
```

title This is the glossary's title (overriding the title specified when the glossary was defined).

toctitle This is the title to use for the table of contents (if the `toc` package option has been used). If omitted, the glossary title is used.

style This specifies which glossary style to use for this glossary, overriding the effect of the `style` option or `\glossarystyle`.

`\glossarypreamble` Information can be added to the start of the glossary by redefining `\glossarypreamble`. For example:

```
\renewcommand{\glossarypreamble}{Numbers in italic indicate
primary definitions.}
```

This needs to be done before the glossary is displayed using `\printglossaries` or `\printglossary`. Note that if you want a different preamble for each glossary, you will need to use a separate `\printglossary` for each glossary and change the definition of `\glossarypreamble` between each glossary. For example:

```
\renewcommand{\glossarypreamble}{Numbers in italic indicate
primary definitions.}
\printglossary
\renewcommand{\glossarypreamble}{}
\printglossary[type=acronym]
```

Alternatively, you can do something like:

```
\renewcommand{\glossarypreamble}{Numbers in italic indicate
primary definitions.\gdef\glossarypreamble{}}
\printglossaries
```

which will print the preamble text for the first glossary and change the preamble to do nothing for subsequent glossaries. (Note that `\gdef` is required as the glossary is placed within a group.)

`\glossarypostamble` There is an analogous command called `\glossarypostamble` which is placed at the end of each glossary.

3.6.1 Changing the way the entry name appears in the glossary

`\glsnamefont` Within each glossary, each entry name is formatted according to `\glsnamefont` which takes one argument: the entry name. This command is always used regardless of the glossary style. By default, `\glsnamefont` simply displays its argument in whatever the surrounding font happens to be. This means that in the list styles the name will appear in bold, since the name is placed in the optional argument of `\item`, whereas in the tabular styles the name will appear in the normal font.

For example, suppose you want all the entry names to appear in medium weight small caps, then you can do:

```
\renewcommand{\glsnamefont}[1]{\textsc{\mdseries #1}}
```

3.7 Defining New Glossaries

`\newglossary` A new glossary can be defined using:

```
\newglossary[⟨log-ext⟩]{⟨name⟩}{⟨in-ext⟩}{⟨out-ext⟩}{⟨title⟩}[⟨counter⟩]
```

where `⟨name⟩` is label to assign to this glossary. The arguments `⟨in-ext⟩` and `⟨out-ext⟩` specify the extensions to give to the input and output files for that glossary, `⟨title⟩` is the default title for this new glossary and the final optional argument `⟨counter⟩` specifies which counter to use for the associated number lists (see also [subsection 3.3](#).) The first optional argument specifies the extension for the `makeindex` transcript file (this information is only used by `makeglossaries` which picks up the information from the auxiliary file.)

Note that the main (default) glossary is automatically created as:

```
\newglossary{main}{gls}{glo}{\glossaryname}
```

so it can be identified by the label `main`. Using the `acronym` package option is equivalent to:

```
\newglossary[alg]{acronym}{acr}{acn}{\acronymname}
```

so it can be identified by the label `acronym`. If you are not sure whether the `acronym` option has been used, you can identify the list of acronyms by the command `\acronymtype` which is set to `acronym`, if the `acronym` option has been used,

otherwise it is set to `main`.

3.8 Acronyms

`\newacronym` As you may have noticed in [subsection 3.2](#), when you specify a new entry, you can specify alternate text to use when the term is first used in the document. This provides a useful means to define acronyms. The `glossaries` package defines the command:

```
\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}
```

This is equivalent to:

```
\newglossaryentry{⟨label⟩}{type=\acronymtype,
name={⟨abbrv⟩},
description={⟨long⟩},
```

```

text={⟨abbrv⟩},
first={⟨long⟩ ⟨⟨abbrv⟩⟩},
plural={⟨abbrv⟩s},
firstplural={⟨long⟩s ⟨⟨abbrv⟩s⟩},
⟨key-val list⟩

```

As mentioned in the previous section, the command `\acronymtype` is the name of the glossary in which the acronyms should appear. If the `acronym` package option has been used, this will be `acronym`, otherwise it will be `main`. The acronyms can then be used in exactly the same way as any other glossary entry. For example, the following defines the acronym IDN:

```
\newacronym{idn}{IDN}{identification number}
```

This is equivalent to:

```

\newglossaryentry{idn}{type=\acronymtype,
name={IDN},
description={identification number},
text={IDN},
first={identification number (IDN)},
plural={IDNs},
firstplural={identification numbers (IDNs)}}

```

so `\gls{idn}` will produce “identification number (IDN)” on first use and “IDN” on subsequent uses.

The `glossaries` package has options that change the definition of `\newacronym` for common acronym formats. These options also change the way the link text is displayed for the acronyms.

Table 1 lists the package options and how the keys are used to store `⟨long⟩` (the long form) and `⟨abbrv⟩` (the short form). Note that the `smallcaps` option redefines `\acronymfont` so that it sets its argument in small captials otherwise `\acronymfont` simply displays its argument in the surrounding font. Note also that if none of the package options `smallcaps`, `description` or `footnote` are used, `\acronymfont` is not used, so changing the definition of `\acronymfont` will have no effect under such circumstances.

Table 1: Package options governing acronyms and how the information is stored in the glossary keys

Package Option	first	text	description	symbol
description,footnote	⟨abbrv⟩	⟨abbrv⟩	user supplied	⟨long⟩
description,dua	⟨long⟩	⟨long⟩	user supplied	⟨abbrv⟩
description	⟨long⟩	⟨abbrv⟩	user supplied	⟨abbrv⟩
footnote	⟨abbrv⟩	⟨abbrv⟩	⟨long⟩	
smallcaps	⟨long⟩	⟨abbrv⟩	⟨long⟩	⟨abbrv⟩
dua	⟨long⟩	⟨long⟩	⟨long⟩	⟨abbrv⟩
None of the above	⟨long⟩ ⟨⟨abbrv⟩⟩	⟨abbrv⟩	⟨long⟩	

In case you can’t remember which key stores the long or short forms (or their plurals) the `glossaries` package provides the commands:

<code>\glsshortkey</code>	• <code>\glsshortkey</code> The key used to store the short form.
<code>\glsshortpluralkey</code>	• <code>\glsshortpluralkey</code> The key used to store the plural version of the short form.
<code>\glslongkey</code>	• <code>\glslongkey</code> The key used to store the long form.
<code>\glslongpluralkey</code>	• <code>\glslongpluralkey</code> The key used to store the plural version of the long form.

These can be used in the optional argument of `\newacronym` to override the defaults. For example:

```
\newacronym[\glslongpluralkey={diagonal matrices}]{dm}{DM}{diagonal matrix}
```

If the first use uses the plural form, `\glspl{dm}` will display: diagonal matrices (DMs).

Each of the package options `smallcaps`, `footnote` and `description` use `\defglsdisplay` and `\defglsdisplayfirst` (described in [subsubsection 3.4.1](#)) to change the way the link text is displayed.

description,footnote

When these two options are used together, the first use displays the entry as:

```
\acronymfont{\langle abbrev \rangle \langle insert \rangle} \footnote{\langle long \rangle}
```

while subsequent use displays the entry as:

```
\acronymfont{\langle abbrev \rangle \langle insert \rangle}
```

where `\langle insert \rangle` indicates the text supplied in the final optional argument to `\gls`, `\glspl` or their uppercase variants.

Note also that when these two package options are used (in the given order), the `glossaries` package additionally implements the `sanitize` option using `sanitize={description=false,symbol=false}`, so remember to protect fragile commands when defining acronyms.

dua

The `dua` package option always displays the expanded form and so may not be used with `footnote` or `smallcaps`. Both first use and subsequent use displays the entry in the form:

```
\langle long \rangle \langle insert \rangle
```

If the `description` option is also used, the `name` key is set to the long form, otherwise the `name` key is set to the short form and the `description` key is set to the long form.

description

This option displays the entry on first use as:

`<long><insert> (\acronymfont{<abbrv>})`

while subsequent use displays the entry as:

`\acronymfont{<abbrv>}<insert>`

Note also that if this package option is used, the `glossaries` package additionally implements the option `sanitize={symbol=false}`, so remember to protect fragile commands when defining acronyms.

footnote

This option displays the entry on first use as:

`\acronymfont{<abbrv>}<insert>\footnote{<long>}`

while subsequent use displays the entry as:

`\acronymfont{<abbrv>}<insert>`

Note also that if this package option is used, the `glossaries` package additionally implements the option `sanitize={description=false}`, so remember to protect fragile commands when defining acronyms.

smallcaps

If neither the `footnote` nor `description` options have been set, this option displays the entry on first use as:

`<long><insert> (\acronymfont{<abbrv>})`

while subsequent use displays the entry as:

`\acronymfont{<abbrv>}<insert>`

where `\acronymfont` is set to `\textsc`.

Note also that if this package option is used, the `glossaries` package additionally implements the option `sanitize={symbol=false}`, so remember to protect fragile commands when defining acronyms.

None of the above

If none of the package options `smallcaps`, `footnote` or `description` are used, then on first use the entry is displayed as:

`<long> (<abbrv>)<insert>`

while subsequent use displays the entry as:

`\acronymfont{<abbrv>}<insert>`

`<abbrv><insert>`

Recall from [subsection 3.4](#) that you can access the values of individual keys using commands like `\glstext`, so it is possible to use these commands to print just the long form or just the abbreviation without affecting the flag that determines

whether the entry has been used. However the keys that store the long and short form vary depending on the acronym style, so the `glossaries` package provides commands that are set according to the package options. These are as follows:

```
\acrshort  \acrshort[⟨options⟩]{⟨label⟩}[⟨insert⟩]
\Acrshort  \ACRshort[⟨options⟩]{⟨label⟩}[⟨insert⟩]
\ACRshort  \ACRshort[⟨options⟩]{⟨label⟩}[⟨insert⟩]
```

Print the abbreviated version with a hyperlink (if necessary) to the relevant entry in the glossary. This is usually equivalent to `\glstext` (or its uppercase variants) but may additionally put the link text within the argument to `\acronymfont`.

```
\acrlong   \acrlong[⟨options⟩]{⟨label⟩}[⟨insert⟩]
\Acrlong   \ACRlong[⟨options⟩]{⟨label⟩}[⟨insert⟩]
\ACRlong   \ACRlong[⟨options⟩]{⟨label⟩}[⟨insert⟩]
```

Print the long version with a hyperlink (if necessary) to the relevant entry in the glossary. This is may be equivalent to `\glstdesc`, `\glssymbol` or `\glsfirst` (or their uppercase variants), depending on package options.

```
\acrfull   \acrfull[⟨options⟩]{⟨label⟩}[⟨insert⟩]
\Acrfull   \ACRfull[⟨options⟩]{⟨label⟩}[⟨insert⟩]
\ACRfull   \ACRfull[⟨options⟩]{⟨label⟩}[⟨insert⟩]
```

Print the long version followed by the abbreviation in brackets with a hyperlink (if necessary) to the relevant entry in the glossary.

Note that if you change the definition of `\newacronym`, you may additionally need to change the above commands as well as the changing way the text is displayed using `\defglstdisplay` and `\defglstdisplayfirst`.

3.9 Unsetting and resetting entry flags

When using commands such as `\gls` it is possible that you may want to use the value given by the `first` key, even though you have already used the glossary entry. Conversely, you may want to use the value given by the `text` key, even though you haven't used the glossary entry. The former can be achieved by one of the following commands:

```
\glsreset  \glsreset{⟨label⟩}
\glslocalreset \glslocalreset{⟨label⟩}
```

while the latter can be achieved by one of the following commands:

```
\glsunset  \glsunset{⟨label⟩}
\glslocalunset \glslocalunset{⟨label⟩}
```

You can determine whether an entry has been used using:

```
\ifglsused \ifglsused{⟨label⟩}{⟨true part⟩}{⟨false part⟩}
```

where `⟨label⟩` is the label of the required entry. If the entry has been used, `⟨true part⟩` will be done, otherwise `⟨false part⟩` will be done.

3.10 Glossary Styles

The `glossaries` package comes with some pre-defined glossary styles. These are as follows:

list The `list` style uses the `description` environment. The entry name is placed in the optional argument of the `\item` command (so it will appear in bold by default). The description follows, and then the associated number list for that entry.

listgroup The `listgroup` style is like `list` but the glossary groups have headings.

listhypergroup The `listhypergroup` style is like `listgroup` but has a set of links to the glossary groups.

altlist The `altlist` style is like `list` but the description is placed on the following line.

altlistgroup The `altlistgroup` style is like `altlist` but the glossary groups have headings.

altlisthypergroup The `altlisthypergroup` style is like `altlistgroup` but has a set of links to the glossary groups.

listdotted This style uses the `description` environment. Each entry starts with `\item[]`, followed by the name followed by a dotted line, followed by the description. Note that this style ignores both the number list and the symbol. The length `\glslistdottedwidth` governs where the description should start.⁴

`\glslistdottedwidth`

long The `long` style uses the `longtable` environment (defined by the `longtable` package). It has two columns: the first column contains the entry's name and the second column contains the description followed by the number list.

longborder The `longborder` style is like `long` but has horizontal and vertical lines around it.

longheader The `longheader` style is like `long` but has a header row.

longheaderborder The `longheaderborder` style is like `longheader` but has horizontal and vertical lines around it.

long3col The `long3col` style is like `long` but has three columns. The first column contains the entry's name, the second column contains the description and the third column contains the number list.

long3colborder The `long3colborder` style is like the `long3col` style but has horizontal and vertical lines around it.

long3colheader The `long3colheader` style is like `long3col` but has a header row.

long3colheaderborder The `long3colheaderborder` style is like `long3colheader` but has horizontal and vertical lines around it.

⁴This style was supplied by Axel Menzel.

long4col The `long4col` style is like `long3col` but has an additional column in which the entry's associated symbol appears.

long4colborder The `long4colborder` style is like the `long4col` style but has horizontal and vertical lines around it.

long4colheader The `long4colheader` style is like `long4col` but has a header row.

long4colheaderborder The `long4colheaderborder` style is like `long4colheader` but has horizontal and vertical lines around it.

super The `super` style uses the `supertabular` environment (defined by the `supertabular` package). It has two columns: the first column contains the entry's name and the second column contains the description followed by the number list.

superborder The `superborder` style is like `super` but has horizontal and vertical lines around it.

superheader The `superheader` style is like `super` but has a header row.

superheaderborder The `superheaderborder` style is like `superheader` but has horizontal and vertical lines around it.

super3col The `super3col` style is like `super` but has three columns. The first column contains the entry's name, the second column contains the description and the third column contains the .

super3colborder The `super3colborder` style is like the `super3col` style but has horizontal and vertical lines around it.

super3colheader The `super3colheader` style is like `super3col` but has a header row.

super3colheaderborder The `super3colheaderborder` style is like `super3colheader` but has horizontal and vertical lines around it.

super4col The `super4col` style is like `super3col` but has an additional column in which the entry's associated symbol appears.

super4colborder The `super4colborder` style is like the `super4col` style but has horizontal and vertical lines around it.

super4colheader The `super4colheader` style is like `super4col` but has a header row.

super4colheaderborder The `super4colheaderborder` style is like `super4colheader` but has horizontal and vertical lines around it.

The glossary style can be set using the `style` package option or using the `style` key in the optional argument to `\printglossary` or using the command:

```
\glossarystyle \glossarystyle{<style-name>}
```

All the styles except for the three- and four-column styles and the `listdotted` style use the command `\glspostdescription` after the description. This simply

displays a full stop by default. To eliminate this full stop (or replace it with something else, say a comma), you will need to redefine `\glspostdescription` before the glossary is displayed.

3.11 Defining your own glossary style

If the predefined styles don't fit your requirements, you can define your own style using:

`\newglossarystyle`

`\newglossarystyle{<name>}{<definitions>}`

where `<name>` is the name of the new glossary style (to be used in `\glossarystyle`). The second argument `<definitions>`, needs to redefine all of the following:

`theglossary` `theglossary`

This environment defines how the main body of the glossary should be typeset. Note that this does not include the section heading, the glossary preamble (defined by `\glossarypreamble`) or the glossary postamble (defined by `\glossarypostamble`.) For example, the list style uses the `description` environment, so the `theglossary` environment is simply redefined to begin and end the `description` environment.

`\glossaryheader` `\glossaryheader`

This macro indicates what to do at the start of the main body of the glossary. Note that this is not the same as `\glossarypreamble`, which should not be affected by changes in the glossary style. The list glossary style redefines `\glossaryheader` to do nothing, whereas the `longheader` glossary style redefines `\glossaryheader` to do a header row.

`\glsgroupheading` `\glsgroupheading{<label>}`

This macro indicates what to do at the start of each logical block within the main body of the glossary. The glossary is sub-divided into twenty-eight logical blocks that are determined by the first character of the `sort` key (or `name` key if the `sort` key is omitted). The sub-divisions are in the following order: symbols, numbers, A, ..., Z. Note that the argument to `\glsgroupheading` is a label *not* the group title. The group title can be obtained via `\glsgroupheading{<label>}`, and a navigation hypertarget can be created using `\glsgroupheading{<label>}`. Most of the predefined glossary styles redefine `\glsgroupheading` to simply ignore its argument. The `listhypergroup` style redefines `\glsgroupheading` as follows:

```
\renewcommand*{\glsgroupheading}[1]{%
\item[\glsgroupheading{##1}]{\glsgroupheading{##1}}}
```

See also `\glsgroupskip` below. (Note that command definitions within `\newglossarystyle` must use `##1` etc instead of `#1` etc.)

`\glsgroupskip` `\glsgroupskip`

This macro determines what to do after one logical group but before the header for the next logical group. The list glossary style simply redefines `\glsgroupskip` to be `\indexspace`.

`\glossaryentryfield` `\glossaryentryfield{<label>}{<formatted name>}{<description>}{<symbol>}{<number list>}`

This macro indicates what to do for a given glossary entry. Note that `<formatted name>` will always be in the form `\glsnamefont{<name>}`. This allows the user to set a given font for the entry name, regardless of the glossary style used. Note that `<label>` is the label used when the glossary entry was defined via either `\newglossaryentry` or `\newacronym`. Each time you use a glossary entry it creates a link⁵ using `\@glslink{<label>}{<text>}` with the label `glo:<label>`. Your new glossary style must therefore redefine `\glossaryentryfield` so that it uses `\@glslink{glo:<label>}{<text>}` to ensure the hyperlinks function correctly.⁶ For example, the list style defines `\glossaryentryfield` as follows:

```
\renewcommand*{\glossaryentryfield}[5]{%
\item[\@glslink{glo:##1}{##2}] ##3\glspostdescription\space ##5}
```

Note also that `<number list>` will always be of the form

```
\glossaryentrynumbers{\relax
\setentrycounter{<counter name>}\glsnumberformat{<number(s)>}}
```

where `<number(s)>` may contain `\delimN` (to delimit individual numbers) and/or `\delimR` (to indicate a range of numbers). There may be multiple occurrences of `\setentrycounter{<counter name>}\glsnumberformat{<number(s)>}`, but note that the entire number list is enclosed within the argument to `\glossaryentrynumbers`. The user can redefine this to change the way the entire number list is formatted, regardless of the glossary style. However the most common use of `\glossaryentrynumbers` is to provide a means of suppressing the number list altogether. (In fact, the `nonumberlist` option redefines `\glossaryentrynumbers` to ignore its argument.) Therefore, when you define a new glossary style, you don't need to worry about whether the user has specified the `nonumberlist` package option.

3.11.1 Example: creating a completely new style

If you want a completely new style, you will need to redefine all of the commands and environment listed above. You also need to take care when using internal commands (commands whose name contain the `@` symbol). These should either be used in a `.sty` file or must be placed within `\makeatletter` and `\makeatother`.

For example, suppose you want each entry to start with a bullet point. This means that the glossary should be placed in the `itemize` environment, so `\theglossary` should start and end that environment. Let's also suppose that you don't want anything between the glossary groups (so `\glsgroupheading` and `\glsgroupskip` should do nothing) and suppose you don't want anything to appear immediately after `\begin{theglossary}` (so `\glossaryheader` should do nothing). In addition, let's suppose the symbol should appear in brackets after the name, followed

⁵if the document doesn't have hyperlinks enabled `\@glslink` ignores the label and simply typesets the text.

⁶again, if the document doesn't support hyperlinks, `\@glslink` will ignore the label, and just typeset the text.

by the description and last of all the number list should appear within square brackets at the end. Then you can create this new glossary style, called, say, `mylist`, as follows:

```
\newglossarystyle{mylist}{%
% put the glossary in the itemize environment:
\renewenvironment{theglossary}{\begin{itemize}}{\end{itemize}}%
% have nothing after \begin{theglossary}:
\renewcommand*{\glossaryheader}{}%
% have nothing between glossary groups:
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glsgroupskip}{}%
% set how each entry should appear:
\renewcommand*{\glossaryentryfield}[5]{%
\item % bullet point
\@glstarget{glo:##1}{##2}% the entry name
\space (##4)% the symbol in brackets
\space ##3% the description
\space [##5]% the number list in square brackets
}%
}
```

3.11.2 Example: creating a new glossary style based on an existing style

If you want to define a new style that is a slightly modified version of an existing style, you can use `\glossarystyle` within the second argument of `\newglossarystyle` followed by whatever alterations you require. For example, suppose you want a style like the `list` style but you don't want the extra vertical space created by `\indexspace` between groups, then you can create a new glossary style called, say, `mylist` as follows:

```
\newglossarystyle{mylist}{%
\glossarystyle{list}% base this style on the list style
\renewcommand*{\glsgroupskip}{}% make nothing happen between groups
}
```

4 Mfirstuc Package

The `glossaries` package is supplied with the package `mfirstuc` which provides the command:

```
\makefirstuc \makefirstuc{<stuff>}
```

which makes the first object of `<stuff>` uppercase unless `<stuff>` starts with a control sequence followed by a group, in which case the first object in the group is converted to uppercase. Examples:

- `\makefirstuc{abc}` produces `Abc`
- `\makefirstuc{\emph{abc}}` produces `Abc` (`\MakeUppercase` has been applied to the letter “a” rather than `\emph.`)
- `\makefirstuc{\{\'a\}bc}` produces `Ábc`

- `\makefirstuc{\ae bc}` produces $\text{\AE}bc$
- `\makefirstuc{\{\ae\}bc}` produces $\text{\AE}bc$

Note however that `\makefirstuc{\ae{}}bc` produces $\text{\ae}Bc$ since the empty group following `\ae` is considered to be the argument of `\ae` which is in fact not the case.

5 Documented Code

5.1 Package Definition

This package requires L^AT_EX 2_ε.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2008/03/08 v1.12 (NLCT)]
```

Required packages:

```
3 \RequirePackage{ifthen}
4 \RequirePackage{xkeyval}[2006/11/18]
5 \RequirePackage{mfirstuc}
```

Need to use `\new@ifnextchar` instead of `\@ifnextchar` in commands that have a final optional argument (such as `\gls`) so require `amsgen`. Thanks to Morten Høgholm for suggesting this.

```
6 \RequirePackage{amsgen}
```

5.2 Package Options

The `toc` package option will add the glossaries to the table of contents. This is a boolean key, if the value is omitted it is taken to be true.

```
7 \define@boolkey{glossaries.sty}[gls]{toc}[true]{}%
```

The `numberline` package option adds `\numberline` to `\addcontentsline`. Note that this option only has an effect if used in with `toc=true`.

```
8 \define@boolkey{glossaries.sty}[gls]{numberline}[true]{}%
```

The sectional unit used to start the glossary is stored in `\@@glossarysec`. If chapters are defined, this is initialised to `chapter`, otherwise it is initialised to `section`.

```
\@@glossarysec
```

```
9 \ifundefined{chapter}{\newcommand*{\@@glossarysec}{section}}{%
10 \newcommand*{\@@glossarysec}{chapter}}
```

The `section` key can be used to set the sectional unit. If no unit is specified, use `section` as the default. The starred form of the named sectional unit will be used. If you want some other way to start the glossary section (e.g. a numbered section) you will have to redefined `\glossarysection`.

```
11 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
12 subsection,subsubsection,paragraph,subparagraph}[section]{}%
13 \renewcommand*{\@@glossarysec}{#1}}
```

Determine whether or not to use numbered sections.

`\@glossarysecstar`

```
14 \newcommand*\@glossarysecstar{*}
```

`\@glossaryseclabel`

```
15 \newcommand*\@glossaryseclabel{}

16 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
17 false,nolabel,autolabel}[nolabel]{%
18 \ifcase\nr\relax
19 \renewcommand*\@glossarysecstar{*}%
20 \renewcommand*\@glossaryseclabel{}%
21 \or
22 \renewcommand*\@glossarysecstar{}%
23 \renewcommand*\@glossaryseclabel{}%
24 \or
25 \renewcommand*\@glossarysecstar{}%
26 \renewcommand*\@glossaryseclabel{\label{\@glo@type}}%
27 \fi}
```

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The `list` style is defined in the accompanying `glossary-list` package described in [subsection 5.16](#).)

`\@glossary@default@style`

```
28 \newcommand*\@glossary@default@style{list}
```

The default glossary style can be changed using the `style` package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the `style` key to set a style that is defined in another package. This package comes with some predefined styles that are defined in [subsection 5.16](#).

```
29 \define@key{glossaries.sty}{style}{%
30 \renewcommand*\@glossary@default@style{#1}}
```

Each entry within a given glossary will have an associated number list. By default, this refers to the page numbers on which that entry has been used, but it can also refer to any counter used in the document (such as the section or equation counters). The default number list format displays the number list “as is”:

`\glossaryentrynumbers`

```
31 \newcommand*\glossaryentrynumbers[1]{#1}
```

Note that the entire number list for a given entry will be passed to `\glossaryentrynumbers` so any font changes will also be applied to the delimiters. The `nonumberlist` package option suppresses the number lists (this simply redefines `\glossaryentrynumbers` to ignore its argument.)

```
32 \DeclareOptionX{nonumberlist}{%
33 \renewcommand*\glossaryentrynumbers[1]{}}
```

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the `type` key in a key-value list.) This was mainly done so that `\loadglsentries` can temporarily change `\glsdefaulttype` while it loads a file containing new glossary entries (see [subsection 5.8](#)).

`\glsdefaulttype`

```
34 \newcommand{\glsdefaulttype}{main}
```

Keep track of which glossary the acronyms are in. This is initialised to `\glsdefaulttype`, but is changed by the `acronym` package option.

`\acronymtype`

```
35 \newcommand{\acronymtype}{\glsdefaulttype}
```

The `acronym` option sets an associated conditional which is used in [subsection 5.14](#) to determine whether or not to define a separate glossary for acronyms.

```
36 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{}%
```

The default counter associated with the numbers in the glossary is stored in `\glscounter`. This is initialised to the page counter. This is used as the default counter when a new glossary is defined, unless a different counter is specified in the optional argument to `\newglossary` (see [subsection 5.5](#)).

`\glscounter`

```
37 \newcommand{\glscounter}{page}
```

The `counter` option changes the default counter. (This just redefines `\glscounter`.)

```
38 \define@key{glossaries.sty}{counter}{%
```

```
39 \renewcommand*{\glscounter}{#1}}%
```

The glossary keys whose values are written to another file (i.e. `sort`, `name`, `description` and `symbol`) need to be sanitized, otherwise fragile commands would not be able to be used in `\newglossaryentry`. However, strange results will occur if you then use those fields in the document. As these fields are not normally used in the document, but are by default only used in the glossary, the default is to sanitize them. If however you want to use these values in the document (either by redefining commands like `\glsdisplay` or by using commands like `\glsentrydesc`) you will have to switch off the sanitization using the `sanitize` package option, but you will then have to use `\protect` to protect fragile commands when defining new glossary entries. The `sanitize` option takes a key-value list as its value, which can be used to switch individual values on and off. For example:

```
\usepackage[sanitize={description,name,symbol=false}]{glossaries}
```

will switch off the sanitization for the `symbol` key, but switch it on for the `description` and `name` keys. This would mean that you can use fragile commands in the `description` and `name` when defining a new glossary entry, but not for the `symbol`.

The default values are defined as:

`\@gls@sanitizedesc`

```
40 \newcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}
```

`\@gls@sanitizename`

```
41 \newcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}
```

`\@gls@sanitizesymbol`

```
42 \newcommand*{\@gls@sanitizesymbol}{\@onelevel@sanitize\@glo@symbol}
```

(There is no equivalent for the `sort` key, since that is only provided for the benefit of `makeindex`, and so will always be sanitized.)

Before defining the `sanitize` package option, The key-value list for the `sanitize` value needs to be defined. These are all boolean keys. If they are not given a value, assume `true`.

Firstly the `description`. If set, it will redefine `\@gls@sanitizedesc` to use `\@onelevel@sanitize`, otherwise `\@gls@sanitizedesc` will do nothing.

```
43 \define@boolkey[gls]{sanitize}{description}[true]{%
44 \ifgls@sanitize@description
45   \renewcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}%
46 \else
47   \renewcommand*{\@gls@sanitizedesc}{}%
48 \fi
49 }
```

Similarly for the `name` key:

```
50 \define@boolkey[gls]{sanitize}{name}[true]{%
51 \ifgls@sanitize@name
52   \renewcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}%
53 \else
54   \renewcommand*{\@gls@sanitizename}{}%
55 \fi}
```

and for the `symbol` key:

```
56 \define@boolkey[gls]{sanitize}{symbol}[true]{%
57 \ifgls@sanitize@symbol
58   \renewcommand*{\@gls@sanitizesymbol}{%
59 \@onelevel@sanitize\@glo@symbol}%
60 \else
61   \renewcommand*{\@gls@sanitizesymbol}{}%
62 \fi}
```

Now define the `sanitize` option. It can either take a key-val list as its value, or it can take the keyword `none`, which is equivalent to `description=false`, `symbol=false`, `name=false`:

```
63 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,
64 name=true]{%
65 \ifthenelse{\equal{#1}{none}}{%
66 \renewcommand*{\@gls@sanitizedesc}{}%
67 \renewcommand*{\@gls@sanitizename}{}%
68 \renewcommand*{\@gls@sanitizesymbol}{}%
69 }{\setkeys{gls}{sanitize}{#1}}%
70 }
```

Define `translate` option. If false don't set up multi-lingual support.

```
71 \define@boolkey{glossaries.sty}[gls]{translate}[true]{}
```

Set the default value:

```
72 \glstranslatefalse
73 \@ifpackageloaded{translator}{\glstranslatetrue}{%
74 \@ifpackageloaded{babel}{\glstranslatetrue}{}}
```

Set the long form of the acronym in footnote on first use.

```
75 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
76 \ifthenelse{\boolean{glsacrdescription}}{}{}}
```

```

77 {\renewcommand*{\@gls@sanitizedesc}{}}%
78 }

Allow acronyms to have a description (needs to be set using the description key in
the optional argument of \newacronym).
79 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
80   \renewcommand*{\@gls@sanitizesymbol}{}}%
81 }

Define \newacronym to set the short form in small capitals.
82 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
83   \renewcommand*{\@gls@sanitizesymbol}{}}%
84 }

Define \newacronym to always use the long forms (i.e. don't use acronyms)
85 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
86   \renewcommand*{\@gls@sanitizesymbol}{}}%
87 }

Process package options:
88 \ProcessOptionsX

If chapters are defined and the user has requested the section counter as a
package option, \@chapter will be modified so that it adds a section.<n>.0
target, otherwise entries placed before the first section of a chapter will have
undefined links.

The same problem will also occur if a lower sectional unit is used, but
this is less likely to happen. If it does, or if you change \glscounter
to section later, you will have to specify a different counter for the en-
tries that give rise to a name{<section-level>.<n>.0} non-existent warning (e.g.
\gls[counter=chapter]{label}).
89 \ifthenelse{\equal{\glscounter}{section}}{%
90   \@ifundefined{chapter}{}%
91   \let\@gls@old@chapter\@chapter
92   \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
93   \@ifundefined{hyperdef}{\hyperdef{section}{\thesection}}{}}}%

```

5.3 Default values

This section sets up default values that are used by this package. Some of the names may already be defined (e.g. by `babel`) so `\providecommand` is used.

Main glossary title:

`\glossaryname`

```
94 \providecommand*\glossaryname{Glossary}
```

The title for the acronym glossary type (which is defined if `acronym` package option is used) is given by `\acronymname`. If the `acronym` package option is not used, `\acronymname` won't be used.

`\acronymname`

```
95 \providecommand*\acronymname{Acronyms}
```

The following commands provide text for the headers used by some of the tabular-like glossary styles. Whether or not they get used in the glossary depends on the glossary style.

`\entryname`

96 `\providecommand*{\entryname}{Notation}`

`\descriptionname`

97 `\providecommand*{\descriptionname}{Description}`

`\symbolname`

98 `\providecommand*{\symbolname}{Symbol}`

`\pagelistname`

99 `\providecommand*{\pagelistname}{Page List}`

Labels for `makeindex`'s symbol and number groups:

`\glssymbolsgroupname`

100 `\providecommand*{\glssymbolsgroupname}{Symbols}`

`\glsnumbersgroupname`

101 `\providecommand*{\glsnumbersgroupname}{Numbers}`

Add multi-lingual support. Thanks to everyone who contributed to the translations from both `comp.text.tex` and via email.

102 `\ifglstranslate`

If translator is not install, used standard `babel` captions, otherwise load translator dictionary.

103 `\@ifpackageloaded{translator}{\usedictionary{glossaries-dictionary}}%`
104 `\renewcommand*{\glossaryname}{\translate{Glossary}}%`
105 `\renewcommand*{\acronymname}{\translate{Acronyms}}%`
106 `\renewcommand*{\entryname}{\translate{Notation (glossaries)}}%`
107 `\renewcommand*{\descriptionname}{\translate{Description (glossaries)}}%`
108 `\renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%`
109 `\renewcommand*{\pagelistname}{\translate{Page List (glossaries)}}%`
110 `\renewcommand*{\glssymbolsgroupname}{\translate{Symbols (glossaries)}}%`
111 `\renewcommand*{\glsnumbersgroupname}{\translate{Numbers (glossaries)}}%`
112 `\fi`
113 `\@ifpackageloaded{babel}{\RequirePackage{glossaries-babel}}{\fi}`
114 `\fi`

The description terminator is given by `\glspostdescription` (except for the 3 and 4 column styles). This is a full stop by default:

`\glspostdescription`

115 `\newcommand*{\glspostdescription}{.}`

The name of the `makeindex` style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so redefining this command will only have an effect if it is done *before* the `.ist` file is created.

`\istfilename`

116 `\providecommand*{\istfilename}{\jobname.ist}`

The `makeglossaries` Perl script picks up this name from the auxiliary file and passes it to `makeindex` using the `-s` option. Since its not required by L^AT_EX, `\@istfilename` ignores its argument.

`\@istfilename`

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

This command is the value of the `page_compositor` `makeindex` key. Again, any redefinition of this command must take place *before* `\writeist` otherwise it will have no effect.

`\glscompositor`

```
118 \newcommand{\glscompositor}{.}
```

(The page compositor is usually defined as a dash when using `makeindex`, but most of the standard counters used by L^AT_EX use a full stop as the compositor, which is why I have used it as the default.)

The command `\glsnumberformat` indicates the default format for the page numbers in the glossary. (Note that this is not the same as `\glossaryentrynumbers`, but applies to individual numbers or groups of numbers within an entry’s associated number list.) If hyperlinks are defined, it will use `\glshypernumber`, otherwise it will simply display its argument “as is”.

`\glsnumberformat`

```
119 \@ifundefined{hyperlink}{%
```

```
120 \newcommand*\glsnumberformat}[1]{#1}}{%
```

```
121 \newcommand*\glsnumberformat}[1]{\glshypernumber{#1}}}
```

Individual numbers in an entry’s associated number list are delimited using `\delimN` (which corresponds to the `delim_n` `makeindex` keyword.) The default value is a comma followed by a space.

`\delimN`

```
122 \newcommand{\delimN}{, }
```

A range of numbers within an entry’s associated number list is delimited using `\delimR` (which corresponds to the `delim_r` `makeindex` keyword.) The default is an en-dash.

`\delimR`

```
123 \newcommand{\delimR}{--}
```

The glossary preamble is given by `\glossarypreamble`. This will appear after the glossary sectioning command, and before the `theglossary` environment. It is designed to allow the user to add information pertaining to the glossary (e.g. “page numbers in italic indicate the primary definition”) therefore `\glossarypreamble` shouldn’t be affected by the glossary style. (So if you define your own glossary style, don’t have it change `\glossarypreamble`.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use `\printglossary` for each glossary type, instead of `\printglossaries`, and redefine `\glossarypreamble` before each `\printglossary`.

`\glossarypreamble`

```
124 \newcommand*\glossarypreamble{}
```

The glossary postamble is given by `\glossarypostamble`. This is provided to allow the user to add something after the end of the `theglossary` environment (again, this shouldn't be affected by the glossary style.) It is, of course, possible to simply add the text after `\printglossary`, but if you only want the postamble to appear after the first glossary, but not after subsequent glossaries, you can do something like:

```
\renewcommand{\glossarypostamble}{For a complete list of terms
see \cite{blah}\gdef\glossarypreamble{}}
```

`\glossarypostamble`

```
125 \newcommand*{\glossarypostamble}{}
```

The sectioning command that starts a glossary is given by `\glossarysection`. (This does not form part of the glossary style, and so should not be changed by a glossary style.) If `\phantomsection` is defined, it uses `\p@glossarysection`, otherwise it uses `\@glossarysection`.

`\glossarysection`

```
126 \newcommand*{\glossarysection}[2][\@gls@title]{%
127 \def\@gls@title{#2}%
128 \@ifundefined{phantomsection}{%
129 \@glossarysection{#1}{#2}}{\p@glossarysection{#1}{#2}}%
130 \mkboth{\glossarytoctitle}{\glossarytoctitle}%
131 }
```

The required sectional unit is given by `\@glossarysec` which was defined by the `section` package option. The starred form of the command is chosen. If you don't want any sectional command, you will need to redefine `\glossarysection`. The sectional unit can be changed, if different sectional units are required.

`\setglossarysection`

```
132 \newcommand*{\setglossarysection}[1]{%
133 \setkeys{glossaries.sty}{section=#1}}
```

The command `\@glossarysection` indicates how to start the glossary section if `\phantomsection` is not defined.

`\@glossarysection`

```
134 \newcommand*{\@glossarysection}[2]{%
135 \ifx\@glossarysecstar\@empty
136 \csname\@glossarysec\endcsname{#2}%
137 \else
138 \csname\@glossarysec\endcsname*{#2}%
139 \@gls@toc{#1}{\@glossarysec}%
140 \fi
141 \@glossaryseclabel}
```

As `\@glossarysection`, but put in `\phantomsection`, and swap where `\@gls@toc` goes. If using chapters do a `\clearpage`. This ensures that the hyper link from the table of contents leads to the line above the heading, rather than the line below it.

`\@pglossarysection`

```

142 \newcommand*{\@pglossarysection}[2]{%
143 \gls@doclearpage
144 \phantomsection
145 \ifx\@glossarysecstar\@empty
146 \csname\@glossarysec\endcsname{#2}%
147 \else
148 \@gls@toc{#1}{\@glossarysec}%
149 \csname\@glossarysec\endcsname*{#2}%
150 \fi
151 \@glossaryseclabel}

```

The `\gls@doclearpage` command is used to issue a `\clearpage` (or `\cleardoublepage`) depending on whether the glossary sectional unit is a chapter. If the sectional unit is something else, do nothing.

`\gls@doclearpage`

```

152 \newcommand{\gls@doclearpage}{%
153 \ifthenelse{\equal{\@glossarysec}{chapter}}{%
154 \@ifundefined{cleardoublepage}{\clearpage}{\cleardoublepage}}{}%
155 }

```

The glossary is added to the table of contents if `glstoc` flag set. If it is set, `\@gls@toc` will add a line to the `.toc` file, otherwise it will do nothing. (The first argument to `\@gls@toc` is the title for the table of contents, the second argument is the sectioning type.)

`\@gls@toc`

```

156 \newcommand*{\@gls@toc}[2]{%
157 \ifglstoc
158 \ifglsnumberline
159 \addcontentsline{toc}{#2}{\numberline{#1}%
160 \else
161 \addcontentsline{toc}{#2}{#1}%
162 \fi
163 \fi}

```

5.4 Loops and conditionals

To iterate through all glossaries (or comma-separated list of glossary names given in optional argument) use:

`\forallglossaries[<glossary list>]{<cmd>}{<code>}`

where *<cmd>* is a control sequence which will be set to the name of the glossary in the current iteration.

`\forallglossaries`

```

164 \newcommand*{\forallglossaries}[3][\@glo@types]{%
165 \@for#2:=#1\do{\ifthenelse{\equal{#2}{} }{\@#3}}

```

To iterate through all entries in a given glossary use:

`\forglentries[<type>]{<cmd>}{<code>}`

where $\langle type \rangle$ is the glossary label and $\langle cmd \rangle$ is a control sequence which will be set to the entry label in the current iteration.

`\forglsentries`

```
166 \newcommand*\forglsentries}[3][\glsdefaulttype]{%
167 \edef\@glo@list{\csname glolist@#1\endcsname}%
168 \@for#2:=\@glo@list\do{%
169 \ifthenelse{\equal{#2}{}}{\#{3}}}
```

To iterate through all glossary entries over all glossaries listed in the optional argument (the default is all glossaries) use:

`\forallglsentries[$\langle glossary list \rangle$]{ $\langle cmd \rangle$ }{ $\langle code \rangle$ }`

Within `\forallglsentries`, the current glossary type is given by `\@this@glo@`.

`\forallglsentries`

```
170 \newcommand*\forallglsentries}[3][\@glo@types]{%
171 \expandafter\forallglossaries\expandafter[#1]{\@this@glo@}{%
172 \forglsentries[\@this@glo@]{#2}{#3}}
```

To check to see if a glossary exists use:

`\ifglossaryexists{ $\langle type \rangle$ }{ $\langle true-text \rangle$ }{ $\langle false-text \rangle$ }`

where $\langle type \rangle$ is the glossary's label.

`\ifglossaryexists`

```
173 \newcommand{\ifglossaryexists}[3]{%
174 \@ifundefined{glo@#1@out}{#3}{#2}}
```

To check to see if a glossary entry has been defined use:

`\ifglsentryexists{ $\langle label \rangle$ }{ $\langle true text \rangle$ }{ $\langle false text \rangle$ }`

where $\langle label \rangle$ is the entry's label.

`\ifglsentryexists`

```
175 \newcommand{\ifglsentryexists}[3]{%
176 \@ifundefined{glo@#1@name}{#3}{#2}}
```

To determine if given glossary entry has been used in the document text yet use:

`\ifglsused{ $\langle label \rangle$ }{ $\langle true text \rangle$ }{ $\langle false text \rangle$ }`

where $\langle label \rangle$ is the entry's label. If true it will do $\langle true text \rangle$ otherwise it will do $\langle false text \rangle$.

`\ifglsused`

```
177 \newcommand*\ifglsused}[3]{\ifthenelse{\boolean{glo@#1@flag}}{#2}{#3}}
```

The following two commands will cause an error if the given condition fails:

`\glsdoifexists{ $\langle label \rangle$ }{ $\langle code \rangle$ }`

Generate an error if entry specified by $\langle label \rangle$ doesn't exist, otherwise do $\langle code \rangle$.

`\glsdoifexists`

```
178 \newcommand{\glsdoifexists}[2]{\ifglentryexists{#1}{#2}{%
179 \PackageError{glossaries}{Glossary entry ‘#1’ has not been
180 defined.}{You need to define a glossary entry before you
181 can use it.}}
```

`\glsdoifnoexists{<label>}{<code>}`

The opposite: only do second argument if the entry doesn't exist. Generate an error message if it exists.

`\glsdoifnoexists`

```
182 \newcommand{\glsdoifnoexists}[2]{\ifglentryexists{#1}{%
183 \PackageError{glossaries}{Glossary entry ‘#1’ has already
184 been defined.}}{#2}}
```

5.5 Defining new glossaries

A comma-separated list of glossary names is stored in `\@glo@types`. When a new glossary type is created, its identifying name is added to this list. This is used by commands that iterate through all glossaries (such as `\makeglossaries` and `\printglossaries`.)

`\@glo@types`

```
185 \newcommand*{\@glo@types}{,}
```

A new glossary type is defined using `\newglossary`. Syntax:

`\newglossary[<log-ext>]{<name>}{<in-ext>}{<out-ext>}{<title>}{<counter>}`

where `<log-ext>` is the extension of the `makeindex` transcript file, `<in-ext>` is the extension of the glossary input file (read in by `\printglossary` and created by `makeindex`), `<out-ext>` is the extension of the glossary output file which is read in by `makeindex` (lines are written to this file by the `\glossary` command), `<title>` is the title of the glossary that is used in `\glossarysection` and `<counter>` is the default counter to be used by entries belonging to this glossary. The `makeglossaries` Perl script reads in the relevant extensions from the auxiliary file, and passes the appropriate file names and switches to `makeindex`.)

`\newglossary`

```
186 \newcommand*{\newglossary}[5][glg]{%
187 \ifglossaryexists{#2}{%
188 \PackageError{glossaries}{Glossary type ‘#2’ already exists}{%
189 You can't define a new glossary called ‘#2’ because it already
190 exists}%
191 }{%
```

Add this to the list of glossary types:

```
192 \toks@{#2}\edef\@glo@types{\@glo@types\the\toks@,}%
```

Define a comma-separated list of labels for this glossary type, so that all the entries for this glossary can be reset with a single command. When a new entry is created, its label is added to this list.

```
193 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store details of this new glossary type:

```
194 \expandafter\def\csname @glotype@#2@in\endcsname{#3}%
195 \expandafter\def\csname @glotype@#2@out\endcsname{#4}%
196 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
197 \protected@write\@auxout{}\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsdisplay` and `\glsdisplayfirst` by default). These can be redefined by the user later if required (see `\defglsdisplay` and `\defglsdisplayfirst`)

```
198 \expandafter\gdef\csname gls@#2@display\endcsname{%
199 \glsdisplay}%
200 \expandafter\gdef\csname gls@#2@displayfirst\endcsname{%
201 \glsdisplayfirst}%
```

Find out if the final optional argument has been specified, and use it to set the counter associated with this glossary. (Uses `\glscounter` if no optional argument is present.)

```
202 \@ifnextchar[{\@gls@setcounter{#2}}{\@gls@setcounter{#2}[\glscounter]}]}
```

Only defined new glossaries in the preamble:

```
203 \@onlypreamble{\newglossary}
```

`\newglossary` is used to specify the file extensions for the `makeindex` input, output and transcript files. It is written to the auxiliary file by `\newglossary`. Since it is not used by L^AT_EX, `\newglossary` simply ignores its arguments.

`\@newglossary`

```
204 \newcommand*{\@newglossary}[4]{}
```

Store counter to be used for given glossary type (the first argument is the glossary label, the second argument is the name of the counter):

`\@gls@setcounter`

```
205 \def\@gls@setcounter#1[#2]{%
206 \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%
207 }
```

Get counter associated with given glossary (the argument is the glossary label):

`\@gls@getcounter`

```
208 \newcommand*{\@gls@getcounter}[1]{%
209 \csname @glotype@#1@counter\endcsname}
```

Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`.

```
210 \newglossary{main}{gls}{glo}{\glossaryname}
```

5.6 Defining new entries

New glossary entries are defined using `\newglossaryentry`. This command requires a label and a key-value list that defines the relevant information for that entry. The definition for these keys follows. Note that the `name`, `description` and `symbol` keys will be sanitized later, depending on the value of the package option `sanitize` (this means that if some of the keys haven't been defined, they can be constructed from the `name` and `description` key before they are sanitized).

The `name` key indicates the name of the term being defined. This is how the term will appear in the glossary. The `name` key is required when defining a new glossary entry.

```
211 \define@key{glossentry}{name}{%
212 \def\@glo@name{#1}%
213 }
```

The `description` key is usually only used in the glossary, but can be made to appear in the text by redefining `\glsdisplay` and `\glsdisplayfirst` (or using `\defglsdisplay` and `\defglsdisplayfirst`), however, you will have to disable the `sanitize` option (using the `sanitize` package option, `sanitize={description=false}`, and protect fragile commands.) The `description` key is required when defining a new glossary entry. (Be careful not to make the description too long, because `makeindex` has a limited buffer. `\@glo@desc` is defined to be a short command to discourage lengthy descriptions for this reason. If you do have a very long description, or if you require paragraph breaks, define a separate command that contains the description, and use it as the value to the `description` key.)

```
214 \define@key{glossentry}{description}{%
215 \def\@glo@desc{#1}%
216 }
```

```
217 \define@key{glossentry}{descriptionplural}{%
218 \def\@glo@descplural{#1}%
219 }
```

The `sort` key needs to be sanitized here (the sort key is provided for `makeindex`'s benefit, not for use in the document.) The `sort` key is optional when defining a new glossary entry. If omitted, the value is given by *<name>* *<description>*.

```
220 \define@key{glossentry}{sort}{%
221 \def\@glo@sort{#1}%
222 \@onelevel@sanitize\@glo@sort}
```

The `text` key determines how the term should appear when used in the document (i.e. outside of the glossary). If omitted, the value of the `name` key is used instead.

```
223 \define@key{glossentry}{text}{%
224 \def\@glo@text{#1}%
225 }
```

The `plural` key determines how the plural form of the term should be displayed in the document. If omitted, the plural is constructed by appending an “s” to the value of the `text` key.

```
226 \define@key{glossentry}{plural}{%
227 \def\@glo@plural{#1}%
228 }
```

The `first` key determines how the entry should be displayed in the document when it is first used. If omitted, it is taken to be the same as the value of the `text` key.

```
229 \define@key{glossentry}{first}{%
230 \def\@glo@first{#1}%
231 }
```

The `firstplural` key is used to set the plural form for first use, in the event that the plural is required the first time the term is used. If omitted, it is constructed by appending an “s” to the value of the first key.

```
232 \define@key{glossentry}{firstplural}{%
233 \def\@glo@firstplural{#1}%
234 }
```

The `symbol` key is ignored by most of the predefined glossary styles, and defaults to `\relax` if omitted. It is provided for glossary styles that require an associated symbol, as well as a name and description. To make this value appear in the glossary, you need to redefine `\glossaryentryfield` so that it uses its fourth parameter. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsdisplay` and `\glsdisplayfirst` (either explicitly for all glossaries or via `\defglsdisplay` and `\defglsdisplayfirst` for individual glossaries.)

```
235 \define@key{glossentry}{symbol}{%
236 \def\@glo@symbol{#1}%
237 }

238 \define@key{glossentry}{symbolplural}{%
239 \def\@glo@symbolplural{#1}%
240 }
```

The `type` key specifies to which glossary this entry belongs. If omitted, the default glossary is used.

```
241 \define@key{glossentry}{type}{%
242 \def\@glo@type{#1}}
```

The `counter` key specifies the name of the counter associated with this glossary entry:

```
243 \define@key{glossentry}{counter}{%
244 \@ifundefined{c@#1}{\PackageError{glossaries}{There is no counter
245 called ‘#1’}{The counter key should have the name of a valid
246 counter as its value}}{%
247 \def\@glo@counter{#1}}}
```

Define `\newglossaryentry` $\langle label \rangle$ $\langle key-val list \rangle$. There are two required fields in $\langle key-val list \rangle$: name and description. (See above.)

`\newglossaryentry`

```
248 \DeclareRobustCommand{\newglossaryentry}[2]{%
  Check to see if this glossary entry has already been defined:
249 \glsdoifnoexists{#1}{%
  Set up defaults. If the name or description keys are omitted, an error will be
  generated.
250 \def\@glo@name{\PackageError{glossaries}{name key required in
251 \string\newglossaryentry}{You haven’t specified the entry name}}%
252 \def\@glo@desc{\PackageError{glossaries}{description key required in
253 \string\newglossaryentry}{You haven’t specified the entry description}}%

254 \def\@glo@descplural{\@glo@desc}%
255 \def\@glo@type{\glsdefaulttype}%
256 \def\@glo@symbol{\relax}%
```

257 \def\@glo@symbolplural{\@glo@symbol}%

258 \def\@glo@text{\@glo@name}%

259 \def\@glo@plural{\@glo@text s}%

260 \def\@glo@first{\@glo@text}%

261 \def\@glo@firstplural{\@glo@first s}%

262 \def\@glo@sort{\@glo@name}%

263 \def\@glo@counter{\@gls@getcounter{\@glo@type}}%

Extract key-val information from third parameter:

264 \setkeys{glossentry}{#2}%

Check to see if this glossary type has been defined, if it has, add this label to the relevant list, otherwise generate an error.

265 \@ifundefined{glolist@\@glo@type}{\PackageError{glossaries}{%

266 Glossary type '\@glo@type' has not been defined}{%

267 You need to define a new glossary type, before making entries

268 in it}}{%

269 \protected@edef\@glolist@\csname glolist@\@glo@type\endcsname}%

270 \expandafter\xdef\csname glolist@\@glo@type\endcsname{\@glolist@{#1},}%

271 }%

Define commands associated with this entry:

272 \expandafter\protected@xdef\csname glo@#1@text\endcsname{\@glo@text}%

273 \expandafter\protected@xdef\csname glo@#1@plural\endcsname{\@glo@plural}%

274 \expandafter\protected@xdef\csname glo@#1@first\endcsname{\@glo@first}%

275 \expandafter\protected@xdef\csname glo@#1@firstpl\endcsname{\@glo@firstplural}%

276 \expandafter\protected@xdef\csname glo@#1@type\endcsname{\@glo@type}%

277 \expandafter\protected@xdef\csname glo@#1@counter\endcsname{\@glo@counter}%

278 \@gls@sanitizename

279 \expandafter\protected@xdef\csname glo@#1@name\endcsname{\@glo@name}%

280 \@gls@sanitizedesc

281 \expandafter\protected@xdef\csname glo@#1@desc\endcsname{\@glo@desc}%

282 \expandafter\protected@xdef\csname glo@#1@descplural\endcsname{\@glo@descplural}%

283 \expandafter\protected@xdef\csname glo@#1@sort\endcsname{\@glo@sort}%

284 \@gls@sanitizesymbol

285 \expandafter\protected@xdef\csname glo@#1@symbol\endcsname{\@glo@symbol}%

286 \expandafter\protected@xdef\csname glo@#1@symbolplural\endcsname{\@glo@symbolplural}%

Define an associated boolean variable to determine whether this entry has been used yet (needs to be defined globally):

287 \expandafter\gdef\csname glo@#1@flagfalse\endcsname{%

288 \expandafter\global\expandafter

289 \let\csname ifglo@#1@flag\endcsname\iffalse}%

290 \expandafter\gdef\csname glo@#1@flagtrue\endcsname{%

291 \expandafter\global\expandafter

292 \let\csname ifglo@#1@flag\endcsname\iftrue}%

293 \csname glo@#1@flagfalse\endcsname

294 }}

Only defined new glossary entries in the preamble:

295 \@onlypreamble{\newglossaryentry}

5.7 Resetting and unsetting entry flags

Each glossary entry is assigned a conditional of the form `\ifglo@<label>@flag` which determines whether or not the entry has been used (see also `\ifglsused` defined below.) These flags can be set and unset using the following macros:

The command `\glsreset{<label>}` can be used to set the entry flag to indicate that it hasn't been used yet. The required argument is the entry label.

`\glsreset`

```
296 \newcommand*{\glsreset}[1]{%
297 \glsdoifexists{#1}{%
298 \expandafter\global\csname glo@#1@flagfalse\endcsname}}
```

As above, but with only a local effect:

`\glslocalreset`

```
299 \newcommand*{\glslocalreset}[1]{%
300 \glsdoifexists{#1}{%
301 \expandafter\let\csname ifglo@#1@flag\endcsname\iffalse}}
```

The command `\glsunset{<label>}` can be used to set the entry flag to indicate that it has been used. The required argument is the entry label.

`\glsunset`

```
302 \newcommand*{\glsunset}[1]{%
303 \glsdoifexists{#1}{%
304 \expandafter\global\csname glo@#1@flagtrue\endcsname}}
```

As above, but with only a local effect:

`\glslocalunset`

```
305 \newcommand*{\glslocalunset}[1]{%
306 \glsdoifexists{#1}{%
307 \expandafter\let\csname ifglo@#1@flag\endcsname\iftrue}}
```

Reset all entries for the named glossaries (supplied in a comma-separated list).
Syntax: `\glsresetall[<glossary-list>]`

`\glsresetall`

```
308 \newcommand*{\glsresetall}[1][\@glo@types]{%
309 \forallglsentries[1]{\@glsentry}{%
310 \glsreset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalresetall`

```
311 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
312 \forallglsentries[1]{\@glsentry}{%
313 \glslocalreset{\@glsentry}}}
```

Unset all entries for the named glossaries (supplied in a comma-separated list).
Syntax: `\glsunsetall[<glossary-list>]`

`\glsunsetall`

```
314 \newcommand*{\glsunsetall}[1][\@glo@types]{%
315 \forallglsentries[1]{\@glsentry}{%
316 \glsunset{\@glsentry}}}
```

As above, but with only a local effect:

```
\glslocalunsetall
317 \newcommand*\glslocalunsetall}[1][\@gls@types]{%
318 \forallglsentries[#1]{\@glsentry}{%
319 \glslocalunset{\@glsentry}}}
```

5.8 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.⁷

```
\loadglsentries[<type>]{<filename>}
```

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the `type` key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary.) The mandatory argument is the filename (with or without `.tex` extension.)

```
\loadglsentries
320 \newcommand*\loadglsentries}[2][\@gls@default]{%
321 \let\@gls@default\glsdefaulttype
322 \def\glsdefaulttype{#1}\input{#2}%
323 \let\glsdefaulttype\@gls@default}

\loadglsentries can only be used in the preamble:
324 \@onlypreamble{\loadglsentries}
```

5.9 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use.) Any formatting commands (such as `\textbf`) is governed by `\glstextformat`. By default this just displays the link text “as is”.

```
\glstextformat
325 \newcommand*\glstextformat}[1]{#1}
```

The first time an entry is used, the way in which it is displayed is governed by `\glsdisplayfirst`. This takes four parameters: `#1` will be the value of the entry's `first` or `firstplural` key, `#2` will be the value of the entry's `description` key, `#3` will be the value of the entry's `symbol` key and `#4` is additional text supplied by the final optional argument to commands like `\gls` and `\glspl`. The default is to display the first parameter followed by the additional text.

⁷and any other valid L^AT_EX code that can be used in the preamble.

`\glsdisplayfirst`

```
326 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

After the first use, the entry is displayed according to the format of `\glsdisplay`. Again, it takes four parameters: `#1` will be the value of the entry's text or plural key, `#2` will be the value of the entry's description key, `#3` will be the value of the entry's symbol key and `#4` is additional text supplied by the final optional argument to commands like `\gls` and `\glspl`.

`\glsdisplay`

```
327 \newcommand*{\glsdisplay}[4]{#1#4}
```

When a new glossary is created it uses `\glsdisplayfirst` and `\glsdisplay` as the default way of displaying its entry in the text. This can be changed for the entries belonging to an individual glossary using `\defglsdisplay` and `\defglsdisplayfirst`.

```
\defglsdisplay[⟨type⟩]{⟨definition⟩}
```

The glossary type is given by `⟨type⟩` (the default glossary if omitted) and `⟨definition⟩` should have at most `#1`, `#2`, `#3` and `#4`. These represent the same arguments as those described for `\glsdisplay`.

`\defglsdisplay`

```
328 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
329 \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}}
```

```
\defglsdisplayfirst[⟨type⟩]{⟨definition⟩}
```

The glossary type is given by `⟨type⟩` (the default glossary if omitted) and `⟨definition⟩` should have at most `#1`, `#2`, `#3` and `#4`. These represent the same arguments as those described for `\glsdisplayfirst`.

`\defglsdisplayfirst`

```
330 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
331 \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}}
```

5.9.1 Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defglsdisplay` and `\defglsdisplayfirst`). It goes against the L^AT_EX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}[s]` rather than, say, `\gls[append=s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{⟨label⟩}` to ignore following spaces, so `\new@ifnextchar` from the `amsgen` package is required.

The following keys can be used in the first optional argument. The `counter` key checks that the value is the name of a valid counter.

```

332 \define@key{glslink}{counter}{%
333 \@ifundefined{c@#1}{\PackageError{glossaries}{There is no counter
334 called ‘#1’}{The counter key should have the name of a valid
335 counter as its value}}{%
336 \def\@gls@counter{#1}}}
```

The value of the format key should be the name of a command (without the initial backslash) that has a single mandatory argument which can be used to format the associated entry number.

```

337 \define@key{glslink}{format}{%
338 \def\@glsnumberformat{#1}}
```

The hyper key is a boolean key, it can either have the value true or false, and indicates whether or not to make a hyperlink to the relevant glossary entry. If hyper is false, an entry will still be made in the glossary, but the given text won't be a hyperlink.

```

339 \define@boolkey{glslink}{hyper}[true]{}
```

Syntax:

```
\glslink[<options>]{<label>}{<text>}
```

Display *<text>* in the document, and add the entry information for *<label>* into the relevant glossary. The optional argument should be a key value list using the `glslink` keys defined above.

There is also a starred version:

```
\glslink*[<options>]{<label>}{<text>}
```

which is equivalent to `\glslink[hyper=false,<options>]{<label>}{<text>}`

First determine whether or not we are using the starred version:

```
\glslink
```

```

340 \newcommand{\glslink}{%
341 \@ifstar{\sgls@link\@gls@link}
```

Define the starred version:

```
\sgls@link
```

```

342 \newcommand*{\sgls@link}[1][]{\@gls@link[hyper=false,#1]}
```

Define the un-starred version:

```
\@gls@link
```

```

343 \newcommand*{\@gls@link}[3][]{%
344 \glsdoifexists{#2}{%
345 \def\@glsnumberformat{glsnumberformat}%
346 \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
347 \KV@glslink@hypertrue
348 \setkeys{glslink}{#1}%
349 \edef\theglsentrycounter{\expandafter\noexpand\csname the\@gls@counter\endcsname}%
350 \ifKV@glslink@hyper
351 \@glslink{glo:#2}{\glstextformat{#3}}%
352 \else
353 \glstextformat{#3}\relax
```

```

354 \fi
355 \protected@edef\@glo@sort{\csname glo@#2@sort\endcsname}%
356 \@gls@checkmkidxchars\@glo@sort
357 \protected@edef\@glo@name{\csname glo@#2@name\endcsname}%
358 \@gls@checkmkidxchars\@glo@name
359 \protected@edef\@glo@name{\string\glsnamefont{\@glo@name}}%
360 \protected@edef\@glo@desc{\csname glo@#2@desc\endcsname}%
361 \@gls@checkmkidxchars\@glo@desc
362 \protected@edef\@glo@symbol{\csname glo@#2@symbol\endcsname}%
363 \@gls@checkmkidxchars\@glo@symbol
364 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
365 \glossary[\csname glo@#2@type\endcsname]{%
366 \@glo@sort\@gls@actualchar
367 \string\glossaryentryfield{#2}{\@glo@name}{\@glo@desc
368 }{\@glo@symbol}\@gls@encapchar\@glo@numfmt}%
369 }}

```

Set the formatting information in the format required by `makeindex`:

`\@set@glo@numformat`

```

370 \def\@set@glo@numformat#1#2#3{%
371 \expandafter\@glo@check@mkidxrangechar#3\@nil
372 \protected@edef#1{\@glo@prefix setentrycounter{#2}%
373 \expandafter\string\csname\@glo@sufffix\endcsname}%
374 \@gls@checkmkidxchars#1}

```

Check to see if the given string starts with a (or). If it does set `\@glo@prefix` to the starting character, and `\@glo@sufffix` to the rest (or `glsnumberformat` if there is nothing else), otherwise set `\@glo@prefix` to nothing and `\@glo@sufffix` to all of it.

```

375 \def\@glo@check@mkidxrangechar#1#2\@nil{%
376 \if#1(\relax
377   \def\@glo@prefix{(%
378   \if\relax#2\relax
379     \def\@glo@sufffix{glsnumberformat}%
380   \else
381     \def\@glo@sufffix{#2}%
382   \fi
383 \else
384   \if#1)\relax
385     \def\@glo@prefix{)}%
386   \if\relax#2\relax
387     \def\@glo@sufffix{glsnumberformat}%
388   \else
389     \def\@glo@sufffix{#2}%
390   \fi
391 \else
392   \def\@glo@prefix{}\def\@glo@sufffix{#1#2}%
393 \fi
394 \fi}

```

Catch `makeindex` special characters:

`\@gls@checkmkidxchars`

```

395 \newcommand{\@gls@checkmkidxchars}[1]{%

```

```

396 \def\@gls@checkedmkidx{%
397 \expandafter\@gls@checkquote#1\@nil""\null%
398 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
399 \def\@gls@checkedmkidx{%
400 \expandafter\@gls@checkescquote#1\@nil""\null%
401 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
402 \def\@gls@checkedmkidx{%
403 \expandafter\@gls@checkescactual#1\@nil"??\null%
404 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
405 \def\@gls@checkedmkidx{%
406 \expandafter\@gls@checkactual#1\@nil??\null%
407 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
408 \def\@gls@checkedmkidx{%
409 \expandafter\@gls@checkbar#1\@nil||\null%
410 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
411 \def\@gls@checkedmkidx{%
412 \expandafter\@gls@checkescbar#1\@nil|||\null%
413 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
414 \def\@gls@checkedmkidx{%
415 \expandafter\@gls@checklevel#1\@nil!!\null%
416 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
417 }

```

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

```

418 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

419 \newtoks\@gls@tmpb

```

\@gls@checkquote Replace " with "" since " is a makeindex special character.

```

420 \def\@gls@checkquote#1"#2"#3\null{%
421 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
422 \toks@={#1}%
423 \ifx\null#2\null%
424 \ifx\null#3\null
425 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
426 \def\@gls@checkquote{\relax}%
427 \else
428 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
429 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
430 \def\@gls@checkquote{\@gls@checkquote#3\null}%
431 \fi
432 \else
433 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
434 \@gls@quotechar\@gls@quotechar}%
435 \ifx\null#3\null
436 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
437 \else
438 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
439 \fi
440 \fi
441 \@gls@checkquote}

```

`\@gls@checkescquote` Do the same for `\`:

```
442 \def\@gls@checkescquote#1\"#2\"#3\null{%
443 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
444 \toks@={#1}%
445 \ifx\null#2\null%
446 \ifx\null#3\null
447 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
448 \def\@@gls@checkescquote{\relax}%
449 \else
450 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
451 \@gls@quotechar\string\" \@gls@quotechar
452 \@gls@quotechar\string\" \@gls@quotechar}%
453 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
454 \fi
455 \else
456 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
457 \@gls@quotechar\string\" \@gls@quotechar}%
458 \ifx\null#3\null
459 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
460 \else
461 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
462 \fi
463 \fi
464 \@@gls@checkescquote}
```

`\@gls@checkescactual` Similarly for `\?` (which is replaces `@` as `makeindex`'s special character):

```
465 \def\@gls@checkescactual#1\?#2\?#3\null{%
466 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
467 \toks@={#1}%
468 \ifx\null#2\null%
469 \ifx\null#3\null
470 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
471 \def\@@gls@checkescactual{\relax}%
472 \else
473 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
474 \@gls@quotechar\string\" \@gls@actualchar
475 \@gls@quotechar\string\" \@gls@actualchar}%
476 \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
477 \fi
478 \else
479 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
480 \@gls@quotechar\string\" \@gls@actualchar}%
481 \ifx\null#3\null
482 \def\@@gls@checkescactual{\@gls@checkescactual#2\?\? \null}%
483 \else
484 \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
485 \fi
486 \fi
487 \@@gls@checkescactual}
```

`\@gls@checkescbar` Similarly for `\|`:

```
488 \def\@gls@checkescbar#1\|#2\|#3\null{%
489 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
490 \toks@={#1}%
491 \ifx\null#2\null%
492 \ifx\null#3\null
493 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
494 \def\@@gls@checkescbar{\relax}%
495 \else
496 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
497 \@gls@quotechar\string\" \@gls@barchar
498 \@gls@quotechar\string\" \@gls@barchar}%
499 \def\@@gls@checkescbar{\@gls@checkescbar#3\null}%
500 \fi
501 \else
502 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
503 \@gls@quotechar\string\" \@gls@barchar}%
504 \ifx\null#3\null
505 \def\@@gls@checkescbar{\@gls@checkescbar#2\|\| \null}%
506 \else
507 \def\@@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
508 \fi
509 \fi
510 \@@gls@checkescbar}
```

```

491 \ifx\null#2\null%
492 \ifx\null#3\null
493   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
494   \def\@gls@checkescbar{\relax}%
495 \else
496   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
497     \@gls@quotechar\string"\@gls@encapchar
498     \@gls@quotechar\string"\@gls@encapchar}%
499   \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
500 \fi
501 \else
502   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
503     \@gls@quotechar\string"\@gls@encapchar}%
504   \ifx\null#3\null
505     \def\@gls@checkescbar{\@gls@checkescbar#2\|\| \null}%
506   \else
507     \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
508   \fi
509 \fi
510 \@gls@checkescbar}

```

\@gls@checkesclevel Similarly for \!:

```

511 \def\@gls@checkesclevel#1\!#2\!#3\null{%
512 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
513 \toks@={#1}%
514 \ifx\null#2\null%
515 \ifx\null#3\null
516   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
517   \def\@gls@checkesclevel{\relax}%
518 \else
519   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
520     \@gls@quotechar\string"\@gls@levelchar
521     \@gls@quotechar\string"\@gls@levelchar}%
522   \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
523 \fi
524 \else
525   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
526     \@gls@quotechar\string"\@gls@levelchar}%
527   \ifx\null#3\null
528     \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\| \null}%
529   \else
530     \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
531   \fi
532 \fi
533 \@gls@checkesclevel}

```

\@gls@checkbar and for |:

```

534 \def\@gls@checkbar#1|#2|#3\null{%
535 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
536 \toks@={#1}%
537 \ifx\null#2\null%
538 \ifx\null#3\null
539   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
540   \def\@gls@checkbar{\relax}%

```

```

541 \else
542 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
543 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
544 \def\@gls@checkbar{\@gls@checkbar#3\null}%
545 \fi
546 \else
547 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
548 \@gls@quotechar\@gls@encapchar}%
549 \ifx\null#3\null
550 \def\@gls@checkbar{\@gls@checkbar#2|\null}%
551 \else
552 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
553 \fi
554 \fi
555 \@gls@checkbar}

```

\@gls@checklevel and for !:

```

556 \def\@gls@checklevel#1!#2!#3\null{%
557 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
558 \toks@={#1}%
559 \ifx\null#2\null%
560 \ifx\null#3\null
561 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
562 \def\@gls@checklevel{\relax}%
563 \else
564 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
565 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
566 \def\@gls@checklevel{\@gls@checklevel#3\null}%
567 \fi
568 \else
569 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
570 \@gls@quotechar\@gls@levelchar}%
571 \ifx\null#3\null
572 \def\@gls@checklevel{\@gls@checklevel#2!\null}%
573 \else
574 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
575 \fi
576 \fi
577 \@gls@checklevel}

```

\@gls@checkactual and for ?:

```

578 \def\@gls@checkactual#1?#2?#3\null{%
579 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
580 \toks@={#1}%
581 \ifx\null#2\null%
582 \ifx\null#3\null
583 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
584 \def\@gls@checkactual{\relax}%
585 \else
586 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
587 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
588 \def\@gls@checkactual{\@gls@checkactual#3\null}%
589 \fi
590 \else

```

```

591 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
592 \@gls@quotechar\@gls@actualchar}%
593 \ifx\@null#3\@null
594 \def\@gls@checkactual{\@gls@checkactual#2??\@null}%
595 \else
596 \def\@gls@checkactual{\@gls@checkactual#2?#3\@null}%
597 \fi
598 \fi
599 \@gls@checkactual}

```

\@glslink If `\hyperlink` is not defined `\@glslink` ignores its first argument and just does the second argument, otherwise it is equivalent to `\hyperlink`.

```

600 \@ifundefined{hyperlink}{%
601 \gdef\@glslink#1#2{#2}%
602 }{%
603 \gdef\@glslink#1#2{\hyperlink{#1}{#2}}%
604 }

```

\@glstarget If `\hypertarget` is not defined, `\@glstarget` ignores its first argument and just does the second argument, otherwise it is equivalent to `\hypertarget`.

```

605 \@ifundefined{hypertarget}{%
606 \gdef\@glstarget#1#2{#2}%
607 }{%
608 \gdef\@glstarget#1#2{\hypertarget{#1}{#2}}

```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

\glsdisablehyper

```

609 \newcommand{\glsdisablehyper}{%
610 \renewcommand*\@glslink[2]{##2}%
611 \renewcommand*\@glstarget[2]{##2}}

```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

\glsenablehyper

```

612 \newcommand{\glsenablehyper}{%
613 \renewcommand*\@glslink[2]{\hyperlink{##1}{##2}}%
614 \renewcommand*\@glstarget[2]{\hypertarget{##1}{##2}}}

```

Syntax:

`\gls[<options>]{<label>}[<insert text>]`

Link to glossary entry using singular form. The link text is taken from the value of the text or first keys used when the entry was defined.

The first optional argument is a key-value list, the same as `\glslink`, the mandatory argument is the entry label. After the mandatory argument, there is another optional argument to insert extra text in the link text (the location of the inserted text is governed by `\glsdisplay` and `\glsdisplayfirst`). As with `\glslink` there is a starred version which is the same as the unstarred version but

with the `hyper` key set to `false`. (Additional options can also be specified in the first optional argument.)

First determine if we are using the starred form:

`\gls`

```
615 \newcommand*{\gls}{\@ifstar\@sgls\@gls}
```

Define the starred form:

`\@sgls`

```
616 \newcommand*{\@sgls}[1] [] {\@gls[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

`\@gls`

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

```
618 \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}]
```

Read in the final optional argument:

```
619 \def\@gls@{#1#2}[#3] {%
```

```
620 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```
621 \def\@gls@link@opts{#1}%
```

```
622 \def\@gls@link@label{#2}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
623 \ifglsused{#2}{\protected@edef\@glo@text{%
```

```
624 \csname gls@\@glo@type @display\endcsname
```

```
625 {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}{%
```

```
626 \protected@edef\@glo@text{%
```

```
627 \csname gls@\@glo@type @displayfirst\endcsname
```

```
628 {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
```

Call `\@gls@link`. If footnote package option has been used, suppress hyperlink for first use.

```
629 \ifglsused{#2}{%
```

```
630 \@gls@link[#1]{#2}{\@glo@text}%
```

```
631 }{%
```

```
632 \ifglsacrfootnote
```

```
633 \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
```

```
634 \else
```

```
635 \@gls@link[#1]{#2}{\@glo@text}%
```

```
636 \fi
```

```
637 }%
```

Indicate that this entry has now been used

```
638 \glsunset{#2}}%
```

```
639 }
```

`\Gls` behaves like `\gls`, but the first letter of the link text is converted to uppercase (note that if the first letter has an accent, the accented letter will need to be grouped when you define the entry.) It is mainly intended for terms that start a sentence:

`\Gls`

```
640 \newcommand*{\Gls}{\@ifstar\@sGls\@Gls}

Define the starred form:
641 \newcommand*{\@sGls}[1] [] {\@Gls[hyper=false,#1]}

Defined the un-starred form. Need to determine if there is a final optional argument
642 \newcommand*{\@Gls}[2] [] {%
643 \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}]

Read in the final optional argument:
644 \def\@Gls@#1#2[#3] {%
645 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%

Save options in \@gls@link@opts and label in \@gls@link@label
646 \def\@gls@link@opts{#1}%
647 \def\@gls@link@label{#2}%

Determine what the link text should be (this is stored in \@glo@text)
648 \ifglsused{#2}{\protected@edef\@glo@text{%
649 \csname gls@\@glo@type @display\endcsname
650 {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
651 \protected@edef\@glo@text{%
652 \csname gls@\@glo@type @displayfirst\endcsname
653 {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.
654 \ifglsused{#2}{%
655   \@gls@link[#1]{#2}{%
656     \expandafter\makefirstuc\expandafter{\@glo@text}}%
657 }{%
658   \ifglsacrfootnote
659     \@gls@link[#1,hyper=false]{#2}{%
660       \expandafter\makefirstuc\expandafter{\@glo@text}}%
661   \else
662     \@gls@link[#1]{#2}{%
663       \expandafter\makefirstuc\expandafter{\@glo@text}}%
664   \fi
665 }%

Indicate that this entry has now been used
666 \glsunset{#2}%
667 }
```

`\Gls` behaves like `\gls`, but the link text is converted to uppercase:

`\GLS`

```
668 \newcommand*{\GLS}{\@ifstar\@sGLS\@GLS}

Define the starred form:
669 \newcommand*{\@sGLS}[1] [] {\@GLS[hyper=false,#1]}

Defined the un-starred form. Need to determine if there is a final optional argument
670 \newcommand*{\@GLS}[2] [] {%
671 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}]
```

Read in the final optional argument:

```

672 \def\@GLS@#1#2[#3]{%
673 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Save options in \@gls@link@opts and label in \@gls@link@label
674 \def\@gls@link@opts{#1}%
675 \def\@gls@link@label{#2}%
    Determine what the link text should be (this is stored in \@glo@text)
676 \ifglsused{#2}{\protected@edef\@glo@text{%
677 \csname gls@\@glo@type @display\endcsname
678 {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
679 \protected@edef\@glo@text{%
680 \csname gls@\@glo@type @displayfirst\endcsname
681 {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%

```

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.

```

682 \ifglsused{#2}{%
683   \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
684 }{%
685   \ifglsacrfootnote
686     \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
687   \else
688     \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
689   \fi
690 }%

```

Indicate that this entry has now been used

```

691 \glsunset{#2}%
692 }

```

\glspl behaves in the same way as \gls except it uses the plural form.

\glspl

```

693 \newcommand*{\glspl}{\@ifstar\@sglspl\@glspl}

```

Define the starred form:

```

694 \newcommand*{\@sglspl}[1][]{\@glspl[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

695 \newcommand*{\@glspl}[2][]{%
696 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}]

```

Read in the final optional argument:

```

697 \def\@glspl@#1#2[#3]{%
698 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Save options in \@gls@link@opts and label in \@gls@link@label
699 \def\@gls@link@opts{#1}%
700 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

701 \ifglsused{#2}{\protected@edef\@glo@text{%
702 \csname gls@\@glo@type @display\endcsname
703 {\glsentryplural{#2}}{\glsentrydescplural{#2}}}%

```

```

704 \glsentrysymbolplural{#2}{#3}}{%
705 \protected@edef\@glo@text{%
706 \csname gls@\@glo@type @displayfirst\endcsname
707 {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}{%
708 \glsentrysymbolplural{#2}{#3}}}%

```

Call `\@gls@link` If footnote package option has been used, suppress hyperlink for first use.

```

709 \ifglsused{#2}{%
710   \@gls@link[#1]{#2}{\@glo@text}%
711 }{%
712   \ifglsacrfootnote
713     \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
714   \else
715     \@gls@link[#1]{#2}{\@glo@text}%
716   \fi
717 }%

```

Indicate that this entry has now been used

```

718 \glsunset{#2}}%
719 }

```

`\Glspl` behaves in the same way as `\glspl`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped.)

`\Glspl`

```

720 \newcommand*{\Glspl}{\@ifstar\@sGlspl\@Glspl}

```

Define the starred form:

```

721 \newcommand*{\@sGlspl}[1][]{\@Glspl[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

722 \newcommand*{\@Glspl}[2][]{%
723 \new@ifnextchar[\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[]}%

```

Read in the final optional argument:

```

724 \def\@Glspl@#1#2[#3]{%
725 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```

726 \def\@gls@link@opts{#1}%
727 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

728 \ifglsused{#2}{\protected@edef\@glo@text{%
729 \csname gls@\@glo@type @display\endcsname
730 {\glsentryplural{#2}}{\glsentrydescplural{#2}}{%
731 \glsentrysymbolplural{#2}{#3}}}%
732 \protected@edef\@glo@text{%
733 \csname gls@\@glo@type @displayfirst\endcsname
734 {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}{%
735 \glsentrysymbolplural{#2}{#3}}}%

```

Call `\@gls@link` If footnote package option has been used, suppress hyperlink for first use.

```

736 \ifglused{#2}{%
737   \@gls@link[#1]{#2}{%
738     \expandafter\makefirstuc\expandafter{\@glo@text}}%
739 }{%
740   \ifglacrfootnote
741     \@gls@link[#1,hyper=false]{#2}{%
742       \expandafter\makefirstuc\expandafter{\@glo@text}}%
743   \else
744     \@gls@link[#1]{#2}{%
745       \expandafter\makefirstuc\expandafter{\@glo@text}}%
746   \fi
747 }%

```

Indicate that this entry has now been used

```

748 \glsunset{#2}}%
749 }

```

`\GLSp1` behaves like `\glspl` except that all the link text is converted to uppercase.

`\GLSp1`

```

750 \newcommand*{\GLSp1}{\@ifstar\@sGLSp1\@GLSp1}

```

Define the starred form:

```

751 \newcommand*{\@sGLSp1}[1] [] {\@GLSp1[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

752 \newcommand*{\@GLSp1}[2] [] {%
753 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}]

```

Read in the final optional argument:

```

754 \def\@GLSp1@#1#2[#3]{%
755 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```

756 \def\@gls@link@opts{#1}%
757 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

758 \ifglused{#2}{\protected@edef\@glo@text{%
759 \csname gls@\@glo@type @display\endcsname
760 {\glsentryplural{#2}}{\glsentrydescplural{#2}}}%
761 \glsentrysymbolplural{#2}}{#3}}{%
762 \protected@edef\@glo@text{%
763 \csname gls@\@glo@type @displayfirst\endcsname
764 {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}}%
765 \glsentrysymbolplural{#2}}{#3}}}%

```

Call `\@gls@link` If footnote package option has been used, suppress hyperlink for first use.

```

766 \ifglused{#2}{%
767   \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
768 }{%
769   \ifglacrfootnote
770     \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
771   \else

```

```

772 \gls@link[#1]{#2}{\MakeUppercase{\glo@text}}%
773 \fi
774 }%

```

Indicate that this entry has now been used

```

775 \glsunset{#2}}%
776 }

```

`\glstext` behaves like `\gls` except it always uses the value given by the `text` key and it doesn't mark the entry as used.

`\glstext`

```

777 \newcommand*{\glstext}{\@ifstar\@sglstext\glstext}

```

Define the starred form:

```

778 \newcommand*{\@sglstext}[1][]{\@glstext[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

779 \newcommand*{\@glstext}[2][]{%
780 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[]}]

```

Read in the final optional argument:

```

781 \def\@glstext@#1#2[#3]{%
782 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

783 \protected@edef\@glo@text{\glsentrytext{#2}}%

```

Call `\@gls@link`

```

784 \@gls@link[#1]{#2}{\@glo@text}%
785 }%
786 }

```

`\GLStext` behaves like `\glstext` except the text is converted to uppercase.

`\GLStext`

```

787 \newcommand*{\GLStext}{\@ifstar\@sGLStext\@GLStext}

```

Define the starred form:

```

788 \newcommand*{\@sGLStext}[1][]{\@GLStext[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

789 \newcommand*{\@GLStext}[2][]{%
790 \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[]}]

```

Read in the final optional argument:

```

791 \def\@GLStext@#1#2[#3]{%
792 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

793 \protected@edef\@glo@text{\glsentrytext{#2}}%

```

Call `\@gls@link`

```

794 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
795 }%
796 }

```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
797 \newcommand*{\Glstext}{\@ifstar\@sGlstext\@Glstext}
```

Define the starred form:

```
798 \newcommand*{\@sGlstext}[1][\@Glstext[hyper=false,#1]]
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
799 \newcommand*{\@Glstext}[2][\%
```

```
800 \new@ifnextchar[\@Glstext@{#1}{#2}]{\@Glstext@{#1}{#2}[]}
```

Read in the final optional argument:

```
801 \def\@Glstext@#1#2[#3]{\%
```

```
802 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
803 \protected@edef\@glo@text{\glsentrytext{#2}}%
```

Call `\@gls@link`

```
804 \@gls@link[#1]{#2}{\%
```

```
805 \expandafter\makefirstuc\expandafter{\@glo@text}}%
```

```
806 }%
```

```
807 }
```

`\glsfirst` behaves like `\gls` except it always uses the value given by the first key and it doesn't mark the entry as used.

`\glsfirst`

```
808 \newcommand*{\glsfirst}{\@ifstar\@sglsfirst\@glsfirst}
```

Define the starred form:

```
809 \newcommand*{\@sglsfirst}[1][\@glsfirst[hyper=false,#1]]
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
810 \newcommand*{\@glsfirst}[2][\%
```

```
811 \new@ifnextchar[\@glsfirst@{#1}{#2}]{\@glsfirst@{#1}{#2}[]}
```

Read in the final optional argument:

```
812 \def\@glsfirst@#1#2[#3]{\%
```

```
813 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
814 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call `\@gls@link`

```
815 \@gls@link[#1]{#2}{\@glo@text}}%
```

```
816 }%
```

```
817 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
818 \newcommand*{\Glsfirst}{\@ifstar\@sGlsfirst\@Glsfirst}
```

Define the starred form:

```

819 \newcommand*{\@sGlsfirst}[1] [] {\@Glsfirst[hyper=false,#1]}
Defined the un-starred form. Need to determine if there is a final optional argu-
ment
820 \newcommand*{\@Glsfirst}[2] [] {%
821 \new@ifnextchar[\@Glsfirst@{#1}{#2}]{\@Glsfirst@{#1}{#2} [] }}
Read in the final optional argument:
822 \def\@Glsfirst@#1#2[#3] {%
823 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
Determine what the link text should be (this is stored in \@glo@text)
824 \protected@edef\@glo@text{\glstryfirst{#2}}%
Call \@gls@link
825 \@gls@link[#1]{#2}{%
826 \expandafter\makefirstuc\expandafter{\@glo@text}}%
827 }%
828 }

```

`\GLSfirst` behaves like `\Glsfirst` except it displays the text in uppercase.

`\GLSfirst`

```

829 \newcommand*{\GLSfirst}{\@ifstar\@sGLSfirst\@GLSfirst}
Define the starred form:
830 \newcommand*{\@sGLSfirst}[1] [] {\@GLSfirst[hyper=false,#1]}
Defined the un-starred form. Need to determine if there is a final optional argu-
ment
831 \newcommand*{\@GLSfirst}[2] [] {%
832 \new@ifnextchar[\@GLSfirst@{#1}{#2}]{\@GLSfirst@{#1}{#2} [] }}
Read in the final optional argument:
833 \def\@GLSfirst@#1#2[#3] {%
834 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
Determine what the link text should be (this is stored in \@glo@text)
835 \protected@edef\@glo@text{\glstryfirst{#2}}%
Call \@gls@link
836 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
837 }%
838 }

```

`\glsplural` behaves like `\gls` except it always uses the value given by the plural key and it doesn't mark the entry as used.

`\glsplural`

```

839 \newcommand*{\glsplural}{\@ifstar\@sglsplural\@glsplural}
Define the starred form:
840 \newcommand*{\@sglsplural}[1] [] {\@glsplural[hyper=false,#1]}
Defined the un-starred form. Need to determine if there is a final optional argu-
ment
841 \newcommand*{\@glsplural}[2] [] {%
842 \new@ifnextchar[\@glsplural@{#1}{#2}]{\@glsplural@{#1}{#2} [] }}

```


Read in the final optional argument:

```

843 \def\@glsplural@#1#2[#3]{%
844 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
845 \protected@edef\@glo@text{\glsentryplural{#2}}%
    Call \@gls@link
846 \@gls@link[#1]{#2}{\@glo@text}%
847 }%
848 }

```

`\Glsplural` behaves like `\glsplural` except that the first letter is converted to uppercase.

`\Glsplural`

```

849 \newcommand*\@Glsplural{\@ifstar\@sGlsplural\@Glsplural}

    Define the starred form:
850 \newcommand*\@sGlsplural[1][\@Glsplural[hyper=false,#1]]{
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
851 \newcommand*\@Glsplural[2][\@Glsplural@{#1}{#2}[]]}
852 \new@ifnextchar[\@Glsplural@{#1}{#2}]{\@Glsplural@{#1}{#2}[]]}

    Read in the final optional argument:
853 \def\@Glsplural@#1#2[#3]{%
854 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
855 \protected@edef\@glo@text{\glsentryplural{#2}}%
    Call \@gls@link
856 \@gls@link[#1]{#2}{%
857     \expandafter\makefirstuc\expandafter{\@glo@text}}%
858 }%
859 }

```

`\GLSplural` behaves like `\glsplural` except that the text is converted to uppercase.

`\GLSplural`

```

860 \newcommand*\@GLSplural{\@ifstar\@sGLSplural\@GLSplural}

    Define the starred form:
861 \newcommand*\@sGLSplural[1][\@GLSplural[hyper=false,#1]]{
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
862 \newcommand*\@GLSplural[2][\@GLSplural@{#1}{#2}[]]}
863 \new@ifnextchar[\@GLSplural@{#1}{#2}]{\@GLSplural@{#1}{#2}[]]}

    Read in the final optional argument:
864 \def\@GLSplural@#1#2[#3]{%
865 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
866 \protected@edef\@glo@text{\glsentryplural{#2}}%

```

Call `\@gls@link`

```
867 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%  
868 }%  
869 }
```

`\glsfirstplural` behaves like `\gls` except it always uses the value given by the `firstplural` key and it doesn't mark the entry as used.

`\glsfirstplural`

```
870 \newcommand*{\glsfirstplural}{\@ifstar\sglsfirstplural\glsfirstplural}
```

Define the starred form:

```
871 \newcommand*{\sglsfirstplural}[1][]{\@glsfirstplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
872 \newcommand*{\@glsfirstplural}[2][]{%  
873 \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
874 \def\@glsfirstplural@#1#2[#3]{%  
875 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%  
876 \protected@edef\@glo@text{\glstryfirstplural{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
876 \protected@edef\@glo@text{\glstryfirstplural{#2}}%
```

Call `\@gls@link`

```
877 \@gls@link[#1]{#2}{\@glo@text}%  
878 }%  
879 }
```

`\Glsfirstplural` behaves like `\glsfirstplural` except that the first letter is converted to uppercase.

`\Glsfirstplural`

```
880 \newcommand*{\Glsfirstplural}{\@ifstar\sglsfirstplural\Glsfirstplural}
```

Define the starred form:

```
881 \newcommand*{\sglsfirstplural}[1][]{\@Glsfirstplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
882 \newcommand*{\@Glsfirstplural}[2][]{%  
883 \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
884 \def\@Glsfirstplural@#1#2[#3]{%  
885 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%  
886 \protected@edef\@glo@text{\glstryfirstplural{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
886 \protected@edef\@glo@text{\glstryfirstplural{#2}}%
```

Call `\@gls@link`

```
887 \@gls@link[#1]{#2}{%  
888 \expandafter\makefirstuc\expandafter{\@glo@text}}%  
889 }%  
890 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
891 \newcommand*{\GLSfirstplural}{\@ifstar\@sGLSfirstplural\@GLSfirstplural}
    Define the starred form:
892 \newcommand*{\@sGLSfirstplural}[1][]{\@GLSfirstplural[hyper=false,#1]}
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
893 \newcommand*{\@GLSfirstplural}[2][]{%
894 \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2}[]}}
    Read in the final optional argument:
895 \def\@GLSfirstplural@#1#2[#3]{%
896 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
897 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
    Call \@gls@link
898 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
899 }%
900 }
```

`\glsname` behaves like `\gls` except it always uses the value given by the `name` key and it doesn't mark the entry as used.

`\glsname`

```
901 \newcommand*{\glsname}{\@ifstar\@sglsname\@glsname}
    Define the starred form:
902 \newcommand*{\@sglsname}[1][]{\@glsname[hyper=false,#1]}
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
903 \newcommand*{\@glsname}[2][]{%
904 \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2}[]}}
    Read in the final optional argument:
905 \def\@glsname@#1#2[#3]{%
906 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
907 \protected@edef\@glo@text{\glsentryname{#2}}%
    Call \@gls@link
908 \@gls@link[#1]{#2}{\@glo@text}%
909 }%
910 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
911 \newcommand*{\Glsname}{\@ifstar\@sGlsname\@Glsname}
```

Define the starred form:

```
912 \newcommand*{\@sGlsname}[1] [] {\@Glsname[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
913 \newcommand*{\@Glsname}[2] [] {%
```

```
914 \new@ifnextchar [{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]
```

Read in the final optional argument:

```
915 \def\@Glsname@#1#2[#3]{%
```

```
916 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
917 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
918 \@gls@link[#1]{#2}{%
```

```
919 \expandafter\makefirstuc\expandafter{\@glo@text}}%
```

```
920 }%
```

```
921 }
```

\GLSname behaves like \glsname except that the link text is converted to uppercase.

\GLSname

```
922 \newcommand*{\GLSname}{\@ifstar\@sGLSname\@GLSname}
```

Define the starred form:

```
923 \newcommand*{\@sGLSname}[1] [] {\@GLSname[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
924 \newcommand*{\@GLSname}[2] [] {%
```

```
925 \new@ifnextchar [{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}]
```

Read in the final optional argument:

```
926 \def\@GLSname@#1#2[#3]{%
```

```
927 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
928 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
929 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
```

```
930 }%
```

```
931 }
```

\glsdesc behaves like \gls except it always uses the value given by the description key and it doesn't mark the entry as used.

\glsdesc

```
932 \newcommand*{\glsdesc}{\@ifstar\@sglsdesc\@glsdesc}
```

Define the starred form:

```
933 \newcommand*{\@sglsdesc}[1] [] {\@glsdesc[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
934 \newcommand*{\@glsdesc}[2] [] {%
935 \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
936 \def\@glsdesc@#1#2[#3] {%
937 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
938 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
939 \@gls@link[#1]{#2}{\@glo@text}%
940 }%
941 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
942 \newcommand*{\Glsdesc}{\@ifstar\@sGlsdesc\@Glsdesc}
```

Define the starred form:

```
943 \newcommand*{\@sGlsdesc}[1] [] {\@Glsdesc[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
944 \newcommand*{\@Glsdesc}[2] [] {%
945 \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
946 \def\@Glsdesc@#1#2[#3] {%
947 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
948 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
949 \@gls@link[#1]{#2}{%
950 \expandafter\makefirstuc\expandafter{\@glo@text}}%
951 }%
952 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
953 \newcommand*{\GLSdesc}{\@ifstar\@sGLSdesc\@GLSdesc}
```

Define the starred form:

```
954 \newcommand*{\@sGLSdesc}[1] [] {\@GLSdesc[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
955 \newcommand*{\@GLSdesc}[2] [] {%
956 \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```

957 \def\@GLSdesc@#1#2[#3]{%
958 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
959 \protected@edef\@glo@text{\glsentrydesc{#2}}%
    Call \@gls@link
960 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
961 }%
962 }

```

`\glsdescplural` behaves like `\gls` except it always uses the value given by the `descriptionplural` key and it doesn't mark the entry as used.

`\glsdescplural`

```

963 \newcommand*\@glsdescplural{\@ifstar\@sglsdescplural\@glsdescplural}
    Define the starred form:
964 \newcommand*\@sglsdescplural[1][\@glsdescplural[hyper=false,#1]]{
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
965 \newcommand*\@glsdescplural[2][\@glsdescplural[hyper=false,#1]]{
966 \new@ifnextchar[\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}}
    Read in the final optional argument:
967 \def\@glsdescplural@#1#2[#3]{%
968 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
969 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
    Call \@gls@link
970 \@gls@link[#1]{#2}{\@glo@text}%
971 }%
972 }
    \Glsdescplural behaves like \glsdescplural except that the first letter is
    converted to uppercase.

```

`\Glsdescplural`

```

973 \newcommand*\@Glsdescplural{\@ifstar\@sGlsdescplural\@Glsdescplural}
    Define the starred form:
974 \newcommand*\@sGlsdescplural[1][\@Glsdescplural[hyper=false,#1]]{
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
975 \newcommand*\@Glsdescplural[2][\@Glsdescplural[hyper=false,#1]]{
976 \new@ifnextchar[\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2}[]}}
    Read in the final optional argument:
977 \def\@Glsdescplural@#1#2[#3]{%
978 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
979 \protected@edef\@glo@text{\glsentrydescplural{#2}}%

```

Call `\@gls@link`

```

980 \@gls@link[#1]{#2}{%
981   \expandafter\makefirstuc\expandafter{\@glo@text}}%
982 }%
983 }

```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```

984 \newcommand*{\GLSdescplural}{\@ifstar\@sGLSdescplural\@GLSdescplural}

```

Define the starred form:

```

985 \newcommand*{\@sGLSdescplural}[1][]{\@GLSdescplural[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

986 \newcommand*{\@GLSdescplural}[2][]{%
987   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2}[]}]

```

Read in the final optional argument:

```

988 \def\@GLSdescplural@#1#2[#3]{%
989   \glsdoifexists{#2}{\edef\@glo@type{\glstentrytype{#2}}}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

990 \protected@edef\@glo@text{\glstentrydescplural{#2}}%

```

Call `\@gls@link`

```

991 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
992 }%
993 }

```

`\glssymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glssymbol`

```

994 \newcommand*{\glssymbol}{\@ifstar\@sglssymbol\@glssymbol}

```

Define the starred form:

```

995 \newcommand*{\@sglssymbol}[1][]{\@glssymbol[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

996 \newcommand*{\@glssymbol}[2][]{%
997   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2}[]}]

```

Read in the final optional argument:

```

998 \def\@glssymbol@#1#2[#3]{%
999   \glsdoifexists{#2}{\edef\@glo@type{\glstentrytype{#2}}}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

1000 \protected@edef\@glo@text{\glstentrysymbol{#2}}%

```

Call `\@gls@link`

```

1001 \@gls@link[#1]{#2}{\@glo@text}%
1002 }%
1003 }

```

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
1004 \newcommand*\Glssymbol{\@ifstar\@sGlssymbol\@Glssymbol}
```

Define the starred form:

```
1005 \newcommand*\@sGlssymbol[1][\@Glssymbol[hyper=false,#1]]
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1006 \newcommand*\@Glssymbol[2][\%
```

```
1007 \new@ifnextchar[\@Glssymbol@{#1}{#2}]{\@Glssymbol@{#1}{#2}[]}}
```

Read in the final optional argument:

```
1008 \def\@Glssymbol@#1#2[#3]{\%
```

```
1009 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
1010 \protected@edef\@glo@text{\glstrytype{#2}}%
```

Call `\@gls@link`

```
1011 \@gls@link[#1]{#2}{\%
```

```
1012 \expandafter\makefirstuc\expandafter{\@glo@text}}%
```

```
1013 }%
```

```
1014 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
1015 \newcommand*\GLSsymbol{\@ifstar\@sGLSsymbol\@GLSsymbol}
```

Define the starred form:

```
1016 \newcommand*\@sGLSsymbol[1][\@GLSsymbol[hyper=false,#1]]
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1017 \newcommand*\@GLSsymbol[2][\%
```

```
1018 \new@ifnextchar[\@GLSsymbol@{#1}{#2}]{\@GLSsymbol@{#1}{#2}[]}}
```

Read in the final optional argument:

```
1019 \def\@GLSsymbol@#1#2[#3]{\%
```

```
1020 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
1021 \protected@edef\@glo@text{\glstrytype{#2}}%
```

Call `\@gls@link`

```
1022 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
```

```
1023 }%
```

```
1024 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the `symbolplural` key and it doesn't mark the entry as used.

`\glssymbolplural`

```
1025 \newcommand*\glssymbolplural{\@ifstar\@sglssymbolplural\@glssymbolplural}
```


Define the starred form:

```

1026 \newcommand*{\sglssymbolplural}[1] [] {\@glssymbolplural[hyper=false,#1]}
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
1027 \newcommand*{\@glssymbolplural}[2] [] {%
1028 \new@ifnextchar [{\@glssymbolplural@{#1}{#2}}]{\@glssymbolplural@{#1}{#2} []}}
    Read in the final optional argument:
1029 \def\@glssymbolplural@#1#2[#3]{%
1030 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
1031 \protected@edef\@glo@text{\glstrysymbolplural{#2}}%
    Call \@gls@link
1032 \@gls@link[#1]{#2}{\@glo@text}%
1033 }%
1034 }

```

\Glsymbolplural behaves like \glssymbolplural except that the first letter is converted to uppercase.

\Glsymbolplural

```

1035 \newcommand*{\Glsymbolplural}{\@ifstar\@sGlsymbolplural\@Glsymbolplural}
    Define the starred form:
1036 \newcommand*{\@sGlsymbolplural}[1] [] {\@Glsymbolplural[hyper=false,#1]}
    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment
1037 \newcommand*{\@Glsymbolplural}[2] [] {%
1038 \new@ifnextchar [{\@Glsymbolplural@{#1}{#2}}]{\@Glsymbolplural@{#1}{#2} []}}
    Read in the final optional argument:
1039 \def\@Glsymbolplural@#1#2[#3]{%
1040 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
1041 \protected@edef\@glo@text{\glstrysymbolplural{#2}}%
    Call \@gls@link
1042 \@gls@link[#1]{#2}{%
1043     \expandafter\makefirstuc\expandafter{\@glo@text}}}%
1044 }%
1045 }

```

\GLSsymbolplural behaves like \glssymbolplural except that the link text is converted to uppercase.

\GLSsymbolplural

```

1046 \newcommand*{\GLSsymbolplural}{\@ifstar\@sGLSsymbolplural\@GLSsymbolplural}
    Define the starred form:
1047 \newcommand*{\@sGLSsymbolplural}[1] [] {\@GLSsymbolplural[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

1048 \newcommand*{\GLSsymbolplural}[2] [] {%
1049 \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}}
    Read in the final optional argument:
1050 \def\@GLSsymbolplural@#1#2[#3] {%
1051 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
1052 \protected@edef\@glo@text{\glsentrysymbolplural{#2}}%
    Call \@gls@link
1053 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
1054 }%
1055 }

```

5.9.2 Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

Get the entry name (as specified by the `name` key when the entry was defined.) The argument is the label associated with the entry. Note that unless you used `name=false` in the `sanitize` package option you may get unexpected results if the `name` key contains any commands.

```

\glsentryname
1056 \newcommand*{\glsentryname}[1]{\csname glo@#1@name\endcsname}

\Glsentryname
1057 \newcommand*{\Glsentryname}[1]{%
1058 \protected@edef\@glo@text{\csname glo@#1@name\endcsname}%
1059 \expandafter\makefirstuc\expandafter{\@glo@text}}

```

Get the entry description (as specified by the `description` when the entry was defined.) The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the `description` key contained any commands.

```

\glsentrydesc
1060 \newcommand*{\glsentrydesc}[1]{\csname glo@#1@desc\endcsname}

\Glsentrydesc
1061 \newcommand*{\Glsentrydesc}[1]{%
1062 \protected@edef\@glo@text{\csname glo@#1@desc\endcsname}%
1063 \expandafter\makefirstuc\expandafter{\@glo@text}}

```

Plural form:

```

\glsentrydescplural
1064 \newcommand*{\glsentrydescplural}[1]{%
1065 \csname glo@#1@descplural\endcsname}

```

`\Glsentrydescplural`

```
1066 \newcommand*{\Glsentrydescplural}[1]{%
1067 \protected@edef\@glo@text{\csname glo@#1@descplural\endcsname}%
1068 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the entry text, as specified by the text key when the entry was defined.
The argument is the label associated with the entry:

`\glsentrytext`

```
1069 \newcommand*{\glsentrytext}[1]{\csname glo@#1@text\endcsname}
```

`\Glsentrytext`

```
1070 \newcommand*{\Glsentrytext}[1]{%
1071 \protected@edef\@glo@text{\csname glo@#1@text\endcsname}%
1072 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form:

`\glsentryplural`

```
1073 \newcommand*{\glsentryplural}[1]{\csname glo@#1@plural\endcsname}
```

`\Glsentryplural`

```
1074 \newcommand*{\Glsentryplural}[1]{%
1075 \protected@edef\@glo@text{\csname glo@#1@plural\endcsname}%
1076 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the symbol associated with this entry. The argument is the label associated with the entry. Note that unless you used `symbol=false` in the `sanitize` package option you may get unexpected results if the `symbol` key contained any commands.

`\glsentrysymbol`

```
1077 \newcommand*{\glsentrysymbol}[1]{\csname glo@#1@symbol\endcsname}
```

`\Glsentrysymbol`

```
1078 \newcommand*{\Glsentrysymbol}[1]{%
1079 \protected@edef\@glo@text{\csname glo@#1@symbol\endcsname}%
1080 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

`\glsentrysymbolplural`

```
1081 \newcommand*{\glsentrysymbolplural}[1]{%
1082 \csname glo@#1@symbolplural\endcsname}
```

`\Glsentrysymbolplural`

```
1083 \newcommand*{\Glsentrysymbolplural}[1]{%
1084 \protected@edef\@glo@text{\csname glo@#1@symbolplural\endcsname}%
1085 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the entry text to be used when the entry is first used in the document (as specified by the `first` key when the entry was defined.)

`\glsentryfirst`

```
1086 \newcommand*{\glsentryfirst}[1]{\csname glo@#1@first\endcsname}
```

`\Glsentryfirst`

```
1087 \newcommand*{\Glsentryfirst}[1]{%
1088 \protected@edef\@glo@text{\csname glo@#1@first\endcsname}%
1089 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form (as specified by the `firstplural` key when the entry was defined.)

`\glsentryfirstplural`

```
1090 \newcommand*{\glsentryfirstplural}[1]{%
1091 \csname glo@#1@firstpl\endcsname}
```

`\Glsentryfirstplural`

```
1092 \newcommand*{\Glsentryfirstplural}[1]{%
1093 \protected@edef\@glo@text{\csname glo@#1@firstpl\endcsname}%
1094 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Display the glossary type with which this entry is associated (as specified by the `type` key used when the entry was defined)

`\glsentrytype`

```
1095 \newcommand*{\glsentrytype}[1]{\csname glo@#1@type\endcsname}
```

Display the sort text used for this entry. Note that the sort key is sanitized, so unexpected results may occur if the sort key contained commands.

`\glsentrysort`

```
1096 \newcommand*{\glsentrysort}[1]{\csname glo@#1@sort\endcsname}
```

5.10 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
1097 \define@key{glossadd}{counter}{\def\@glo@counter{#1}}
1098 \define@key{glossadd}{format}{\def\@glo@format{#1}}
```

This key is only used by `\glsaddall`:

```
1099 \define@key{glossadd}{types}{\def\@glo@type{#1}}
```

`\glsadd[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: `counter` and `format` (the `types` key will be ignored).

`\glsadd`

```
1100 \newcommand*{\glsadd}[2][]{%
1101 \glsdoifexists{#2}{%
1102 \def\@glo@format{glsnumberformat}%
1103 \edef\@glo@counter{\csname glo@#2@counter\endcsname}%
1104 \setkeys{glossadd}{#1}%
1105 \edef\theglsentrycounter{\expandafter\noexpand\csname the\@glo@counter\endcsname}%
1106 \protected@edef\@glo@sort{\csname glo@#2@sort\endcsname}%
1107 \expandafter\glsaddall\expandafter{\theglsentrycounter}{\@glo@sort}%
1108 }
```

```

1107 \@gls@checkmkidxchars\@glo@sort
1108 \protected@edef\@glo@name{\csname glo@#2@name\endcsname}%
1109 \@gls@checkmkidxchars\@glo@name
1110 \protected@edef\@glo@name{\string\glsnamefont{\@glo@name}}%
1111 \protected@edef\@glo@desc{\csname glo@#2@desc\endcsname}%
1112 \@gls@checkmkidxchars\@glo@desc
1113 \protected@edef\@glo@symbol{\csname glo@#2@symbol\endcsname}%
1114 \@gls@checkmkidxchars\@glo@symbol
1115 \@set@glo@numformat\@glo@numfmt\@glo@counter\@glo@format
1116 \glossary[\csname glo@#2@type\endcsname]{%
1117 \@glo@sort\@gls@actualchar\string\glossaryentryfield
1118 {#2}{\@glo@name}{\@glo@desc}{\@glo@symbol}\@gls@encapchar
1119 \@glo@numfmt}%
1120 }}

```

`\glsaddall` [*<glossary list>*]

Add all terms defined for the listed glossaries (without displaying any text.) If `types` key is omitted, apply to all glossary types.

`\glsaddall`

```

1121 \newcommand*{\glsaddall}[1][]{%
1122 \def\@glo@type{\@glo@types}%
1123 \setkeys{glossadd}{#1}%
1124 \forallglsentries[\@glo@type]{\@glo@entry}{%
1125 \glsadd[#1]{\@glo@entry}}%
1126 }

```

5.11 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters.)

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `glossary-hypernav`. This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

Some of these lines are too long to fit on the page, but as I have temporarily disabled the comment character, I can't split the lines. If you want to see the code in full, have a look at `glossaries.sty`.

`\writeist`

```

1127 \newwrite\istfile

```

```

1128 \bgroup
1129 \catcode'\%12\relax
1130 \catcode'\ "12\relax
1131 \catcode'\|12\relax
1132 \catcode'\!12\relax
1133 \catcode'\?12\relax
1134 \gdef\@gls@actualchar{?}
1135 \gdef\@gls@encapchar{|}
1136 \gdef\@gls@levelchar{!}
1137 \gdef\@gls@quotechar{"}
1138 \gdef\writeist{\relax
1139 \protected@write\@auxout{}\string\@istfilename{\istfilename}}
1140 \openout\istfile=\istfilename
1141 \write\istfile{% makeindex style file created by the glossaries package}
1142 \write\istfile{% for document '\jobname' on \the\year-\the\month-\the\day}
1143 \write\istfile{actual '\@gls@actualchar'}
1144 \write\istfile{encap '\@gls@encapchar'}
1145 \write\istfile{level '\@gls@levelchar'}
1146 \write\istfile{quote '\@gls@quotechar'}
1147 \write\istfile{keyword "\string\glossaryentry"}
1148 \write\istfile{preamble "\string\glossarysection[\string\glossarytoctitle]{\string\glossary"}
1149 \write\istfile{postamble "\string\n\string\end{theglossary}\string\n\string\glossarypostamb"}
1150 \write\istfile{group_skip "\string\glsgroupskip\string\n"}
1151 \write\istfile{item_0 "\string\n"}
1152 \write\istfile{delim_0 "{\string\glossaryentrynumbers{\string\relax "}}
1153 \write\istfile{delim_t "\}\}"}
1154 \write\istfile{delim_n "\string\delimN "}
1155 \write\istfile{delim_r "\string\delimR "}
1156 \write\istfile{headings_flag 1}
1157 \write\istfile{heading_prefix "\string\glsgroupheading{"}}
1158 \write\istfile{heading_suffix "\}\}"}
1159 \write\istfile{symhead_positive "glsymbols"}
1160 \write\istfile{numhead_positive "glsnumbers"}
1161 \write\istfile{page_compositor "\glscompositor"}
1162 \noist}
1163 \egroup

```

The command `\noist` will suppress the creation of the `.ist` file (it simply redefines `\writeist` to do nothing.) Obviously you need to use this command before `\writeist` to have any effect. Since the `.ist` file should only be created once, `\noist` is called at the end of `\writeist`.

`\noist`

```

1164 \newcommand{\noist}{\let\writeist\relax}

```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the `<out-ext>` parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file.)

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where \TeX is trying to write to a non-existent file.) The relevant glossary must be defined prior to using `\@makeglossary`.

`\makeglossary`

```

1165 \newcommand*{\makeglossary}[1]{%
1166 \ifglossaryexists{#1}{%
1167 \edef\glo@out{\csname @glo@type@#1@out\endcsname}%
1168 \expandafter\newwrite\csname glo@#1@file\endcsname
1169 \edef\@glo@file{\csname glo@#1@file\endcsname}%
1170 \immediate\openout\@glo@file=\jobname.\glo@out
1171 \@gls@renewglossary
1172 \PackageInfo{glossaries}{Writing glossary file \jobname.\glo@out}
1173 \writeist
1174 }{\PackageError{glossaries}{%
1175 Glossary type ‘#1’ not defined}{New glossaries must be defined before
1176 using \string\makeglossary}}}
```

`\makeglossaries` will use `\makeglossary` for each glossary type that has been defined. New glossaries need to be defined before using `\makeglossary`, so have `\makeglossaries` redefine `\newglossary` to prevent it being used afterwards.

`\makeglossaries`

```

1177 \newcommand*{\makeglossaries}{%
1178 \@for\@glo@type:=\@glo@types\do{%
1179 \ifthenelse{equal{\@glo@type}{}}{}}{ }{%
1180 \makeglossary{\@glo@type}}}%
1181 \renewcommand*{\newglossary}[4][]{%
1182 \PackageError{glossaries}{New glossaries
1183 must be created before \string\makeglossaries}{You need
1184 to move \string\makeglossaries\space after all your
1185 \string\newglossary\space commands}}}%
1186 \let\@makeglossary\empty
1187 \let\makeglossary\empty}
```

The `\makeglossary` command is redefined to be identical to `\makeglossaries`. (This is done to reinforce the message that you must either use `\makeglossary` for all the glossaries or for none of them.)

`\makeglossary`

```

1188 \let\makeglossary\makeglossaries
```

5.12 Writing information to associated files

The `\glossary` command is redefined so that it takes an optional argument *<type>* to specify the glossary type (use `\glsdefaulttype` glossary by default). This shouldn't be used at user level as `\glslink` sets the correct format. The associated number should be stored in `\theglsentrycounter` before using `\glossary`.

`\glossary`

```

1189 \renewcommand*{\glossary}[1][\glsdefaulttype]{%
1190 \@glossary{#1}}
```

Define internal `\@glossary` to ignore its argument. This gets redefined in `\makeglossary`.

`\@glossary`

```

1191 \def\@glossary{#1}{\@bspack\begin@group\@sanitiz\@index}
```

This is a convenience command to set `\@glossary`. It is used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`@gls@renewglossary`

```
1192 \newcommand{\@gls@renewglossary}{%
1193 \gdef\@glossary[##1]{\@bsphack\beginingroup\@wrglossary{##1}}%
1194 \let\@gls@renewglossary\@empty
1195 }
```

The `\@wrglossary` command is redefined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in `\glslink`.)

`\@wrglossary`

```
1196 \renewcommand*{\@wrglossary}[2]{%
1197 \expandafter\protected@write\csname glo@#1@file\endcsname{}{%
1198 \string\glossaryentry{#2}{\theglentrycounter}}\endgroup\@esphack}
```

5.13 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[⟨key-val list⟩]`. If the `type` key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`\printglossary`

```
1199 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
1200 \def\@glo@type{\glsdefaulttype}%
1201 \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
1202 \def\glossarytoctitle{\glossarytitle}%
1203 \def\@glossarystyle{}%
1204 \setkeys{printgloss}{#1}%
1205 \bgroup
1206 \@glossarystyle
1207 \makeatletter
1208 \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
1209 \egroup
1210 }
```

The `\printglossaries` command will do `\printglossary` for each glossary type that has been defined. It is better to use `\printglossaries` rather than individual `\printglossary` commands to ensure that you don't forget any new glossaries you may have created. It also makes it easier to chop and change the value of the `acronym` package option. However, if you want to list the glossaries in a different order, or if you want to set the title or table of contents entry, or if you want to use different glossary styles for each glossary, you will need to use `\printglossary` explicitly for each glossary type.

`\printglossaries`

```
1211 \newcommand*{\printglossaries}{%
1212 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}}
```


The keys that can be used in the optional argument to `\printglossary` are as follows: The `type` key sets the glossary type.

```
1213 \define@key{printgloss}{type}{\def\@glo@type{#1}}
```

The `title` key sets the title used in the glossary section header. This overrides the title used in `\newglossary`.

```
1214 \define@key{printgloss}{title}{\def\glossarytitle{#1}}
```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```
1215 \define@key{printgloss}{toctitle}{\def\glossarytoctitle{#1}}
```

The `style` key sets the glossary style (but only for the given glossary.)

```
1216 \define@key{printgloss}{style}{%
1217 \@ifundefined{@glsstyle@#1}{\PackageError{glossaries}{Glossary
1218 style ‘#1’ undefined}{}}{%
1219 \def\@glossarystyle{\csname @glsstyle@#1\endcsname}}
```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
1220 \@ifundefined{theglossary}{%
1221 \newenvironment{theglossary}{}{}{%
1222 \PackageWarning{glossaries}{overriding ‘theglossary’ environment}%
1223 \renewenvironment{theglossary}{}{}}
```

The glossary header is given by `\glossaryheader`. This forms part of the glossary style, and must indicate what should appear immediately after the start of the `theglossary` environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don’t want a header row, the glossary style must redefine `\glossaryheader` to do nothing.

`\glossaryheader`

```
1224 \newcommand*{\glossaryheader}{}%
```

```
\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command governs how each entry row should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore `<symbol>`.

`\glossaryentryfield`

```
1225 \newcommand*{\glossaryentryfield}[5]{%
1226 \@glstarget{glo:#1}{#2} #4 #3. #5\par}
```

Within each glossary, the entries form 28 distinct groups which are determined by the first character of the sort key. There will be a maximum of 28 groups: symbols, numbers, and the 26 alphabetical groups A, ..., Z. The command `\glsgroupskip` specifies what to do between glossary groups. Glossary styles must redefine this command. (Note that `\glsgroupskip` only occurs between groups, not at the start or end of the glossary.)

`\glsgroupskip`

```
1227 \newcommand*{\glsgroupskip}{}%
```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glssymbols`, `glsnumbers`, A, ..., Z. Glossary styles must be redefined this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
1228 \newcommand*{\glsgroupheading}[1]{}
```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the `sort` key. For example, all entries belonging to one group could be defined so that the `sort` key starts with an a, while entries belonging to another group could be defined so that the `sort` key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa.)

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by `<label>`. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command.

`\glsgetgrouptitle`

```
1229 \newcommand*{\glsgetgrouptitle}[1]{%
1230 \@ifundefined{#1groupname}{#1}{\csname #1groupname\endcsname}}
```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`\glsgetgrouplabel`

```
1231 \newcommand*{\glsgetgrouplabel}[1]{%
1232 \ifthenelse{\equals{#1}{\glssymbolsgroupname}}{\glssymbols}{%
1233 \ifthenelse{\equals{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}
```

The command `\setentrycounter` sets the entry’s associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`\setentrycounter`

```
1234 \newcommand*{\setentrycounter}[1]{\def\glsetentrycounter{#1}}
```

The current glossary style can be set using `\glossarystyle{<style>}`.

`\glossarystyle`

```
1235 \newcommand*{\glossarystyle}[1]{%
1236 \@ifundefined{glsstyle@#1}{\PackageError{glossaries}{Glossary
1237 style ‘#1’ undefined}{}}{%
1238 \csname @glsstyle@#1\endcsname}}
```

New glossary styles can be defined using:

`\newglossarystyle{<name>}{<definition>}`

The `<definition>` argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [subsection 5.16](#) for the definitions of predefined styles.) Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

`\newglossarystyle`

```
1239 \newcommand*{\newglossarystyle}[2]{%
1240 \@ifundefined{glsstyle@#1}{%
1241 \expandafter\def\csname @glsstyle@#1\endcsname{#2}}{%
1242 \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}}
```

Glossary entries are encoded so that the second argument to `\glossaryentryfield` is always specified as `\glsnamefont{<name>}`. This allows the user to change the font used to display the name term without having to redefine `\glossaryentryfield`. The default uses the surrounding font, so in the list type styles (which place the name in the optional argument to `\item`) the name will appear in bold.

`\glsnamefont`

```
1243 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the `format` key in commands like `\glslink`. The default format is given by `\glshypernumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The `hyperref` package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the `hyperref` package.

`\glshypernumber`

```
1244 \@ifundefined{hyperlink}{%
1245 \def\glshypernumber#1{#1}}{%
1246 \def\glshypernumber#1{%
1247 \@delimR#1\delimR\delimR\}}
```

`\@delimR` displays a range of numbers for the counter whose name is given by `\@gls@counter` (which must be set prior to using `\glshypernumber`.)

`\@delimR`

```

1248 \def\@delimR#1\delimR #2\delimR #3\\{%
1249 \ifx\\#2\\%
1250   \@delimN{#1}%
1251 \else
1252   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
1253 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

1254 \def\@delimN#1{\@delimN#1\delimN \delimN\\}
1255 \def\@delimN#1\delimN #2\delimN#3\\{%
1256 \ifx\\#3\\%
1257   \@gls@numberlink{#1}%
1258 \else
1259   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
1260 \fi
1261 }

```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

1262 \def\@gls@numberlink#1{%
1263 \begingroup
1264 \toks@={}%
1265 \@gls@removespaces#1 \@nil
1266 \endgroup}

1267 \def\@gls@removespaces#1 #2\@nil{%
1268 \toks@=\expandafter{\the\toks@#1}%
1269 \ifx\\#2\\%
1270   \edef\x{\the\toks@}%
1271   \ifx\x\empty
1272     \else
1273       \hyperlink{\glsentrycounter.\the\toks@}{\the\toks@}%
1274     \fi
1275   \else
1276     \@gls@ReturnAfterFi{%
1277       \@gls@removespaces#2\@nil
1278     }%
1279   \fi
1280 }
1281 \long\def\@gls@ReturnAfterFi#1\fi{\fi#1}

```

The following commands will switch to the appropriate font, and create a hyperlink, if hyperlinks are supported. If hyperlinks are not supported, they will just display their argument in the appropriate font.

`\hyperrm`

```

1282 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}

```

`\hypersf`

```

1283 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}

```

`\hypertt`

```

1284 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

```

```

\hyperbf
1285 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}

\hypermd
1286 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}

\hyperit
1287 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}

\hypersl
1288 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}

\hyperup
1289 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}

\hypersc
1290 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
1291 \newcommand*\hyperemph[1]{\emph{\glshypernumber{#1}}}

```

5.14 Acronyms

If the acronym package option is used, a new glossary called `acronym` is created

```

1292 \ifglacronym
1293 \newglossary[alg]{acronym}{acr}{acn}{\acronymname}
    and \acronymtype is set to the name of this new glossary.
1294 \renewcommand{\acronymtype}{acronym}

```

In the event that the user redefines `\glsdisplay` and `\glsdisplayfirst`, the relevant commands for the new acronym glossary are set to match the format given by `\newacronym`. If you redefine `\newacronym` you may need to set these to something else.

```

1295 \defglsdisplay[acronym]{#1#4}
1296 \defglsdisplayfirst[acronym]{#1#4}
1297 \fi

```

```
\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrev⟩}{⟨long⟩}
```

This is a quick way of defining acronyms, all it does is call `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be `acronym` if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

```

\newacronym
1298 \newcommand{\newacronym}[4][{}]{%
1299 \newglossaryentry{#2}{type=\acronymtype,%
1300 name={#3},description={#4},text={#3},descriptionplural={#4s},%
1301 first={#4 (#3)},plural={#3s},%
1302 firstplural={\@glo@descplural\space (\@glo@plural)},%
1303 #1}}

```

New acronyms can only be defined in the preamble:

```
1304 \@onlypreamble{\newacronym}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the `description` key is changed).

Make a note of the keys that are used to store the long and short forms:

`\glsshortkey`

```
1305 \newcommand*{\glsshortkey}{text}
```

`\glsshortpluralkey`

```
1306 \newcommand*{\glsshortpluralkey}{plural}
```

`\glslongkey`

```
1307 \newcommand*{\glslongkey}{description}
```

`\glslongpluralkey`

```
1308 \newcommand*{\glslongpluralkey}{descriptionplural}
```

Using the default definitions, `\acrshort` is the same as `\glstext`, which means that it will print the abbreviation.

`\acrshort`

```
1309 \newcommand*{\acrshort}[2][]{%
1310 \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}}
1311 \def\@acrshort#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

`\Acrshort`

```
1312 \newcommand*{\Acrshort}[2][]{%
1313 \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}}
1314 \def\@Acrshort#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

`\ACRshort`

```
1315 \newcommand*{\ACRshort}[2][]{%
1316 \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}}
1317 \def\@ACRshort#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

`\acrlong` is set to `\glstdesc`, so it will print the long form, unless the `description` key has been set to something else.

`\acrlong`

```
1318 \newcommand*{\acrlong}[2][]{%
1319 \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}}
1320 \def\@acrlong#1#2[#3]{\@glstdesc@{#1}{#2}[#3]}
```

`\Acrlong`

```
1321 \newcommand*{\Acrlong}[2][]{%
1322 \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}}
1323 \def\@Acrlong#1#2[#3]{\@glstdesc@{#1}{#2}[#3]}
```

`\ACRlong`

```
1324 \newcommand*{\ACRlong}[2][]{%
1325 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2} []}}
1326 \def\@ACRlong#1#2[#3]{\@glstdesc@{#1}{#2}[#3]}
```

`\acrfull` is set to `\glsfirst`, so it should display the full form.

`\acrfull`

```
1327 \newcommand*{\acrfull}[2] [] {%
1328 \new@ifnextchar [{\@acrfull{#1}{#2}}]{\@acrfull{#1}{#2} [] }}
1329 \def\@acrfull#1#2[#3]{\@glsfirst@{#1}{#2}[#3]}
```

`\Acrfull`

```
1330 \newcommand*{\Acrfull}[2] [] {%
1331 \new@ifnextchar [{\@Acrfull{#1}{#2}}]{\@Acrfull{#1}{#2} [] }}
1332 \def\@Acrfull#1#2[#3]{\@Glsfirst@{#1}{#2}[#3]}
```

`\ACRfull`

```
1333 \newcommand*{\ACRfull}[2] [] {%
1334 \new@ifnextchar [{\@ACRfull{#1}{#2}}]{\@ACRfull{#1}{#2} [] }}
1335 \def\@ACRfull#1#2[#3]{\@GLSfirst@{#1}{#2}[#3]}
```

5.15 Additional predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
1336 \newcommand{\acronymfont}[1]{#1}
```

```
1337 \ifglsacrdescription
```

If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key.

```
1338 \ifglsacrfootnote
1339 \renewcommand{\newacronym}[4] [] {%
1340 \newglossaryentry{#2}{type=\acronymtype,%
1341 name={#3},%
1342 symbol={#4},%
1343 #1}}
```

Set up the commands to make a note of the keys to store the long and short forms:

```
1344 \def\glsshortkey{text}%
1345 \def\glsshortpluralkey{plural}%
1346 \def\glslongkey{symbol}%
1347 \def\glslongpluralkey{symbolplural}%

```

Set up short cuts. Short form:

```
1348 \def\@acrshort#1#2[#3]{\acronymfont{\@glstext@{#1}{#2}[#3]}}
1349 \def\@Acrshort#1#2[#3]{\acronymfont{\@Glstext@{#1}{#2}[#3]}}
1350 \def\@ACRshort#1#2[#3]{\acronymfont{\@GLStext@{#1}{#2}[#3]}}
```

Long form:

```
1351 \def\@acrlong#1#2[#3]{\@glssymbol@{#1}{#2}[#3]}
1352 \def\@Acrlong#1#2[#3]{\@Glsymbol@{#1}{#2}[#3]}
1353 \def\@ACRlong#1#2[#3]{\@GLSsymbol@{#1}{#2}[#3]}
```

Full form:

```
1354 \def\@acrfull#1#2[#3]{\@glssymbol@{#1}{#2}[#3]
1355 (\acronymfont{\@glstext@{#1}{#2}[#3]})}
1356 \def\@Acrfull#1#2[#3]{\@Glsymbol@{#1}{#2}[#3]
1357 (\acronymfont{\@Glstext@{#1}{#2}[#3]})}
1358 \def\@ACRfull#1#2[#3]{\@GLSsymbol@{#1}{#2}[#3]
1359 (\acronymfont{\@GLStext@{#1}{#2}[#3]})}
```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

1360 \def\glsdisplayfirst[\acronymtype]{%
1361 \acronymfont{#1}#4\noexpand\protect\noexpand\footnote{%
1362 \noexpand\protect\noexpand\glslink
1363 [\@gls@link@opts]{\@gls@link@label}{#3}}}%
1364 \def\glsdisplay[\acronymtype]{\acronymfont{#1}#4}%
1365 \ifglsacrsmallcaps
1366 \renewcommand*{\acronymfont}[1]{\textsc{#1}}
1367 \fi

```

Check for package option clash

```

1368 \ifglsacrdua
1369 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
1370 can’t both be set}{}%
1371 \fi
1372 \else

```

Footnote not required. Should the acronym always be expanded?

```

1373 \ifglsacrdua
1374 \ifglsacrsmallcaps
1375 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
1376 can’t both be set}{}%
1377 \fi
1378 \renewcommand{\newacronym}[4][\%
1379 \newglossaryentry{#2}{type=\acronymtype,%
1380 name={#4},%
1381 first={#4},%
1382 symbol={#3},%
1383 #1}}

```

Set up the commands to make a note of the keys to store the long and short forms:

```

1384 \def\glsshortkey{symbol}%
1385 \def\glsshortpluralkey{symbolplural}%
1386 \def\glslongkey{first}%
1387 \def\glslongpluralkey{plural}%

```

Set up short cuts. Short form:

```

1388 \def\@acrshort#1#2[#3]{\acronymfont{\@glssymbol@{#1}{#2}[#3]}}
1389 \def\@Acrshort#1#2[#3]{\acronymfont{\@GLssymbol@{#1}{#2}[#3]}}
1390 \def\@ACRshort#1#2[#3]{\acronymfont{\@GLSsymbol@{#1}{#2}[#3]}}

```

Long form:

```

1391 \def\@acrlong#1#2[#3]{\@glsfirst@{#1}{#2}[#3]}
1392 \def\@Acrlong#1#2[#3]{\@Glsfirst@{#1}{#2}[#3]}
1393 \def\@ACRlong#1#2[#3]{\@GLSfirst@{#1}{#2}[#3]}

```

Full form:

```

1394 \def\@acrfull#1#2[#3]{\@glsfirst@{#1}{#2}[#3]
1395 (\acronymfont{\@glssymbol@{#1}{#2}[#3]})}
1396 \def\@Acrfull#1#2[#3]{\@Glsfirst@{#1}{#2}[#3]
1397 (\acronymfont{\@glssymbol@{#1}{#2}[#3]})}
1398 \def\@ACRfull#1#2[#3]{\@GLSfirst@{#1}{#2}[#3]
1399 (\acronymfont{\@GLSsymbol@{#1}{#2}[#3]})}

```

Set display.


```

1400     \defglsdisplayfirst[\acronymtype]{#1#4}
1401     \defglsdisplay[\acronymtype]{#1#4}
1402     \else
        Store long form in first key and short form in text and symbol key.
1403     \renewcommand{\newacronym}[4][\%
1404     \newglossaryentry{#2}{type=\acronymtype,%
1405     name={#3},%
1406     first={#4},%
1407     symbol={#3},%
1408     #1}}

        Set up the commands to make a note of the keys to store the long and short forms:
1409 \def\glsshortkey{text}%
1410 \def\glsshortpluralkey{plural}%
1411 \def\glslongkey{first}%
1412 \def\glslongpluralkey{firstplural}%

        Set up short cuts. Short form:
1413     \def\@acrshort#1#2[#3]{\acronymfont{\@glstext@{#1}{#2}[#3]}}
1414     \def\@Acrshort#1#2[#3]{\acronymfont{\@Glstext@{#1}{#2}[#3]}}
1415     \def\@ACRshort#1#2[#3]{\acronymfont{\@GLStext@{#1}{#2}[#3]}}

        Long form:
1416     \def\@acrlong#1#2[#3]{\@glsfirst@{#1}{#2}[#3]}
1417     \def\@Acrlong#1#2[#3]{\@Glsfirst@{#1}{#2}[#3]}
1418     \def\@ACRlong#1#2[#3]{\@GLSfirst@{#1}{#2}[#3]}

        Full form:
1419     \def\@acrfull#1#2[#3]{\@glsfirst@{#1}{#2}[#3]
1420     (\acronymfont{\@glssymbol@{#1}{#2}[#3]})}
1421     \def\@Acrfull#1#2[#3]{\@Glsfirst@{#1}{#2}[#3]
1422     (\acronymfont{\@glssymbol@{#1}{#2}[#3]})}
1423     \def\@ACRfull#1#2[#3]{\@GLSfirst@{#1}{#2}[#3]
1424     (\acronymfont{\@GLSsymbol@{#1}{#2}[#3]})}

        Set display.
1425     \defglsdisplayfirst[\acronymtype]{#1#4 (\acronymfont{#3})}
1426     \defglsdisplay[\acronymtype]{\acronymfont{#1}#4}

        Set small caps and check for option clash
1427     \ifglsacrsmallcaps
1428     \renewcommand{\acronymfont}[1]{\textsc{#1}}
1429     \fi
1430     \fi
1431     \fi
1432 \else

        If here, acronyms do not require additional description.
1433 \ifglsacrfootnote

        If footnote package option is specified, set the first use to append the long form
        (stored in description) as a footnote. Use the description key to store the long form.
1434     \renewcommand{\newacronym}[4][\%
1435     \newglossaryentry{#2}{type=\acronymtype,%
1436     name={#3},%
1437     description={#4},%
1438     #1}}

```

Set up the commands to make a note of the keys to store the long and short forms:

```

1439 \def\glsshortkey{text}%
1440 \def\glsshortpluralkey{plural}%
1441 \def\glslongkey{description}%
1442 \def\glslongpluralkey{descriptionplural}%

```

Set display

```

1443 \defglsdisplayfirst[\acronymtype]{%
1444 \acronymfont{#1}#4\noexpand\protect\noexpand\footnote{%
1445 \noexpand\protect\noexpand\glslink
1446 [\@gls@link@opts]{\@gls@link@label}{#2}}}%
1447 \defglsdisplay[\acronymtype]{\acronymfont{#1}#4}%

```

Set up short cuts. Short form:

```

1448 \def\@acrshort#1#2[#3]{\acronymfont{\@glstext@{#1}{#2}[#3]}}
1449 \def\@Acrshort#1#2[#3]{\acronymfont{\@Glstext@{#1}{#2}[#3]}}
1450 \def\@ACRshort#1#2[#3]{\acronymfont{\@GLStext@{#1}{#2}[#3]}}

```

Long form:

```

1451 \def\@acrlong#1#2[#3]{\@glsdesc@{#1}{#2}[#3]}
1452 \def\@Acrlong#1#2[#3]{\@Glsdesc@{#1}{#2}[#3]}
1453 \def\@ACRlong#1#2[#3]{\@GLSdesc@{#1}{#2}[#3]}

```

Full form:

```

1454 \def\@acrfull#1#2[#3]{\@glsdesc@{#1}{#2}[#3]
1455 (\@glstext@{#1}{#2}[#3])}
1456 \def\@Acrfull#1#2[#3]{\@Glsdesc@{#1}{#2}[#3]
1457 (\@glstext@{#1}{#2}[#3])}
1458 \def\@ACRfull#1#2[#3]{\@GLSdesc@{#1}{#2}[#3]
1459 (\@GLStext@{#1}{#2}[#3])}

```

Redefine \acronymfont is small caps required

```

1460 \ifglsmallcaps
1461 \renewcommand*\acronymfont{1}{\textsc{#1}}%
1462 \fi

```

Check for option clash

```

1463 \ifglacrdua
1464 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
1465 can’t both be set}{}%
1466 \fi
1467 \else

```

No footnotes required.

```

1468 \ifglsmallcaps

```

Neither footnote nor description required. Use the symbol key to store the short form and first to store the long form.

```

1469 \renewcommand{\newacronym}[4][{}]{%
1470 \newglossaryentry{#2}{type=\acronymtype,%
1471 name={#3},%
1472 first={#4},%
1473 description={#4},%
1474 symbol={#3},%
1475 #1}}

```

Set up the commands to make a note of the keys to store the long and short forms:

```
1476 \def\glsshortkey{symbol}%
1477 \def\glsshortpluralkey{symbolplural}%
1478 \def\glslongkey{first}%
1479 \def\glslongpluralkey{firstplural}%
```

Change the display since first only contains long form.

```
1480 \defglsdisplayfirst[\acronymtype]{#1#4 (\acronymfont{#3})}
1481 \defglsdisplay[\acronymtype]{\acronymfont{#1}#4}
```

Change \acronymfont to small caps:

```
1482 \renewcommand*{\acronymfont}[1]{\textsc{#1}}
```

Set up short cuts. Short form:

```
1483 \def\@acrshort#1#2[#3]{\acronymfont{\@glstext@{#1}{#2}[#3]}}
1484 \def\@Acrshort#1#2[#3]{\acronymfont{\@Glstext@{#1}{#2}[#3]}}
1485 \def\@ACRshort#1#2[#3]{\acronymfont{\@GLStext@{#1}{#2}[#3]}}
```

Long form:

```
1486 \def\@acrlong#1#2[#3]{\@glsfirst@{#1}{#2}[#3]}
1487 \def\@Acrlong#1#2[#3]{\@Glsfirst@{#1}{#2}[#3]}
1488 \def\@ACRlong#1#2[#3]{\@GLSfirst@{#1}{#2}[#3]}
```

Full form:

```
1489 \def\@acrfull#1#2[#3]{\@glsfirst@{#1}{#2}[#3]
1490 (\acronymfont{\@glstext@{#1}{#2}[#3]})}
1491 \def\@Acrfull#1#2[#3]{\@Glsfirst@{#1}{#2}[#3]
1492 (\acronymfont{\@glstext@{#1}{#2}[#3]})}
1493 \def\@ACRfull#1#2[#3]{\@GLSfirst@{#1}{#2}[#3]
1494 (\acronymfont{\@GLStext@{#1}{#2}[#3]})}
```

check for option clash

```
1495 \ifglsacrdua
1496 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
1497 can’t both be set}{}%
1498 \fi
1499 \else
```

Should acronyms always be expanded?

```
1500 \ifglsacrdua
1501 \renewcommand{\newacronym}[4][\%
1502 \newglossaryentry{#2}{type=\acronymtype,%
1503 name={#3},%
1504 description={#4},%
1505 text={#4},%
1506 symbol={#3},%
1507 #1}}
```

Set up the commands to make a note of the keys to store the long and short forms:

```
1508 \def\glsshortkey{symbol}%
1509 \def\glsshortpluralkey{symbolplural}%
1510 \def\glslongkey{text}%
1511 \def\glslongpluralkey{plural}%
```

Set the display

```
1512 \defglsdisplayfirst[\acronymtype]{#1#4}
1513 \defglsdisplay[\acronymtype]{#1#4}
```

Set up short cuts. Short form:

```
1514 \def\@acrshort#1#2[#3]{\@glssymbol@{#1}{#2}[#3]}
1515 \def\@Acrshort#1#2[#3]{\@Glsymbol@{#1}{#2}[#3]}
1516 \def\@ACRshort#1#2[#3]{\@GLSsymbol@{#1}{#2}[#3]}
```

Long form:

```
1517 \def\@acrlong#1#2[#3]{\@glstext@{#1}{#2}[#3]}
1518 \def\@Acrlong#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
1519 \def\@ACRlong#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

Full form:

```
1520 \def\@acrfull#1#2[#3]{\@glstext@{#1}{#2}[#3]
1521 (\acronymfont{\@glssymbol@{#1}{#2}[#3]})}
1522 \def\@Acrfull#1#2[#3]{\@Glstext@{#1}{#2}[#3]
1523 (\acronymfont{\@Glsymbol@{#1}{#2}[#3]})}
1524 \def\@ACRfull#1#2[#3]{\@GLStext@{#1}{#2}[#3]
1525 (\acronymfont{\@GLSsymbol@{#1}{#2}[#3]})}
1526 \fi
1527 \fi
1528 \fi
1529 \fi
```

5.16 Predefined Glossary Styles

The glossaries bundle comes with some predefined glossary styles which are defined in the following packages:

```
1530 \RequirePackage{glossary-hypernav}
1531 \RequirePackage{glossary-list}
1532 \RequirePackage{glossary-long}
1533 \RequirePackage{glossary-super}
```

The default glossary style is set according to the style package option, but can be overridden by `\glossarystyle`.

```
1534 \glossarystyle{\@glossary@default@style}
```

6 Mfirstuc package

```
1535 \NeedsTeXFormat{LaTeX2e}
1536 \ProvidesPackage{mfirstuc}[2008/03/08 v1.0 (NLCT)]
```

`\makefirstuc` Syntax:

```
\makefirstuc{<text>}
```

Makes the first letter uppercase, but will skip initial control sequences if they are followed by a group and make the first thing in the group uppercase. Thus `\makefirstuc{abc}` will produce: *Abc*, `\makefirstuc{\ae bc}` will produce: *Æbc*, but `\makefirstuc{\emph{abc}}` will produce *Abc*. This is required by `\Gls` and `\Glspl`.

```
1537 \newif\if@glscs
1538 \def\makefirstuc#1{%
1539 \def\gls@argi{#1}%
1540 \ifx\gls@argi\@empty
1541 \else
1542 \protected@edef\@gls@tmp{\ #1}%
```

```

1543 \@onelevel@sanitize\@gls@tmp
1544 \expandafter\@gls@checkcs\@gls@tmp\relax\relax
1545 \if@glscs
1546   \@gls@getbody #1{}\@nil
1547   \ifx\@gls@rest\@empty
1548     \MakeUppercase #1%
1549   \else
1550     \@gls@body{\expandafter\@gls@makefirstuc\@gls@rest}%
1551   \fi
1552 \else
1553   \MakeUppercase #1%
1554 \fi
1555 \fi
1556 }

1557 \def\@gls@checkcs#1 #2#3\relax{%
1558 \def\@gls@argi{#1}\def\@gls@argii{#2}%
1559 \ifx\@gls@argi\@gls@argii
1560   \@glscstrue
1561 \else
1562   \@glscsfalse
1563 \fi
1564 }

```

Make first thing upper case:

```

1565 \def\@gls@makefirstuc#1{\MakeUppercase #1}

    Get the first grouped argument and stores in \@gls@body.
1566 \def\@gls@getbody#1#{\def\@gls@body{#1}\@gls@gobbletonil}

    Scoop up everything to \@nil and store in \@gls@rest:
1567 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}

```

6.0.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```

1568 \ProvidesPackage{glossary-hypernav}[2007/07/04 v1.01 (NLCT)]

```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [subsection 5.13](#).) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

`\glsnavhyperlink[⟨type⟩]{⟨label⟩}{⟨text⟩}`

This command makes `⟨text⟩` a hyperlink to the glossary group whose label is given by `⟨label⟩` for the glossary given by `⟨type⟩`.

`\glsnavhyperlink`

```

1569 \@ifundefined{hyperlink}{%
1570 \newcommand*\glsnavhyperlink}[3][\@#3]{%
1571 \newcommand*\glsnavhyperlink}[3][\@glo@type]{%
1572 \edef\gls@grplabel{#2}\edef\gls@grptitle{#3}%
1573 \hyperlink{glsn:#1@#2}{#3}}

```

`\glsnavhypertarget[⟨type⟩]{⟨label⟩}{⟨text⟩}`

This command makes `⟨text⟩` a hypertarget for the glossary group whose label is given by `⟨label⟩` in the glossary given by `⟨type⟩`.

`\glsnavhypertarget`

```
1574 \@ifundefined{hypertarget}{%
1575 \newcommand*{\glsnavhypertarget}[3][]{#3}{%
1576 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
1577 \hypertarget{glsn:#1#2}{#3}}}
```

The `\glsnavigation` command displays a simple glossary group navigation. The symbol and number elements are defined separately, so that they can be suppressed if need be. Note that this command will produce a link to all 28 groups, but some groups may not be defined if there are groups that do not contain any terms, in which case you will get an undefined hyperlink warning. Now for the whole navigation bit:

`\glsnavigation`

```
1578 \newcommand*{\glsnavigation}{%
1579 \glssymbolnav
1580 \glsnavhyperlink{A}{\glsgetgrouptitle{A}} \textbar\
1581 \glsnavhyperlink{B}{\glsgetgrouptitle{B}} \textbar\
1582 \glsnavhyperlink{C}{\glsgetgrouptitle{C}} \textbar\
1583 \glsnavhyperlink{D}{\glsgetgrouptitle{D}} \textbar\
1584 \glsnavhyperlink{E}{\glsgetgrouptitle{E}} \textbar\
1585 \glsnavhyperlink{F}{\glsgetgrouptitle{F}} \textbar\
1586 \glsnavhyperlink{G}{\glsgetgrouptitle{G}} \textbar\
1587 \glsnavhyperlink{H}{\glsgetgrouptitle{H}} \textbar\
1588 \glsnavhyperlink{I}{\glsgetgrouptitle{I}} \textbar\
1589 \glsnavhyperlink{J}{\glsgetgrouptitle{J}} \textbar\
1590 \glsnavhyperlink{K}{\glsgetgrouptitle{K}} \textbar\
1591 \glsnavhyperlink{L}{\glsgetgrouptitle{L}} \textbar\
1592 \glsnavhyperlink{M}{\glsgetgrouptitle{M}} \textbar\
1593 \glsnavhyperlink{N}{\glsgetgrouptitle{N}} \textbar\
1594 \glsnavhyperlink{O}{\glsgetgrouptitle{O}} \textbar\
1595 \glsnavhyperlink{P}{\glsgetgrouptitle{P}} \textbar\
1596 \glsnavhyperlink{Q}{\glsgetgrouptitle{Q}} \textbar\
1597 \glsnavhyperlink{R}{\glsgetgrouptitle{R}} \textbar\
1598 \glsnavhyperlink{S}{\glsgetgrouptitle{S}} \textbar\
1599 \glsnavhyperlink{T}{\glsgetgrouptitle{T}} \textbar\
1600 \glsnavhyperlink{U}{\glsgetgrouptitle{U}} \textbar\
1601 \glsnavhyperlink{V}{\glsgetgrouptitle{V}} \textbar\
1602 \glsnavhyperlink{W}{\glsgetgrouptitle{W}} \textbar\
1603 \glsnavhyperlink{X}{\glsgetgrouptitle{X}} \textbar\
1604 \glsnavhyperlink{Y}{\glsgetgrouptitle{Y}} \textbar\
1605 \glsnavhyperlink{Z}{\glsgetgrouptitle{Z}}}
```

The `\glssymbolnav` produces a simple navigation set of links for just the symbol and number groups. This is used at the start of `\glsnavigation`. If your glossary doesn't contain any symbol or navigation groups, you can redefine this command to do nothing.

`\glssymbolnav`

```

1606 \newcommand*{\glssymbolnav}{%
1607 \glsnahyperlink{glssymbols}{\glsgroup{title{glssymbols}} \textbar\
1608 \glsnahyperlink{glsnumbers}{\glsgroup{title{glsnumbers}} \textbar\
1609 }

```

6.0.2 List Style (glossary-list package)

The glossary-list package defines glossary styles that use the `description` environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```

1610 \ProvidesPackage{glossary-list}[2008/02/16 v1.03 (NLCT)]

```

The list glossary style uses the `description` environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. This is used as the default style for the glossaries package.

```

1611 \newglossarystyle{list}{%
1612 \renewenvironment{theglossary}{\begin{description}}{\end{description}}%
1613 \renewcommand*{\glossaryheader}{}%
1614 \renewcommand*{\glsgroupheading}[1]{}%
1615 \renewcommand*{\glossaryentryfield}[5]{%
1616 \item[\@glstarget{glo:##1}{##2}] ##3\glspostdescription\space ##5}%
1617 \renewcommand*{\glsgroupskip}{\indexspace}}

```

The `listgroup` style is like the `list` style, but the glossary groups have headings.

```

1618 \newglossarystyle{listgroup}{%
1619 \glossarystyle{list}%
1620 \renewcommand*{\glsgroupheading}[1]{\item[\glsgroup{title{##1}}]}

```

The `listhypergroup` style is like the `listgroup` style, but has a set of links to the groups at the start of the glossary.

```

1621 \newglossarystyle{listhypergroup}{%
1622 \glossarystyle{list}%
1623 \renewcommand*{\glossaryheader}{%
1624 \item[]\glsnavigation}%
1625 \renewcommand*{\glsgroupheading}[1]{%
1626 \item[\glsnahypertarget{##1}{\glsgroup{title{##1}}]}]}

```

The `altlist` glossary style is like the `list` style, but places the description on a new line.

```

1627 \newglossarystyle{altlist}{%
1628 \glossarystyle{list}%
1629 \renewcommand*{\glossaryentryfield}[5]{%
1630 \item[\@glstarget{glo:##1}{##2}]\mbox{} \newline ##3\glspostdescription\space ##5}%
1631 }

```

The `altlistgroup` glossary style is like the `altlist` style, but the glossary groups have headings.

```

1632 \newglossarystyle{altlistgroup}{%
1633 \glossarystyle{altlist}%
1634 \renewcommand*{\glsgroupheading}[1]{\item[\glsgroup{title{##1}}]}

```

The `altlisthypergroup` glossary style is like the `altlisthypergroup` style, but has a set of links to the groups at the start of the glossary.

```

1635 \newglossarystyle{altlisthypergroup}{%

```

```

1636 \glossarystyle{altlist}{%
1637 \renewcommand*{\glossaryheader}{%
1638 \item[]\glsnavigation}%
1639 \renewcommand*{\glsgroupheading}[1]{%
1640 \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}

```

The listdotted glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`.

`\glslistdottedwidth`

```

1641 \newlength\glslistdottedwidth
1642 \setlength{\glslistdottedwidth}{.5\linewidth}

```

Note that this style ignores the page numbers as well as the symbol.

```

1643 \newglossarystyle{listdotted}{%
1644 \glossarystyle{list}%
1645 \renewcommand*{\glossaryentryfield}[5]{%
1646 \item[]\makebox[\glslistdottedwidth][l]{\@glstarget{glo:##1}{##2}%
1647 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}}

```

6.0.3 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the glossary-long package used the longtable environment in the glossary.

```

1648 \ProvidesPackage{glossary-long}[2007/07/04 v1.01 (NLCT)]

```

Requires the longtable package:

```

1649 \RequirePackage{longtable}

```

This is a length that governs the width of the description column.

`\glsdescwidth`

```

1650 \newlength\glsdescwidth

```

This is a length that governs the width of the page list column.

`\glspagelistwidth`

```

1651 \newlength\glspagelistwidth

```

Default values:

```

1652 \setlength{\glsdescwidth}{0.6\linewidth}
1653 \setlength{\glspagelistwidth}{0.1\linewidth}

```

The long glossary style command which uses the longtable environment:

```

1654 \newglossarystyle{long}{%
1655 \renewenvironment{theglossary}{\begin{longtable}\lpf\glsdescwidth}}{%
1656 \end{longtable}}%
1657 \renewcommand*{\glossaryheader}{}%
1658 \renewcommand*{\glsgroupheading}[1]{}%
1659 \renewcommand*{\glossaryentryfield}[5]{%
1660 \@glstarget{glo:##1}{##2} & ##3\glspostdescription\space ##5\\}%
1661 \renewcommand*{\glsgroupskip}{ & \\}}

```


The longborder style is like the above, but with horizontal and vertical lines:

```
1662 \newglossarystyle{longborder}{%
1663 \glossarystyle{long}%
1664 \renewenvironment{theglossary}{%
1665 \begin{longtable}{|l|p{\glsdescwidth}|}{\end{longtable}}%
1666 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
1667 }
```

The longheader style is like the long style but with a header:

```
1668 \newglossarystyle{longheader}{%
1669 \glossarystyle{long}%
1670 \renewcommand*{\glossaryheader}{%
1671 \bfseries \entryname & \bfseries \descriptionname\\
1672 \endhead}}
```

The longheaderborder style is like the long style but with a header and border:

```
1673 \newglossarystyle{longheaderborder}{%
1674 \glossarystyle{longborder}%
1675 \renewcommand*{\glossaryheader}{%
1676 \hline\bfseries \entryname & \bfseries \descriptionname\\ \hline
1677 \endhead
1678 \hline\endfoot}}
```

The long3col style is like long but with 3 columns

```
1679 \newglossarystyle{long3col}{%
1680 \renewenvironment{theglossary}{\begin{longtable}{lp{\glsdescwidth}p{\gls pagelistwidth}}}{%
1681 \end{longtable}}%
1682 \renewcommand*{\glossaryheader}{}%
1683 \renewcommand*{\gls groupheading}[1]{}%
1684 \renewcommand*{\glossaryentryfield}[5]{%
1685 \@glstarget{glo:##1}{##2} & ##3 & ##5\\}%
1686 \renewcommand*{\gls groupskip}{ & & \\}}
```

The long3colborder style is like the long3col style but with a border:

```
1687 \newglossarystyle{long3colborder}{%
1688 \glossarystyle{long3col}%
1689 \renewenvironment{theglossary}{%
1690 \begin{longtable}{|l|p{\glsdescwidth}|p{\gls pagelistwidth}|}{%
1691 \end{longtable}}%
1692 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
1693 }
```

The long3colheader style is like long3col but with a header row:

```
1694 \newglossarystyle{long3colheader}{%
1695 \glossarystyle{long3col}%
1696 \renewcommand*{\glossaryheader}{%
1697 \bfseries\entryname&\bfseries\descriptionname&
1698 \bfseries\pagelistname\\ \endhead}%
1699 }
```

The long3colheaderborder style is like the above but with a border

```
1700 \newglossarystyle{long3colheaderborder}{%
1701 \glossarystyle{long3colborder}%
1702 \renewcommand*{\glossaryheader}{%
1703 \hline
1704 \bfseries\entryname&\bfseries\descriptionname&
```

```

1705 \bfseries\pagelistname\\hline\endhead
1706 \hline\endfoot}%
1707 }

```

The long4col style has four columns where the third column contains the value of the associated symbol key.

```

1708 \newglossarystyle{long4col}{%
1709 \renewenvironment{theglossary}{%
1710 \begin{longtable}{llll}{%
1711 \end{longtable}}%
1712 \renewcommand*{\glossaryheader}{}%
1713 \renewcommand*{\glsgroupheading}[1]{}%
1714 \renewcommand*{\glossaryentryfield}[5]{%
1715 \@glsstarget{glo:##1}{##2} & ##3 & ##4 & ##5\\}%
1716 \renewcommand*{\glsgroupskip}{ & & & \\}}

```

The long4colheader style is like long4col but with a header row.

```

1717 \newglossarystyle{long4colheader}{%
1718 \glossarystyle{long4col}%
1719 \renewcommand*{\glossaryheader}{%
1720 \bfseries\entryname&\bfseries\descriptionname&
1721 \bfseries \symbolname&
1722 \bfseries\pagelistname\\endhead}%
1723 }

```

The long4colborder style is like long4col but with a border.

```

1724 \newglossarystyle{long4colborder}{%
1725 \glossarystyle{long4col}%
1726 \renewenvironment{theglossary}{%
1727 \begin{longtable}{|ll|l|l|l|}{%
1728 \end{longtable}}%
1729 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
1730 }

```

The long4colheaderborder style is like the above but with a border.

```

1731 \newglossarystyle{long4colheaderborder}{%
1732 \glossarystyle{long4col}%
1733 \renewenvironment{theglossary}{%
1734 \begin{longtable}{|ll|l|l|l|}{%
1735 \end{longtable}}%
1736 \renewcommand*{\glossaryheader}{%
1737 \hline\bfseries\entryname&\bfseries\descriptionname&
1738 \bfseries \symbolname&
1739 \bfseries\pagelistname\\hline\endhead\hline\\endfoot}%
1740 }

```

6.0.4 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the glossary-super package use the supertabular environment.

```

1741 \ProvidesPackage{glossary-super}[2007/07/04 v1.01 (NLCT)]

```

Requires the supertabular package:

```

1742 \RequirePackage{supertabular}

```

The super glossary style uses the supertabular environment (it uses lengths defined in the glossary-long package.)

```

1743 \newglossarystyle{super}{%
1744 \renewenvironment{theglossary}{%
1745 \tablehead{}\tabletail}%
1746 \begin{supertabular}{lp{\glsdescwidth}}{%
1747 \end{supertabular}}%
1748 \renewcommand*{\glossaryheader}{}%
1749 \renewcommand*{\glsgroupheading}[1]{%
1750 \renewcommand*{\glossaryentryfield}[5]{%
1751 \@glstarget{glo:##1}{##2} & ##3\glspostdescription\space ##5\\}%
1752 \renewcommand*{\glsgroupskip}{ & \\}}

```

The superborder style is like the above, but with horizontal and vertical lines:

```

1753 \newglossarystyle{superborder}{%
1754 \glossarystyle{super}%
1755 \renewenvironment{theglossary}{%
1756 \tablehead{\hline}\tabletail{\hline}%
1757 \begin{supertabular}{|lp{\glsdescwidth}|}\end{supertabular}}%
1758 }

```

The superheader style is like the super style, but with a header:

```

1759 \newglossarystyle{superheader}{%
1760 \glossarystyle{super}%
1761 \renewenvironment{theglossary}{%
1762 \tablehead{\bfseries \entryname & \bfseries \descriptionname\\}%
1763 \tabletail{}}%
1764 \begin{supertabular}{lp{\glsdescwidth}}\end{supertabular}}%
1765 }

```

The superheaderborder style is like the super style but with a header and border:

```

1766 \newglossarystyle{superheaderborder}{%
1767 \glossarystyle{super}%
1768 \renewenvironment{theglossary}{%
1769 \tablehead{\hline\bfseries \entryname & \bfseries \descriptionname\\ \hline}%
1770 \tabletail{\hline}%
1771 \begin{supertabular}{|lp{\glsdescwidth}|}\end{supertabular}}%
1772 }

```

The super3col style is like the super style, but with 3 columns:

```

1773 \newglossarystyle{super3col}{%
1774 \renewenvironment{theglossary}{%
1775 \tablehead{}\tabletail}%
1776 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}{%
1777 \end{supertabular}}%
1778 \renewcommand*{\glossaryheader}{}%
1779 \renewcommand*{\glsgroupheading}[1]{%
1780 \renewcommand*{\glossaryentryfield}[5]{%
1781 \@glstarget{glo:##1}{##2} & ##3 & ##5\\}%
1782 \renewcommand*{\glsgroupskip}{ & & \\}}

```

The super3colborder style is like the super3col style, but with a border:

```

1783 \newglossarystyle{super3colborder}{%
1784 \glossarystyle{super3col}%
1785 \renewenvironment{theglossary}{%
1786 \tablehead{\hline}\tabletail{\hline}%

```

```

1787 \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}{%
1788 \end{supertabular}}}%
1789 }

```

The `super3colheader` style is like the `super3col` style but with a header row:

```

1790 \newglossarystyle{super3colheader}{%
1791 \glossarystyle{super3col}%
1792 \renewenvironment{theglossary}{%
1793 \tablehead{\bfseries\entryname&\bfseries\descriptionname&
1794 \bfseries\pagelistname\\}\tabletail{%
1795 \begin{supertabular}{|p{\glstdescwidth}|p{\glspagelistwidth}|}{%
1796 \end{supertabular}}}%
1797 }

```

The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```

1798 \newglossarystyle{super3colheaderborder}{%
1799 \glossarystyle{super3colborder}%
1800 \renewenvironment{theglossary}{%
1801 \tablehead{\hline
1802 \bfseries\entryname&\bfseries\descriptionname&
1803 \bfseries\pagelistname\\}\tabletail{\hline}%
1804 \tabletail{\hline}%
1805 \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}{%
1806 \end{supertabular}}}%
1807 }

```

The `super4col` glossary style has four columns, where the third column contains the value of the corresponding symbol key used when that entry was defined.

```

1808 \newglossarystyle{super4col}{%
1809 \renewenvironment{theglossary}{%
1810 \tablehead{}\tabletail{%
1811 \begin{supertabular}{|l|l|l|l|}{%
1812 \end{supertabular}}}%
1813 \renewcommand*{\glossaryheader}{}%
1814 \renewcommand*{\glsgroupheading}[1]{}%
1815 \renewcommand*{\glossaryentryfield}[5]{%
1816 \@glstarget{glo:##1}{##2} & ##3 & ##4 & ##5\\}%
1817 \renewcommand*{\glsgroupskip}{ & & & \\}

```

The `super4colheader` style is like the `super4col` but with a header row.

```

1818 \newglossarystyle{super4colheader}{%
1819 \glossarystyle{super4col}%
1820 \renewenvironment{theglossary}{%
1821 \tablehead{\bfseries\entryname&\bfseries\descriptionname&
1822 \bfseries\symbolname &
1823 \bfseries\pagelistname\\}\tabletail{%
1824 \begin{supertabular}{|l|l|l|l|}{%
1825 \end{supertabular}}}%
1826 }

```

The `super4colborder` style is like the `super4col` but with a border.

```

1827 \newglossarystyle{super4colborder}{%
1828 \glossarystyle{super4col}%
1829 \renewenvironment{theglossary}{%
1830 \tablehead{\hline}\tabletail{\hline}%

```

```

1831 \begin{supertabular}{|l|l|l|l|l|}{%
1832 \end{supertabular}}%
1833 }

```

The `super4colheaderborder` style is like the `super4col` but with a header and border.

```

1834 \newglossarystyle{super4colheaderborder}{%
1835 \glossarystyle{super4col}%
1836 \renewenvironment{theglossary}{%
1837 \tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
1838 \bfseries\symbolname &
1839 \bfseries\pagelistname\\}\tabletail{\hline}%
1840 \begin{supertabular}{|l|l|l|l|l|}{%
1841 \end{supertabular}}%
1842 }

```

6.1 Multi-Lingual Support

6.1.1 Babel Captions

Define babel captions if multi-lingual support is required, but the translator package is not loaded.

```

1843 \NeedsTeXFormat{LaTeX2e}
1844 \ProvidesPackage{glossaries-babel}[2008/02/22 v1.0 (NLCT)]

```

English:

```

1845 \addto\captionseenglish{%
1846 \renewcommand*{\glossaryname}{Glossary}%
1847 \renewcommand*{\acronymname}{Acronyms}%
1848 \renewcommand*{\entryname}{Notation}%
1849 \renewcommand*{\descriptionname}{Description}%
1850 \renewcommand*{\symbolname}{Symbol}%
1851 \renewcommand*{\pagelistname}{Page List}%
1852 \renewcommand*{\glssymbolsgroupname}{Symbols}%
1853 \renewcommand*{\glsnumbersgroupname}{Numbers}%
1854 }%

```

German (quite a few variations were suggested for German; I settled on the following):

```

1855 \addto\captionsgerman{%
1856 \renewcommand*{\glossaryname}{Glossar}%
1857 \renewcommand*{\acronymname}{Akronyme}%
1858 \renewcommand*{\entryname}{Bezeichnung}%
1859 \renewcommand*{\descriptionname}{Beschreibung}%
1860 \renewcommand*{\symbolname}{Symbol}%
1861 \renewcommand*{\pagelistname}{Seiten}%
1862 \renewcommand*{\glssymbolsgroupname}{Symbole}%
1863 \renewcommand*{\glsnumbersgroupname}{Zahlen}}

```

ngerman is identical to German:

```

1864 \addto\captionsgerman{%
1865 \renewcommand*{\glossaryname}{Glossar}%
1866 \renewcommand*{\acronymname}{Akronyme}%
1867 \renewcommand*{\entryname}{Bezeichnung}%
1868 \renewcommand*{\descriptionname}{Beschreibung}%
1869 \renewcommand*{\symbolname}{Symbol}%

```

```

1870 \renewcommand*{\pagelistname}{Seiten}%
1871 \renewcommand*{\glssymbolsgroupname}{Symbole}%
1872 \renewcommand*{\glsnumbersgroupname}{Zahlen}}

```

Italian:

```

1873 \addto\captionsitalian{%
1874 \renewcommand*{\glossaryname}{Glossario}%
1875 \renewcommand*{\acronymname}{Acronimi}%
1876 \renewcommand*{\entryname}{Nomenclatura}%
1877 \renewcommand*{\descriptionname}{Descrizione}%
1878 \renewcommand*{\symbolname}{Simbolo}%
1879 \renewcommand*{\pagelistname}{Elenco delle pagine}%
1880 \renewcommand*{\glssymbolsgroupname}{Simboli}%
1881 \renewcommand*{\glsnumbersgroupname}{Numeri}}

```

Dutch:

```

1882 \addto\captionsdutch{%
1883 \renewcommand*{\glossaryname}{Woordenlijst}%
1884 \renewcommand*{\acronymname}{Acroniemen}%
1885 \renewcommand*{\entryname}{Benaming}%
1886 \renewcommand*{\descriptionname}{Beschrijving}%
1887 \renewcommand*{\symbolname}{Symbool}%
1888 \renewcommand*{\pagelistname}{Pagina's}%
1889 \renewcommand*{\glssymbolsgroupname}{Symbolen}%
1890 \renewcommand*{\glsnumbersgroupname}{Cijfers}}

```

Spanish:

```

1891 \addto\captionsspanish{%
1892 \renewcommand*{\glossaryname}{Glosario}%
1893 \renewcommand*{\acronymname}{Siglas}%
1894 \renewcommand*{\entryname}{Entrada}%
1895 \renewcommand*{\descriptionname}{Descripci\'on}%
1896 \renewcommand*{\symbolname}{S\'mbolo}%
1897 \renewcommand*{\pagelistname}{Lista de p\'aginas}%
1898 \renewcommand*{\glssymbolsgroupname}{S\'mbolos}%
1899 \renewcommand*{\glsnumbersgroupname}{N\'umeros}}

```

French:

```

1900 \addto\captionsfrench{%
1901 \renewcommand*{\glossaryname}{Glossaire}%
1902 \renewcommand*{\acronymname}{Acronymes}%
1903 \renewcommand*{\entryname}{Terme}%
1904 \renewcommand*{\descriptionname}{Description}%
1905 \renewcommand*{\symbolname}{Symbole}%
1906 \renewcommand*{\pagelistname}{Pages}%
1907 \renewcommand*{\glssymbolsgroupname}{Symboles}%
1908 \renewcommand*{\glsnumbersgroupname}{Nombres}}

```

Danish:

```

1909 \addto\captionsdanish{%
1910 \renewcommand*{\glossaryname}{Ordliste}%
1911 \renewcommand*{\acronymname}{Akronymer}%
1912 \renewcommand*{\entryname}{Symbolforklaring}%
1913 \renewcommand*{\descriptionname}{Beskrivelse}%
1914 \renewcommand*{\symbolname}{Symbol}%
1915 \renewcommand*{\pagelistname}{Side}%

```

```

1916 \renewcommand*{\glssymbolsgroupname}{Symboler}%
1917 \renewcommand*{\glslnumbersgroupname}{Tal}}

Irish:
1918 \addto\captionssirish{%
1919 \renewcommand*{\glossaryname}{Gluais}%
1920 \renewcommand*{\acronymname}{Acrainmneacha}%

wasn't sure whether to go for Nóta (Note), Ciall ('Meaning', 'sense') or Brí ('Mean-
ing'). In the end I chose Ciall.
1921 \renewcommand*{\entryname}{Ciall}%
1922 \renewcommand*{\descriptionname}{Tuairisc}%

Again, not sure whether to use Comhartha/Comharthaí or Siombail/Siombaile,
so have chosen the former.
1923 \renewcommand*{\symbolname}{Comhartha}%
1924 \renewcommand*{\glssymbolsgroupname}{Comhartha\'}{\i}}%
1925 \renewcommand*{\pagelistname}{Leathanaigh}%
1926 \renewcommand*{\glslnumbersgroupname}{Uimhreacha}}

Hungarian:
1927 \addto\captionsmagyar{%
1928 \renewcommand*{\glossaryname}{Sz\'ojegyz\'ek}%
1929 \renewcommand*{\acronymname}{Bet\H uszavak}%
1930 \renewcommand*{\entryname}{Kifejez\'es}%
1931 \renewcommand*{\descriptionname}{Magyar\'azat}%
1932 \renewcommand*{\symbolname}{Jel\'ol\'es}%
1933 \renewcommand*{\pagelistname}{Oldalsz\'am}%
1934 \renewcommand*{\glssymbolsgroupname}{Jelek}%
1935 \renewcommand*{\glslnumbersgroupname}{Sz\'amjegyek}%
1936 }

```

6.1.2 Danish Dictionary

This is a dictionary file provided for use with the translator package.

```

1937 \ProvidesDictionary{glossaries-dictionary}{Danish}

Provide Danish translations:
1938 \providetranslation{Glossary}{Ordliste}
1939 \providetranslation{Acronyms}{Akronymer}
1940 \providetranslation{Notation (glossaries)}{Symbolforklaring}
1941 \providetranslation{Description (glossaries)}{Beskrivelse}
1942 \providetranslation{Symbol (glossaries)}{Symbol}
1943 \providetranslation{Page List (glossaries)}{Side}
1944 \providetranslation{Symbols (glossaries)}{Symboler}
1945 \providetranslation{Numbers (glossaries)}{Tal}

```

6.1.3 Dutch Dictionary

This is a dictionary file provided for use with the translator package.

```

1946 \ProvidesDictionary{glossaries-dictionary}{Dutch}

Provide Dutch translations:
1947 \providetranslation{Glossary}{Woordenlijst}
1948 \providetranslation{Acronyms}{Acroniemen}
1949 \providetranslation{Notation (glossaries)}{Benaming}

```

```

1950 \providetranslation{Description (glossaries)}{Beschrijving}
1951 \providetranslation{Symbol (glossaries)}{Symbool}
1952 \providetranslation{Page List (glossaries)}{Pagina's}
1953 \providetranslation{Symbols (glossaries)}{Symbolen}
1954 \providetranslation{Numbers (glossaries)}{Cijfers}

```

6.1.4 English Dictionary

This is a dictionary file provided for use with the translator package.

```

1955 \ProvidesDictionary{glossaries-dictionary}{English}

```

Provide English translations:

```

1956 \providetranslation{Glossary}{Glossary}
1957 \providetranslation{Acronyms}{Acronyms}
1958 \providetranslation{Notation (glossaries)}{Notation}
1959 \providetranslation{Description (glossaries)}{Description}
1960 \providetranslation{Symbol (glossaries)}{Symbol}
1961 \providetranslation{Page List (glossaries)}{Page List}
1962 \providetranslation{Symbols (glossaries)}{Symbols}
1963 \providetranslation{Numbers (glossaries)}{Numbers}

```

6.1.5 French Dictionary

This is a dictionary file provided for use with the translator package.

```

1964 \ProvidesDictionary{glossaries-dictionary}{French}

```

Provide French translations:

```

1965 \providetranslation{Glossary}{Glossaire}
1966 \providetranslation{Acronyms}{Acronymes}
1967 \providetranslation{Notation (glossaries)}{Terme}
1968 \providetranslation{Description (glossaries)}{Description}
1969 \providetranslation{Symbol (glossaries)}{Symbole}
1970 \providetranslation{Page List (glossaries)}{Pages}
1971 \providetranslation{Symbols (glossaries)}{Symboles}
1972 \providetranslation{Numbers (glossaries)}{Nombres}

```

6.1.6 German Dictionary

This is a dictionary file provided for use with the translator package.

```

1973 \ProvidesDictionary{glossaries-dictionary}{German}

```

Provide German translations (quite a few variations were suggested for German; I settled on the following):

```

1974 \providetranslation{Glossary}{Glossar}
1975 \providetranslation{Acronyms}{Akronyme}
1976 \providetranslation{Notation (glossaries)}{Bezeichnung}
1977 \providetranslation{Description (glossaries)}{Beschreibung}
1978 \providetranslation{Symbol (glossaries)}{Symbol}
1979 \providetranslation{Page List (glossaries)}{Seiten}
1980 \providetranslation{Symbols (glossaries)}{Symbole}
1981 \providetranslation{Numbers (glossaries)}{Zahlen}

```


6.1.7 Irish Dictionary

This is a dictionary file provided for use with the translator package.

```
1982 \ProvidesDictionary{glossaries-dictionary}{Irish}
```

Provide Irish translations:

```
1983 \providetranslation{Glossary}{Gluais}
1984 \providetranslation{Acronyms}{Acrainmneacha}
1985 \providetranslation{Notation (glossaries)}{Ciall}
1986 \providetranslation{Description (glossaries)}{Tuairisc}
1987 \providetranslation{Symbol (glossaries)}{Comhartha}
1988 \providetranslation{Page List (glossaries)}{Leathanaigh}
1989 \providetranslation{Symbols (glossaries)}{Comhartha'\{i}}
1990 \providetranslation{Numbers (glossaries)}{Uimhreacha}
```

6.1.8 Italian Dictionary

This is a dictionary file provided for use with the translator package.

```
1991 \ProvidesDictionary{glossaries-dictionary}{Italian}
```

Provide Italian translations:

```
1992 \providetranslation{Glossary}{Glossario}
1993 \providetranslation{Acronyms}{Acronimi}
1994 \providetranslation{Notation (glossaries)}{Nomenclatura}
1995 \providetranslation{Description (glossaries)}{Descrizione}
1996 \providetranslation{Symbol (glossaries)}{Simbolo}
1997 \providetranslation{Page List (glossaries)}{Elenco delle pagine}
1998 \providetranslation{Symbols (glossaries)}{Simboli}
1999 \providetranslation{Numbers (glossaries)}{Numeri}
```

6.1.9 Magyar Dictionary

This is a dictionary file provided for use with the translator package.

```
2000 \ProvidesDictionary{glossaries-dictionary}{Magyar}
```

Provide translations:

```
2001 \providetranslation{Glossary}{Sz\'ojegyz\'ek}
2002 \providetranslation{Acronyms}{Bet\H uszavak}
2003 \providetranslation{Notation (glossaries)}{Kifejez\'es}
2004 \providetranslation{Description (glossaries)}{Magyar\'azat}
2005 \providetranslation{Symbol (glossaries)}{Jel\'ol\'es}
2006 \providetranslation{Page List (glossaries)}{Oldalsz\'am}
2007 \providetranslation{Symbols (glossaries)}{Jelek}
2008 \providetranslation{Numbers (glossaries)}{Sz\'amjegyek}
```

6.1.10 Spanish Dictionary

This is a dictionary file provided for use with the translator package.

```
2009 \ProvidesDictionary{glossaries-dictionary}{Spanish}
```

Provide Spanish translations:

```
2010 \providetranslation{Glossary}{Glosario}
2011 \providetranslation{Acronyms}{Siglas}
2012 \providetranslation{Notation (glossaries)}{Entrada}
2013 \providetranslation{Description (glossaries)}{Descripci\'on}
```

2014 \providetranslation{Symbol (glossaries)}{S'\{i\}mbolo}
2015 \providetranslation{Page List (glossaries)}{Lista de p\`aginas}
2016 \providetranslation{Symbols (glossaries)}{S'\{i\}mbolos}
2017 \providetranslation{Numbers (glossaries)}{N\`umeros}