

Documented Code For glossaries v3.07

Nicola L.C. Talbot

Dickimaw Books

<http://www.dickimaw-books.com/>

2013-07-05

This is the documented code for the glossaries package. This bundle comes with the following documentation:

[glossariesbegin.pdf](#) If you are a complete beginner, start with “The glossaries package: a guide for beginners”.

[glossary2glossaries.pdf](#) If you are moving over from the obsolete glossary package, read “Upgrading from the glossary package to the glossaries package”.

[glossaries-user.pdf](#) For the main user guide, read “glossaries.sty v3.07: L^AT_EX2e Package to Assist Generating Glossaries”.

[mfirstuc-manual.pdf](#) The commands provided by the mfirstuc package are briefly described in “mfirstuc.sty: uppercasing first letter”.

[glossaries-code.pdf](#) This document is for advanced users wishing to know more about the inner workings of the glossaries package.

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

Contents

1	Main Package Code	3
1.1	Package Definition	3
1.2	Package Options	5
1.3	Default values	20
1.4	Xindy	29
1.5	Loops and conditionals	38
1.6	Defining new glossaries	40
1.7	Defining new entries	43
1.8	Resetting and unsetting entry flags	55
1.9	Loading files containing glossary entries	56
1.10	Using glossary entries in the text	57
1.10.1	Links to glossary entries	58
1.10.2	Displaying entry details without adding information to the glossary	109
1.11	Adding an entry to the glossary without generating text	116
1.12	Creating associated files	117
1.13	Writing information to associated files	126
1.14	Glossary Entry Cross-References	131
1.15	Displaying the glossary	132
1.16	Acronyms	146
1.17	Predefined acronym styles	150
1.18	Predefined Glossary Styles	165
1.19	Debugging Commands	165
1.20	Compatibility with version 2.07 and below	170
2	Mfirstuc Documented Code	170
3	Glossary Styles	172
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	172
3.2	In-line Style (glossary-inline.sty)	175
3.3	List Style (glossary-list.sty)	177
3.4	Glossary Styles using longtable (the glossary-long package)	180
3.5	Glossary Styles using longtable (the glossary-longragged package)	186
3.6	Glossary Styles using multicol (glossary-mcols.sty)	191
3.7	Glossary Styles using supertabular environment (glossary-super package)	195
3.8	Glossary Styles using supertabular environment (glossary-superragged package)	201
3.9	Tree Styles (glossary-tree.sty)	207
4	glossaries-compatible-207	215

5 Accessibility Support (glossaries-accsupp Code)	221
5.1 Defining Replacement Text	222
5.2 Accessing Replacement Text	225
5.3 Displaying the Glossary	237
5.4 Acronyms	238
5.5 Debugging Commands	241
6 Multi-Lingual Support	243
6.1 Babel Captions	243
6.2 Polyglossia Captions	249
6.3 Brazilian Dictionary	252
6.4 Danish Dictionary	252
6.5 Dutch Dictionary	253
6.6 English Dictionary	253
6.7 French Dictionary	253
6.8 German Dictionary	253
6.9 Irish Dictionary	254
6.10 Italian Dictionary	254
6.11 Magyar Dictionary	254
6.12 Polish Dictionary	255
6.13 Serbian Dictionary	255
6.14 Spanish Dictionary	255
Glossary	255
Change History	256
Index	266

1 Main Package Code

1.1 Package Definition

This package requires $\text{\LaTeX}2_{\epsilon}$.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2013/07/05 v3.07 (NLCT)]
```

Required packages:

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

7 \RequirePackage{datatool-base}
```

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

```
8 \RequirePackage{amsgen}
```

As from v3.0, now loading `etoolbox`:

```
9 \RequirePackage{etoolbox}
```

Check if doc has been loaded.

```
\if@gls@docloaded
```

```
10 \newif\if@gls@docloaded
11 \@ifpackageloaded{doc}%
12 {%
13   \@gls@docloadedtrue
14 }%
15 {%
16   \@ifclassloaded{nlctdoc}{\@gls@docloadedtrue}{\@gls@docloadedfalse}%
17 }

18 \if@gls@docloaded
```

It has been loaded, so some modifications need to be made to ensure both packages can work together.

```
\glsorg@glossary First, save the original behaviour of \glossary
```

```
19 \newcommand{\glsorg@glossary}{%
20   \@bsphack
21   \begingroup
22     \@sanitize \glsorg@wrglossary
23 }
```

```
\glsorg@wrglossary
```

```
24 \newcommand{\glsorg@wrglossary}[1]{%
25   \protected@write\@glossaryfile{}{%
26     \string \glossaryentry{#1}{\thepage}}%
27   \endgroup
28   \@esphack
29 }
```

```
\changes Now we need to redefine \changes so that it uses the original definition of
\glossary.
```

```
30 \let\glsorg@changes\changes
31 \renewcommand{\changes}[3]{%
32   \begingroup
33     \let\glossary\glsorg@glossary
34     \glsorg@changes{#1}{#2}{#3}%
35   \endgroup
36 }
```

`\PrintChanges` needs to use doc's version of `theglossary`, so save that.

\glsorg@theglossary

```
37 \let\glsorg@theglossary\theglossary
```

sorg@endtheglossary

```
38 \let\glsorg@endtheglossary\endtheglossary
```

\PrintChanges Now redefine \PrintChanges so that it uses the original theglossary environment.

```
39 \let\glsorg@PrintChanges\PrintChanges
40 \renewcommand{\PrintChanges}{%
41   \begingroup
42     \let\theglossary\glsorg@theglossary
43     \let\endtheglossary\glsorg@endtheglossary
44     \glsorg@PrintChanges
45   \endgroup
46 }
```

End of doc stuff.

```
47 \fi
```

1.2 Package Options

toc 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.

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

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

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

\@@glossarysec 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.

```
50 \ifcsundef{chapter}%
51   {\newcommand*\@@glossarysec}{section}}%
52   {\newcommand*\@@glossarysec}{chapter}}
```

section 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.

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

Determine whether or not to use numbered sections.

```

\@glossarysecstar
56 \newcommand*{\@glossarysecstar}{*}

\@glossaryseclabel
57 \newcommand*{\@glossaryseclabel}{}

\glsautoprefix Prefix to add before label if automatically generated:
58 \newcommand*{\glsautoprefix}{}

numberedsection
59 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
60 false,nolabel,autolabel}[nolabel]{%
61   \ifcase\nr\relax
62     \renewcommand*{\@glossarysecstar}{*}%
63     \renewcommand*{\@glossaryseclabel}{}%
64   \or
65     \renewcommand*{\@glossarysecstar}{}%
66     \renewcommand*{\@glossaryseclabel}{}%
67   \or
68     \renewcommand*{\@glossarysecstar}{}%
69     \renewcommand*{\@glossaryseclabel}{}%
70     \label{\glsautoprefix\glo@type}%
71   \fi
72 }

```

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The `list` style is defined in the accompanying package described in [subsection 1.18](#).)

```

ssary@default@style
73 \newcommand*{\@glossary@default@style}{list}

style 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 1.18.
74 \define@key{glossaries.sty}{style}{%
75 \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”:

```

lossaryentrynumbers
76 \newcommand*{\glossaryentrynumbers}[1]{#1\gls@save@numberlist{#1}}

```

`nonumberlist` 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).

```

77 \DeclareOptionX{nonumberlist}{%
78   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
79 }

```

`savenumberlist` Provide means to store the number list for entries.

```

80 \define@boolkey{glossaries.sty}{gls}{savenumberlist}[true]{}
81 \glssavenumberlistfalse

```

`o@seeautonumberlist`

```

82 \newcommand*{\glo@seeautonumberlist}{}

```

`seeautonumberlist` Automatically activates number list for entries containing the see key.

```

83 \DeclareOptionX{seeautonumberlist}{%
84   \renewcommand*{\glo@seeautonumberlist}{%
85     \def\glo@prefix{\glsnextpages}%
86   }%
87 }

```

`\@gls@loadlong`

```

88 \newcommand*{\@gls@loadlong}{\RequirePackage{glossary-long}}

```

`nolong` This option prevents from being loaded. This means that the glossary styles that use the `longtable` environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

```

89 \DeclareOptionX{nolong}{\renewcommand*{\@gls@loadlong}{} }

```

`\@gls@loadsuper` The package isn't loaded if isn't installed.

```

90 \IfFileExists{supertabular.sty}{%
91   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
92   \newcommand*{\@gls@loadsuper}{} }

```

`nosuper` This option prevents from being loaded. This means that the glossary styles that use the `supertabular` environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

```

93 \DeclareOptionX{nosuper}{\renewcommand*{\@gls@loadsuper}{} }

```

`\@gls@loadlist`

```

94 \newcommand*{\@gls@loadlist}{\RequirePackage{glossary-list}}

```

`nolist` This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

```

95 \DeclareOptionX{nolist}{\renewcommand*{\@gls@loadlist}{} }

```

`\@gls@loadtree`

```

96 \newcommand*{\@gls@loadtree}{\RequirePackage{glossary-tree}}

notree This option prevents from being loaded (to reduce overheads if required). Nat-
       urally, the styles defined in will not be available if this option is used.
97 \DeclareOptionX{notree}{\renewcommand*{\@gls@loadtree}{} }

nostyles Provide an option to suppress all the predefined styles (in the event that the
        user has custom styles that are not dependent on the predefined styles).
98 \DeclareOptionX{nostyles}{%
99   \renewcommand*{\@gls@loadlong}{} }%
100  \renewcommand*{\@gls@loadsuper}{} }%
101  \renewcommand*{\@gls@loadlist}{} }%
102  \renewcommand*{\@gls@loadtree}{} }%
103  \let\@glossary@default@style\relax
104 }

\glspostdescription The description terminator is given by \glspostdescription (except for the
                    3 and 4 column styles). This is a full stop by default. The spacefactor is ad-
                    justed in case the description ends with an upper case letter. (Patch provided
                    by Michael Pock.)
105 \newcommand*{\glspostdescription}{%
106   \ifglsnopostdot\else.\spacefactor\sfcode'\fi
107 }

nopostdot Boolean option to suppress post description dot
108 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
109 \glsnopostdotfalse

nogroupskip Boolean option to suppress vertical space between groups in the pre-defined
            styles.
110 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
111 \glsnogroupskipfalse

ucmark Boolean option to determine whether or not to use \MakeUppercase in defini-
        tion of \glossarymark
112 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
113 \glsucmarkfalse

entrycounter Defines a counter that can be used in the standard glossary styles to number
            each (main) entry. If true, this will define a counter called glossaryentry.
114 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
115 \glsentrycounterfalse

entrycounterwithin This option can be used to set a parent counter for glossaryentry. This option
                    automatically sets entrycounter=true.
116 \define@key{glossaries.sty}{counterwithin}{%

```



```

117 \renewcommand*{\@gls@counterwithin}{#1}%
118 \glsentrycountertrue
119 }

\@gls@counterwithin The default value is no parent counter:
120 \newcommand*{\@gls@counterwithin}{%

subentrycounter Define a counter that can be used in the standard glossary styles to number
each level 1 entry. If true, this will define a counter called glossarysubentry.
121 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{%
122 \glssubentrycounterfalse

sort Define the sort method: sort=standard (default), sort=def (order of definition)
or sort=use (order of use).
123 \define@choicekey{glossaries.sty}{sort}{standard,def,use}{%
124 \csname @gls@setupsort@#1\endcsname
125 }

@setupsort@standard Set up the macros for default sorting.
126 \newcommand*{\@gls@setupsort@standard}{%
Store entry information when it's defined.
127 \def\do@glo@storeentry{\@glo@storeentry}%
No count register required for standard sort.
128 \def\@gls@defsortcount##1{%
Sort according to sort key (\@glo@sort) if provided otherwise sort according
to the entry's name (\@glo@name).
129 \def\@gls@defsort##1##2{%
130 \ifx\@glo@sort\@glsdefaultsort
131 \let\@glo@sort\@glo@name
132 \fi

133 \@gls@sanitizesort
134 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
135 }%

Don't need to do anything when the entry is used.
136 \def\@gls@setsort##1{%
137 }

Set standard sort as the default:
138 \@gls@setupsort@standard

\glssortnumberfmt Format the number used as the sort key by sort=def and sort=use. Defaults to
six digit numbering.
139 \newcommand*\glssortnumberfmt[1]{%
140 \ifnum#1<100000 0\fi
141 \ifnum#1<10000 0\fi

```

```

142 \ifnum#1<1000 0\fi
143 \ifnum#1<100 0\fi
144 \ifnum#1<10 0\fi
145 \number#1%
146 }

```

`\@gls@setupsort@def` Set up the macros for order of definition sorting.

```

147 \newcommand*{\@gls@setupsort@def}{%
    Store entry information when it's defined.
148 \def\do@glo@storeentry{\@glo@storeentry}%
    Defined count register associated with the glossary.
149 \def\@gls@defsortcount##1{%
150     \expandafter\global
151     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
152 }%
    Increment count register associated with the glossary and use as the sort key.
153 \def\@gls@defsort##1##2{%
154     \expandafter\global\expandafter
155     \advance\csname glossary@##1@sortcount\endcsname by 1\relax
156     \expandafter\protected\edef\csname glo@##2@sort\endcsname{%
157         \expandafter\glssortnumberfmt
158         {\csname glossary@##1@sortcount\endcsname}}%
159 }%
    Don't need to do anything when the entry is used.
160 \def\@gls@setsort##1{%
161 }

```

`\@gls@setupsort@use` Set up the macros for order of use sorting.

```

162 \newcommand*{\@gls@setupsort@use}{%
    Don't store entry information when it's defined.
163 \let\do@glo@storeentry\@gobble
    Defined count register associated with the glossary.
164 \def\@gls@defsortcount##1{%
165     \expandafter\global
166     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
167 }%
    Initialise the sort key to empty.
168 \def\@gls@defsort##1##2{%
169     \expandafter\gdef\csname glo@##2@sort\endcsname{%
170 }%
    If the sort key hasn't been set, increment the counter associated with the glossary and set the sort key.
171 \def\@gls@setsort##1{%

```

Get the parent, if one exists

```
172 \edef\@glo@parent{\csname glo@##1@parent\endcsname}%
```

Set the information for the parent entry if not already done.

```
173 \ifx\@glo@parent\@empty
```

```
174 \else
```

```
175 \expandafter\@gls@setsort\expandafter{\@glo@parent}%
```

```
176 \fi
```

Set index information for this entry

```
177 \edef\@glo@type{\csname glo@##1@type\endcsname}%
```

```
178 \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
```

```
179 \ifx\@gls@tmp\@empty
```

```
180 \expandafter\global\expandafter
```

```
181 \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
```

```
182 \expandafter\protected@xdef\csname glo@##1@sort\endcsname{%
```

```
183 \expandafter\glssortnumberfmt
```

```
184 {\csname glossary@\@glo@type @sortcount\endcsname}}%
```

```
185 \@glo@storeentry{##1}%
```

```
186 \fi
```

```
187 }%
```

```
188 }
```

`\glsdefmain` Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`. The default extensions conflict if used with `doc`, so provide different extensions if `doc` loaded. (If these extensions are inappropriate, use `nomain` and manually define the main glossary with the desired extensions.)

```
189 \newcommand*\glsdefmain{%
```

```
190 \if@gls@docloaded
```

```
191 \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
```

```
192 \else
```

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

```
194 \fi
```

```
195 }
```

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 1.9](#)).

`\glsdefaulttype`

```
196 \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`

```
197 \newcommand*{\acronymtype}{\glsdefaulttype}
```

The `nomain` option suppress the creation of the main glossary.

```
198 \DeclareOptionX{nomain}{%
199   \let\glsdefaulttype\relax
200   \renewcommand*{\glsdefmain}{}%
201 }
```

`acronym` The `acronym` option sets an associated conditional which is used in [subsection 1.16](#) to determine whether or not to define a separate glossary for acronyms.

```
202 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
203   \DeclareAcronymList{acronym}%
204 }
```

`\@glsacronymlists` Comma-separated list of glossary labels indicating which glossaries contain acronyms. Note that `\SetAcronymStyle` must be used after adding labels to this macro.

```
205 \newcommand*{\@glsacronymlists}{}%
```

`\@addtoacronymlists`

```
206 \newcommand*{\@addtoacronymlists}[1]{%
207   \ifx\@glsacronymlists\@empty
208     \protected@xdef\@glsacronymlists{#1}%
209   \else
210     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
211   \fi
212 }
```

`\DeclareAcronymList` Identifies the named glossary as a list of acronyms and adds to the list. (Doesn't check if the glossary exists, but checks if label already in list. Use `\SetAcronymStyle` after identifying all the acronym lists.)

```
213 \newcommand*{\DeclareAcronymList}[1]{%
214   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
215 }
```

`\glsIfListOfAcronyms` `\glsIfListOfAcronyms{<label>}{<true part>}{<false part>}`

Determines if the glossary with the given label has been identified as being a list of acronyms.

```
216 \newcommand{\glsIfListOfAcronyms}[1]{%
217   \edef\@do@gls@islistofacronyms{%
218     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
219   \@do@gls@islistofacronyms
220 }
```

Internal command requires label and list to be expanded:

```
221 \newcommand{\@gls@islistofacronyms}[4]{%
222   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
223     \def\@before{##1}\def\@after{##2}}%
224   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
225   \ifx\@after\@nnil
```

Not found

```
226     #4%
227   \else
```

Found

```
228     #3%
229   \fi
230 }
```

`\if@gls@isacronymlist` Convenient boolean.

```
231 \newif\if@gls@isacronymlist
```

`\@checkisacronymlist` Sets the above boolean if argument is a label representing a list of acronyms.

```
232 \newcommand*\@gls@checkisacronymlist[1]{%
233   \glsIfListOfAcronyms{#1}%
234   {\@gls@isacronymlisttrue}{\@gls@isacronymlistfalse}%
235 }
```

`\SetAcronymLists` Sets the “list of acronyms” list. Argument must be a comma-separated list of glossary labels. (Doesn’t check at this point if the glossaries exists.)

```
236 \newcommand*\@SetAcronymLists[1]{%
237   \renewcommand*\@gls@acronymlists{#1}%
238 }
```

`acronymlists`

```
239 \define@key{glossaries.sty}{acronymlists}{%
240   \@addtoacronymlists{#1}%
241 }
```

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 1.6](#)).

`\glscounter`

```
242 \newcommand{\glscounter}{page}
```

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

```
243 \define@key{glossaries.sty}{counter}{%
244   \renewcommand*\@glscounter{#1}%
245 }
```

```

\@gls@nohyperlist
246 \newcommand*\@gls@nohyperlist{}

\DeclareNoHyperList
247 \newcommand*\GlsDeclareNoHyperList[1]{%
248   \ifdefempty\@gls@nohyperlist
249   {%
250     \renewcommand*\@gls@nohyperlist{#1}%
251   }%
252   {%
253     \appto\@gls@nohyperlist{, #1}%
254   }%
255 }

nohypertypes
256 \define@key{glossaries.sty}{nohypertypes}{%
257   \GlsDeclareNoHyperList{#1}%
258 }

```

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

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

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

```

\@gls@sanitizesort

```
262 \newcommand*{\@gls@sanitizesort}{\@onelevel@sanitize\@glo@sort}
```

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.

```
263 \define@boolkey[gls]{sanitize}{description}[true]{%
264 \ifgls@sanitize@description
265   \renewcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}%
266 \else
267   \renewcommand*{\@gls@sanitizedesc}{}%
268 \fi
269 }
```

Similarly for the name key:

```
270 \define@boolkey[gls]{sanitize}{name}[true]{%
271 \ifgls@sanitize@name
272   \renewcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}%
273 \else
274   \renewcommand*{\@gls@sanitizename}{}%
275 \fi}
```

and for the symbol key:

```
276 \define@boolkey[gls]{sanitize}{symbol}[true]{%
277 \ifgls@sanitize@symbol
278   \renewcommand*{\@gls@sanitizesymbol}{%
279 \@onelevel@sanitize\@glo@symbol}%
280 \else
281   \renewcommand*{\@gls@sanitizesymbol}{}%
282 \fi}
```

and for the sort key:

```
283 \define@boolkey[gls]{sanitize}{sort}[true]{%
284 \ifgls@sanitize@sort
285   \renewcommand*{\@gls@sanitizesort}{%
286 \@onelevel@sanitize\@glo@sort}%
287 \else
288   \renewcommand*{\@gls@sanitizesort}{}%
289 \fi}
```

sanitize 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:

```
290 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,
291 name=true]{%
292 \ifthenelse{\equal{#1}{none}}{}%
293 }
```

```

294 \renewcommand*{\@gls@sanitizedesc}{}%
295 \renewcommand*{\@gls@sanitizename}{}%
296 \renewcommand*{\@gls@sanitizesymbol}{}%
297 }%
298 {%
299 \setkeys{gls}{sanitize}{#1}}%
300 }

```

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

```

301 \define@boolkey{glossaries.sty}[gls]{translate}[true]{%

Set the default value:
302 \glstranslatefalse
303 \ifpackageloaded{translator}%
304 {\glstranslatetrue}%
305 {%
306 \ifpackageloaded{polyglossia}%
307 {\glstranslatetrue}%
308 {%
309 \ifpackageloaded{babel}{\glstranslatetrue}}}%
310 }%
311 }

```

indexonlyfirst Set whether to only index on first use.

```

312 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{%
313 \glsexindexonlyfirstfalse

```

hyperfirst Set whether or not terms should have a hyperlink on first use.

```

314 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{%
315 \glshyperfirsttrue

```

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

```

316 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
317 \ifthenelse{\boolean{glsacrdescription}}{}}%
318 {\renewcommand*{\@gls@sanitizedesc}{}}%
319 }

```

description Allow acronyms to have a description (needs to be set using the description key in the optional argument of `\newacronym`).

```

320 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
321 \renewcommand*{\@gls@sanitizesymbol}{}}%
322 }

```

smallcaps Define `\newacronym` to set the short form in small capitals.

```

323 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
324 \renewcommand*{\@gls@sanitizesymbol}{}}%
325 }

```


`smaller` Define `\newacronym` to set the short form using `\smaller` which obviously needs to be defined by loading the appropriate package.

```

326 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
327   \renewcommand*{\@gls@sanitizesymbol}{}%
328 }

```

`dua` Define `\newacronym` to always use the long forms (i.e. don't use acronyms)

```

329 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
330   \renewcommand*{\@gls@sanitizesymbol}{}%
331 }

```

`shortcuts` Define acronym shortcuts.

```

332 \define@boolkey{glossaries.sty}[glsacr]{shortcuts}[true]{%

```

`\glsorder` Stores the glossary ordering. This may either be “word” or “letter”. This passes the relevant information to `makeglossaries`. The default is word ordering.

```

333 \newcommand*{\glsorder}{word}

```

`\@glsorder` The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.

```

334 \newcommand*{\@glsorder}[1]{%

```

`order`

```

335 \define@choicekey{glossaries.sty}{order}{word,letter}{%
336   \def\glsorder{#1}}

```

`\ifglxindy` Provide boolean to determine whether `xindy` or `makeindex` will be used to sort the glossaries.

```

337 \newif\ifglxindy

```

The default is `makeindex`:

```

338 \glxindyfalse

```

Define package option to specify that `makeindex` will be used to sort the glossaries:

```

339 \DeclareOptionX{makeindex}{\glxindyfalse}

```

The `xindy` package option may have a value which in turn can be a key=value list. First define the keys for this sub-list. The boolean `glsnumbers` determines whether to automatically add the `glsnumbers` letter group.

```

340 \define@boolkey[glx]{xindy}{glsnumbers}[true]{%
341 \glx@xindy@glsnumberstrue

```

`\@xdy@main@language` Define what language to use for each glossary type (if a language is not defined for a particular glossary type the language specified for the main glossary is used.)

```

342 \def\@xdy@main@language{\@language}%

```

Define key to set the language

```
343 \define@key[glS]{xindy}{language}{\def\@xdy@main@language{#1}}
```

`\glS@codepage` Define the code page. If `\inputencodingname` is defined use that, otherwise have initialise with no codepage.

```
344 \ifcsundef{inputencodingname}{%
345   \def\glS@codepage{}}{%
346   \def\glS@codepage{\inputencodingname}
347 }
```

Define a key to set the code page.

```
348 \define@key[glS]{xindy}{codepage}{\def\glS@codepage{#1}}
```

Define package option to specify that xindy will be used to sort the glossaries:

```
349 \define@key{glossaries.sty}{xindy}[]{%
350   \glSxindytrue
351   \setkeys[glS]{xindy}{#1}%
352 }
```

`savewrites` The `savewrites` package option is provided to save on the number of write registers.

```
353 \define@boolkey{glossaries.sty}[glS]{savewrites}[true]{}
```

Set default:

```
354 \glSsavewritesfalse
```

`\GlossariesWarning` Prints a warning message.

```
355 \newcommand*\GlossariesWarning[1]{%
356   \PackageWarning{glossaries}{#1}%
357 }
```

`sariesWarningNoLine` Prints a warning message without the line number.

```
358 \newcommand*\GlossariesWarningNoLine[1]{%
359   \PackageWarningNoLine{glossaries}{#1}%
360 }
```

Define package option to suppress warnings

```
361 \DeclareOptionX{nowarn}{%
362   \renewcommand*\GlossariesWarning[1]{}%
363   \renewcommand*\GlossariesWarningNoLine[1]{}%
364 }
```

`compatible-2.07`

```
365 \define@boolkey{glossaries.sty}[glS]{compatible-2.07}[true]{}
366 \csname glScompatible-2.07false\endcsname
```

Process package options:

```
367 \ProcessOptionsX
```

If package is loaded, check to see if is installed, but only if translation is required.

```
368 \ifglstranslate
```

```
369 \ifpackageloaded{polyglossia}%
```

```
370 {%
```

polyglossia fakes babel so need to check for polyglossia first.

```
371 }%
```

```
372 {%
```

```
373 \ifpackageloaded{babel}%
```

```
374 {%
```

```
375 \IfFileExists{translator.sty}%
```

```
376 {%
```

```
377 \RequirePackage{translator}%
```

```
378 }%
```

```
379 }%
```

```
380 }%
```

```
381 {}
```

```
382 }
```

```
383 \fi
```

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 entries that give rise to a name{<section-level>.<n>.0} non-existent warning (e.g. `\gls[counter=chapter]{label}`).

```
384 \ifthenelse{\equal{\glscounter}{section}}{%
```

```
385 {%
```

```
386 \ifcsundef{chapter}{}%
```

```
387 {%
```

```
388 \let\@gls@old@chapter\@chapter
```

```
389 \def\@chapter[#1]#2{\@gls@old@chapter[{#1}]{#2}%
```

```
390 \ifcsundef{hyperdef}}{\hyperdef{section}{\thesection}}}%
```

```
391 }%
```

```
392 }%
```

```
393 {}
```

`\@gls@onlypremakeg` Some commands only have an effect when used before `\makeglossaries`. So define a list of commands that should be disabled after `\makeglossaries`

```
394 \newcommand*{\@gls@onlypremakeg}{}
```

`\@onlypremakeg` Adds the specified control sequence to the list of commands that must be disabled after `\makeglossaries`.

```

395 \newcommand*{\@onlypremakeg}[1]{%
396 \ifx\@gls@onlypremakeg\@empty
397   \def\@gls@onlypremakeg{#1}%
398 \else
399   \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
400   \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
401 \fi}

```

`\@disable@onlypremakeg` Disable all commands listed in `\@gls@onlypremakeg`

```

402 \newcommand*{\@disable@onlypremakeg}{%
403 \@for\@thiscs:=\@gls@onlypremakeg\do{%
404   \expandafter\@disable@premakecs\@thiscs%
405 }}

```

`\@disable@premakecs` Disables the given command.

```

406 \newcommand*{\@disable@premakecs}[1]{%
407   \def#1{\PackageError{glossaries}{\string#1\space may only be
408     used before \string\makeglossaries}{You can't use
409     \string#1\space after \string\makeglossaries}}%
410 }

```

1.3 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) so `\providecommand` is used.

Main glossary title:

`\glossaryname`

```
411 \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`

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

`\glssettoctitle` Sets the TOC title for the given glossary.

```

413 \newcommand*{\glssettoctitle}[1]{%
414 \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}

```

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`

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

```

\descriptionname
416 \providecommand*{\descriptionname}{Description}

\symbolname
417 \providecommand*{\symbolname}{Symbol}

\pagelistname
418 \providecommand*{\pagelistname}{Page List}

Labels for makeindex's symbol and number groups:

glssymbolsgroupname
419 \providecommand*{\glssymbolsgroupname}{Symbols}

glsnumbersgroupname
420 \providecommand*{\glsnumbersgroupname}{Numbers}

\glspluralsuffix The default plural is formed by appending \glspluralsuffix to the singular
form.
421 \newcommand*{\glspluralsuffix}{s}

\seename
422 \providecommand*{\seename}{see}

\andname
423 \providecommand*{\andname}{\&}

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

dglossarytocaptions If using , \glossaryname should be defined in terms of \translate, but if ba-
bel is also loaded, it will redefine \glossaryname whenever the language is set,
so override it. (Don't use \addto as doesn't define it.)
424 \newcommand*{\addglossarytocaptions}[1]{%
425   \ifcsundef{captions#1}{}%
426   {%
427     \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
428     \expandafter\toks@\expandafter{\@gls@tmp
429       \renewcommand*{\glossaryname}{\translate{Glossary}}}%
430     }%
431     \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
432   }%
433 }

434 \ifglstranslate

```

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

```

435 \@ifpackageloaded{translator}{%
436   \usedictionary{glossaries-dictionary}%
437   \addglossarytocaptions{portuges}%
438   \addglossarytocaptions{portuguese}%
439   \addglossarytocaptions{brazil}%
440   \addglossarytocaptions{brazilian}%
441   \addglossarytocaptions{danish}%
442   \addglossarytocaptions{dutch}%
443   \addglossarytocaptions{afrikaans}%
444   \addglossarytocaptions{english}%
445   \addglossarytocaptions{UKenglish}%
446   \addglossarytocaptions{USenglish}%
447   \addglossarytocaptions{american}%
448   \addglossarytocaptions{australian}%
449   \addglossarytocaptions{british}%
450   \addglossarytocaptions{canadian}%
451   \addglossarytocaptions{newzealand}%
452   \addglossarytocaptions{french}%
453   \addglossarytocaptions{frenchb}%
454   \addglossarytocaptions{francais}%
455   \addglossarytocaptions{acadian}%
456   \addglossarytocaptions{canadien}%
457   \addglossarytocaptions{german}%
458   \addglossarytocaptions{germanb}%
459   \addglossarytocaptions{austrian}%
460   \addglossarytocaptions{naustrian}%
461   \addglossarytocaptions{ngerman}%
462   \addglossarytocaptions{irish}%
463   \addglossarytocaptions{italian}%
464   \addglossarytocaptions{magyar}%
465   \addglossarytocaptions{hungarian}%
466   \addglossarytocaptions{polish}%
467   \addglossarytocaptions{spanish}%
468   \renewcommand*{\glstttitle}[1]{%
469     \ifthenelse{\equal{#1}{main}}{%
470       \translatelet{\glossarytoctitle}{Glossary}}{%
471       \ifthenelse{\equal{#1}{acronym}}{%
472         \translatelet{\glossarytoctitle}{Acronyms}}{%
473         \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}}%
474   \renewcommand*{\glossaryname}{\translate{Glossary}}%
475   \renewcommand*{\acronymname}{\translate{Acronyms}}%
476   \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
477   \renewcommand*{\descriptionname}{%
478     \translate{Description (glossaries)}}%
479   \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
480   \renewcommand*{\pagelistname}{%
481     \translate{Page List (glossaries)}}%
482   \renewcommand*{\glssymbolsgroupname}{%

```

```

483     \translate{Symbols (glossaries)}}}%
484     \renewcommand*{\glsnumbersgroupname}{%
485     \translate{Numbers (glossaries)}}}%
486 }{%

487     \@ifpackageloaded{polyglossia}%
488     {\RequirePackage{glossaries-polyglossia}}%
489     {%
490     \@ifpackageloaded{babel}{%
491     \RequirePackage{glossaries-babel}}}%
492 }}
493 \fi

```

`\nopostdesc` Provide a means to suppress description terminator for a given entry. (Useful for entries with no description.) Has no effect outside the glossaries.

```

494 \DeclareRobustCommand*\nopostdesc{}

```

`\@nopostdesc` Suppress next description terminator.

```

495 \newcommand*\@nopostdesc{%
496   \let\org@glspostdescription\glspostdescription
497   \def\glspostdescription{%
498     \let\glspostdescription\org@glspostdescription}%
499 }

```

`\glspar` Provide means of having a paragraph break in glossary entries

```

500 \newcommand{\glspar}{\par}

```

`\setStyleFile` Sets the style file. The relevant extension is appended.

```

501 \ifglxindy
502   \newcommand{\setStyleFile}[1]{%
503     \renewcommand{\istfilename}{#1.xdy}}
504 \else
505   \newcommand{\setStyleFile}[1]{%
506     \renewcommand{\istfilename}{#1.ist}}
507 \fi

```

This command only has an effect prior to using `\makeglossaries`.

```

508 \@onlypremakeg\setStyleFile

```

The name of the makeindex or xindy 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* `\makeglossaries`. As from v1.17, use `\setStyleFile` instead of directly redefining `\istfilename`.

`\istfilename`

```

509 \ifglxindy
510   \def\istfilename{\jobname.xdy}
511 \else
512   \def\istfilename{\jobname.ist}
513 \fi

```

The `makeglossaries` Perl script picks up this name from the auxiliary file. If the name ends with `.xdy` it calls `xindy` otherwise it calls `makeindex`. Since its not required by \TeX , `\@istfilename` ignores its argument.

`\@istfilename`

```
514 \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. As from 1.17, use `\glsSetCompositor` instead of directly redefining `\glscompositor`.

`\glscompositor`

```
515 \newcommand*\glscompositor}{.}

```

`\glsSetCompositor` Sets the compositor.

```
516 \newcommand*\glsSetCompositor}[1]{%
517   \renewcommand*\glscompositor}{#1}}

```

Only use before `\makeglossaries`

```
518 \@onlypremakeg\glsSetCompositor

```

(The page compositor is usually defined as a dash when using `makeindex`, but most of the standard counters used by \TeX use a full stop as the compositor, which is why I have used it as the default.) If `xindy` is used `\glscompositor` only affects the `arabic-page-numbers` location class.

`\@glsAlphacompositor` This is only used by `xindy`. It specifies the compositor to use when location numbers are in the form `\langle letter \rangle \langle compositor \rangle \langle number \rangle`. For example, if `\@glsAlphacompositor` is set to `“.”` then it allows locations such as `A.1` whereas if `\@glsAlphacompositor` is set to `“-”` then it allows locations such as `A-1`.

```
519 \newcommand*\@glsAlphacompositor}{\glscompositor}

```

`\glsSetAlphaCompositor` Sets the alpha compositor.

```
520 \ifglsxindy
521   \newcommand*\glsSetAlphaCompositor[1]{%
522     \renewcommand*\@glsAlphacompositor}{#1}}
523 \else
524   \newcommand*\glsSetAlphaCompositor[1]{%
525     \glsnoxindywarning\glsSetAlphaCompositor}
526 \fi

```

Can only be used before `\makeglossaries`

```
527 \@onlypremakeg\glsSetAlphaCompositor

```

`\gls@suffiXF` Suffix to use for a two page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
528 \newcommand*\gls@suffiXF}{}

```


`\glsSetSuffixF` Sets the suffix to use for a two page list.

```
529 \newcommand*\glsSetSuffixF}[1]{%
530   \renewcommand*\gls@suffixF{#1}}
```

Only has an effect when used before `\makeglossaries`

```
531 \@onlypremakeg\glsSetSuffixF
```

`\gls@suffixFF` Suffix to use for a three page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
532 \newcommand*\gls@suffixFF{-}
```

`\glsSetSuffixFF` Sets the suffix to use for a three page list.

```
533 \newcommand*\glsSetSuffixFF}[1]{%
534   \renewcommand*\gls@suffixFF{#1}%
535 }
```

`\glsnumberformat` 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”.

```
536 \ifcsundef{hyperlink}%
537 {%
538   \newcommand*\glsnumberformat}[1]{#1}%
539 }%
540 {%
541   \newcommand*\glsnumberformat}[1]{\glshypernumber{#1}}%
542 }
```

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`

```
543 \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`

```
544 \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 `\glossary preamble`.) 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 `\glossary preamble` before each `\printglossary`.

`\glossary preamble`

```
545 \newcommand*\glossary preamble{%
546   \csuse{glossary preamble@currentglossary}%
547 }
```

`\setglossary preamble`

```
\setglossary preamble[<type>]{<text>}
```

Code provided by Michael Pock.

```
548 \newcommand\setglossary preamble[2][\gls default type]{%
549   \ifglossary exists{#1}{%
550     \csgdef@glossary preamble@#1{#2}%
551   }{%
552     \PackageWarning{glossaries}{%
553       Glossary ‘#1’ is not defined%
554     }%
555   }%
556 }
```

The glossary postamble is given by `\glossary postamble`. This is provided to allow the user to add something after the end of the `\glossary` 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\glossary postamble{For a complete list of terms
see \cite{blah}\gdef\glossary preamble{}}
```

`\glossary postamble`

```
557 \newcommand*\glossary postamble{}
```

`\glossary section`

The sectioning command that starts a glossary is given by `\glossary section`. (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@glossary section`, otherwise it uses `\@glossary section`.

```
558 \newcommand*\glossary section[2][\@gls@title]{%
559   \def\@gls@title{#2}%
560   \ifcsundef{phantomsection}%
561   {%
562     \@glossary section{#1}{#2}%
```

```

563 }%
564 {%
565   \@pglossarysection{#1}{#2}%
566 }%
567 \glossarymark{\glossarytoctitle}%
568 }

```

`\glossarymark` Sets the header mark for the glossary. Takes the glossary short (TOC) title as the argument.

```

569 \ifcsundef{glossarymark}%
570 {%
571   \ifglsucmark
572     \newcommand{\glossarymark}[1]{%
573       \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
574     }
575   \else
576     \newcommand{\glossarymark}[1]{\@mkboth{#1}{#1}}
577   \fi
578 }%
579 {%
580   \GlossariesWarning{overriding \string\glossarymark}%
581   \@ifclassloaded{memoir}%
582   {
583     \ifglsucmark
584       \renewcommand{\glossarymark}[1]{%
585         \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
586       }
587     \else
588       \renewcommand{\glossarymark}[1]{%
589         \markboth{\memUHead{#1}}{\memUHead{#1}}%
590       }
591     \fi
592   }
593   {
594     \ifglsucmark
595       \renewcommand{\glossarymark}[1]{%
596         \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
597       }
598     \else
599       \renewcommand{\glossarymark}[1]{\@mkboth{#1}{#1}}
600     \fi
601   }
602 }

```

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`

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

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

`\@glossarysection`

```
605 \newcommand*\@glossarysection}[2]{%
606 \ifx\@glossarysecstar\@empty
607   \csname\@glossarysec\endcsname{#2}%
608 \else
609   \csname\@glossarysec\endcsname*{#2}%
610   \@gls@toc{#1}{\@glossarysec}%
611 \fi
612 \@@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`

```
613 \newcommand*\@pglossarysection}[2]{%
614 \glsclearpage
615 \phantomsection
616 \ifx\@glossarysecstar\@empty
617   \csname\@glossarysec\endcsname{#2}%
618 \else
619   \@gls@toc{#1}{\@glossarysec}%
620   \csname\@glossarysec\endcsname*{#2}%
621 \fi
622 \@@glossaryseclabel}
```

`\gls@doclearpage` 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.

```
623 \newcommand*\gls@doclearpage{%
624   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
625     {%
626       \ifcsundef{cleardoublepage}%
627       {%
628         \clearpage
629       }%
630     }%
631     \ifcsdef{if@openright}%
632     {%
633       \if@openright
634       \cleardoublepage
```

```

635         \else
636         \clearpage
637         \fi
638     }%
639     {%
640         \cleardoublepage
641     }%
642 }%
643 }%
644 {}%
645 }

```

`\glsclearpage` This just calls `\gls@doclearpage`, but it makes it easier to have a user command so that the user can override it.

```

646 \newcommand*{\glsclearpage}{\gls@doclearpage}

```

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
647 \newcommand*{\@gls@toc}[2]{%
648 \ifglstoc
649     \ifglslnumberline
650     \addcontentsline{toc}{#2}{\numberline{}}{#1}%
651 \else
652     \addcontentsline{toc}{#2}{#1}%
653 \fi
654 \fi}

```

1.4 Xindy

This section defines commands that only have an effect if `xindy` is used to sort the glossaries.

`\glsnoxindywarning` Issues a warning if `xindy` hasn't been specified. These warnings can be suppressed by redefining `\glsnoxindywarning` to ignore its argument

```

655 \newcommand*{\glsnoxindywarning}[1]{%
656 \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
657 }

```

`\@xdyattributes` Define list of attributes (`\string` is used in case the double quote character has been made active)

```

658 \ifglsxindy
659 \edef\@xdyattributes{\string"default\string"}%
660 \fi

```

```

\@xdyattributelist  Comma-separated list of attributes.
661 \ifglxsindy
662   \edef\@xdyattributelist{}%
663 \fi

\@xdylocref  Define list of markup location references.
664 \ifglxsindy
665   \def\@xdylocref{}
666 \fi

\@gls@ifinlist
667 \newcommand*\@gls@ifinlist[4]{%
668   \def\@do@ifinlist##1,##2\end@do@ifinlist{%
669     \def\@gls@listsuffix{##2}%
670     \ifx\@gls@listsuffix\@empty
671       #4%
672     \else
673       #3%
674     \fi
675   }%
676   \@do@ifinlist,##2,##1,\end@do@ifinlist
677 }

\GlsAddXdyCounters  Need to know all the counters that will be used in location numbers for Xindy.
Argument may be a single counter name or a comma-separated list of counter
names.
678 \ifglxsindy
679   \newcommand*\@xdycounters{\@glscounter}
680   \newcommand*\GlsAddXdyCounters[1]{%
681     \@for\@gls@ctr:=#1\do{%
        Check if already in list before adding.
682       \edef\@do@addcounter{%
683         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
684         {%
685           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
686             \noexpand\@gls@ctr}%
687         }%
688       }%
689       \@do@addcounter
690     }
691   }

  Only has an effect before \writeist:
692   \@onlypremakeg\GlsAddXdyCounters
693 \else
694   \newcommand*\GlsAddXdyCounters[1]{%
695     \glsnnoxindywarning\GlsAddXdyAttribute
696   }
697 \fi

```

d@gl saddx dy counters Counters must all be identified before adding attributes.

```

698 \newcommand*{@disabled@gl saddx dy counters}{%
699   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
700   can't be used after \string\GlsAddXdyAttribute}{Move all
701   occurrences of \string\GlsAddXdyCounters\space before the first
702   instance of \string\GlsAddXdyAttribute}%
703 }

```

\GlsAddXdyAttribute Adds an attribute.

```

704 \ifgl sxindy

```

First define internal command that adds an attribute for a given counter (2nd argument is the counter):

```

705   \newcommand*{@gl saddx dy attribute}[2]{%

```

Add to xindy attribute list

```

706     \edef{@xdyattributes}{@xdyattributes ^^J \string"#1\string" ^^J
707     \string"#2#1\string"}%

```

Add to xindy markup location.

```

708     \expandafter\toks@\expandafter{@xdylocref}%
709     \edef{@xdylocref}{the\toks@ ^^J%
710     (markup-locref
711     :open \string"\string~n%
712     \expandafter\string\csname gl sX#2X#1\endcsname
713     \string" ^^J
714     :close \string"\string" ^^J
715     :attr \string"#2#1\string")}%

```

Define associated attribute command `\gl sX<counter>X<attribute>{\Hprefix}\{<n>}`

```

716     \expandafter\gdef\csname gl sX#2X#1\endcsname##1##2{%
717     \setentrycounter{##1}{#2}\csname #1\endcsname{##2}%
718     }%
719   }

```

High-level command:

```

720   \newcommand*{\GlsAddXdyAttribute}[1]{%

```

Add to comma-separated attribute list

```

721     \ifx{@xdyattributelist}\empty
722     \edef{@xdyattributelist}{#1}%
723   \else
724     \edef{@xdyattributelist}{@xdyattributelist,#1}%
725   \fi

```

Iterate through all specified counters and add counter-dependent attributes:

```

726     \@for\@this@counter:=\@xdycounters\do{%
727     \protected@edef\gl s@do@addx dy attribute{%
728     \noexpand\@gl saddx dy attribute{#1}{\@this@counter}%
729     }
730     \gl s@do@addx dy attribute
731   }%

```

All occurrences of `\GlsAddXdyCounters` must be used before this command

```
732 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
733 }
```

Only has an effect before `\writeist`:

```
734 \@onlypremakeg\GlsAddXdyAttribute
735 \else
736 \newcommand*\GlsAddXdyAttribute[1]{%
737 \glsnoxindywarning\GlsAddXdyAttribute}
738 \fi
```

`redefinedattributes` Add known attributes for all defined counters

```
739 \ifglsxindy
740 \newcommand*\@gls@addpredefinedattributes{%
741 \GlsAddXdyAttribute{glsnumberformat}
742 \GlsAddXdyAttribute{textrm}
743 \GlsAddXdyAttribute{textsf}
744 \GlsAddXdyAttribute{texttt}
745 \GlsAddXdyAttribute{textbf}
746 \GlsAddXdyAttribute{textmd}
747 \GlsAddXdyAttribute{textit}
748 \GlsAddXdyAttribute{textup}
749 \GlsAddXdyAttribute{textsl}
750 \GlsAddXdyAttribute{textsc}
751 \GlsAddXdyAttribute{emph}
752 \GlsAddXdyAttribute{glshypernumber}
753 \GlsAddXdyAttribute{hyperrrm}
754 \GlsAddXdyAttribute{hypersf}
755 \GlsAddXdyAttribute{hypertt}
756 \GlsAddXdyAttribute{hyperbf}
757 \GlsAddXdyAttribute{hypermd}
758 \GlsAddXdyAttribute{hyperit}
759 \GlsAddXdyAttribute{hyperup}
760 \GlsAddXdyAttribute{hypersl}
761 \GlsAddXdyAttribute{hypersc}
762 \GlsAddXdyAttribute{hyperemph}
763 }
764 \else
765 \let\@gls@addpredefinedattributes\relax
766 \fi
```

`\@xdyuseralphabets` List of additional alphabets

```
767 \def\@xdyuseralphabets{}
```

`\GlsAddXdyAlphabet` `\GlsAddXdyAlphabet{<name>}{<definition>}` adds a new alphabet called `<name>`.
The definition must use xindy syntax.

```
768 \ifglsxindy
769 \newcommand*\GlsAddXdyAlphabet[2]{%
770 \edef\@xdyuseralphabets{%
```



```

771 \@xdyuseralphabets ^^J
772 (define-alphabet "#1" (#2))}}
773 \else
774 \newcommand*{\GlsAddXdyAlphabet}[2]{%
775 \glsnoinxdywarning\GlsAddXdyAlphabet}
776 \fi

```

This code is only required for xindy:

```

777 \ifglsxindy

```

`\s@xdy@locationlist` List of predefined location names.

```

778 \newcommand*{\@glx@xdy@locationlist}{%
779 roman-page-numbers,%
780 Roman-page-numbers,%
781 arabic-page-numbers,%
782 alpha-page-numbers,%
783 Alpha-page-numbers,%
784 Appendix-page-numbers,%
785 arabic-section-numbers%
786 }

```

Each location class *<name>* has the format stored in `\@glx@xdy@Lclass@<name>`.
Set up predefined formats.

`\@roman-page-numbers` Lower case Roman numerals (i, ii, ...). In the event that `\roman` has been redefined to produce a fancy form of roman numerals, attempt to work out how it will be written to the output file.

```

787 \protected@edef\@glx@roman{\@roman{0}\string"
788 \string"roman-numbers-lowercase\string" :sep \string"}}%
789 \@onelevel@sanitize\@glx@roman
790 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
791 :sep \string"}%
792 \@onelevel@sanitize\@tmp
793 \ifx\@tmp\@glx@roman
794 \expandafter
795 \edef\csname @glx@xdy@Lclass@roman-page-numbers\endcsname{%
796 \string"roman-numbers-lowercase\string"%
797 }%
798 \else
799 \expandafter
800 \edef\csname @glx@xdy@Lclass@roman-page-numbers\endcsname{
801 :sep \string"\@glx@roman\string"%
802 }%
803 \fi

```

`\@Roman-page-numbers` Upper case Roman numerals (I, II, ...).

```

804 \expandafter\def\csname @glx@xdy@Lclass@Roman-page-numbers\endcsname{%
805 \string"roman-numbers-uppercase\string"%
806 }%

```

arabic-page-numbers Arabic numbers (1, 2, ...).

```

807 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
808 \string"arabic-numbers\string"%
809 }%

```

@alpha-page-numbers Lower case alphabetical (a, b, ...).

```

810 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
811 \string"alpha\string"%
812 }%

```

@Alpha-page-numbers Upper case alphabetical (A, B, ...).

```

813 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
814 \string"ALPHA\string"%
815 }%

```

appendix-page-numbers Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given by \@glsAlphacompositor.

```

816 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
817 \string"ALPHA\string"
818 :sep \string"\@glsAlphacompositor\string"
819 \string"arabic-numbers\string"%
820 }

```

arabic-section-numbers Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by \@glscompositor.

```

821 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
822 \string"arabic-numbers\string"
823 :sep \string"\@glscompositor\string"
824 \string"arabic-numbers\string"%
825 }%

```

@xdyuserlocationdefs List of additional location definitions (separated by ^^J)

```

826 \def\@xdyuserlocationdefs{}

```

@xdyuserlocationnames List of additional user location names

```

827 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

828 \fi

```

\GlsAddXdyLocation [*<prefix-loc>*] {*<name>*} {*<definition>*} Define a new location called *<name>*. The definition must use xindy syntax. (Note that this doesn't check to see if the location is already defined. That is left to xindy to complain about.)

```

829 \ifglsxindy
830 \newcommand*\GlsAddXdyLocation[3][{}]{%
831 \def\@gls@tmp{#1}%

```

```

832 \ifx\@gls@tmp\@empty
833 \edef\@xdyuserlocationdefs{%
834 \@xdyuserlocationdefs ^^J%
835 (define-location-class \string"#2\string"^^J\space\space
836 \space(:sep \string"{}\glsoopenbrace\string" #3
837 :sep \string"\glsclosebrace\string"))
838 }%
839 \else
840 \edef\@xdyuserlocationdefs{%
841 \@xdyuserlocationdefs ^^J%
842 (define-location-class \string"#2\string"^^J\space\space
843 \space(:sep "\glsoopenbrace"
844 #1
845 :sep "\glsclosebrace\glsoopenbrace" #3
846 :sep "\glsclosebrace"))
847 }%
848 \fi
849 \edef\@xdyuserlocationnames{%
850 \@xdyuserlocationnames^^J\space\space\space
851 \string"#1\string"}%
852 }

```

Only has an effect before `\writeist`:

```

853 \@onlypremakeg\GlsAddXdyLocation
854 \else
855 \newcommand*{\GlsAddXdyLocation}[2]{%
856 \glsnnoxindywarning\GlsAddXdyLocation}
857 \fi

```

`ylocationclassorder` Define location class order

```

858 \ifglxindy
859 \edef\@xdylocationclassorder{^^J\space\space\space
860 \string"roman-page-numbers\string"^^J\space\space\space
861 \string"arabic-page-numbers\string"^^J\space\space\space
862 \string"arabic-section-numbers\string"^^J\space\space\space
863 \string"alpha-page-numbers\string"^^J\space\space\space
864 \string"Roman-page-numbers\string"^^J\space\space\space
865 \string"Alpha-page-numbers\string"^^J\space\space\space
866 \string"Appendix-page-numbers\string"
867 \@xdyuserlocationnames^^J\space\space\space
868 \string"see\string"
869 }
870 \fi

```

Change the location order.

`yLocationClassOrder`

```

871 \ifglxindy
872 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
873 \def\@xdylocationclassorder{#1}}

```

```

874 \else
875   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
876     \glsnoxywarning\GlsSetXdyLocationClassOrder}
877 \fi

\@xdysortrules   Define sort rules
878 \ifglxindy
879   \def\@xdysortrules{}
880 \fi

\GlsAddSortRule  Add a sort rule
881 \ifglxindy
882   \newcommand*\GlsAddSortRule[2]{%
883     \expandafter\toks@\expandafter{\@xdysortrules}%
884     \protected@edef\@xdysortrules{\the\toks@ ^^J
885       (sort-rule \string"#1\string" \string"#2\string")}%
886   }
887 \else
888   \newcommand*\GlsAddSortRule[2]{%
889     \glsnoxywarning\GlsAddSortRule}
890 \fi

\@xdyrequiredstyles  Define list of required styles (this should be a comma-separated list of xindy
styles)
891 \ifglxindy
892   \def\@xdyrequiredstyles{tex}
893 \fi

\GlsAddXdyStyle  Add a xindy style to the list of required styles
894 \ifglxindy
895   \newcommand*\GlsAddXdyStyle[1]{%
896     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
897 \else
898   \newcommand*\GlsAddXdyStyle[1]{%
899     \glsnoxywarning\GlsAddXdyStyle}
900 \fi

\GlsSetXdyStyles  Reset the list of required styles
901 \ifglxindy
902   \newcommand*\GlsSetXdyStyles[1]{%
903     \edef\@xdyrequiredstyles{#1}}
904 \else
905   \newcommand*\GlsSetXdyStyles[1]{%
906     \glsnoxywarning\GlsSetXdyStyles}
907 \fi

```

`\findrootlanguage` This used to determine the root language, using a bit of trickery since babel doesn't supply the information, but now that babel is once again actively maintained, we can't do this any more, so `\findrootlanguage` no longer available. Now provide a command that does nothing (in case it's been patched).

```
908 \newcommand*\findrootlanguage{}
```

`\@xdylanguage` The xindy language setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```
909 \def\@xdylanguage#1#2{}
```

`\GlsSetXdyLanguage` Define a command that allows the user to set the language for a given glossary type. The first argument indicates the glossary type. If omitted the main glossary is assumed.

```
910 \ifglxindy
911   \newcommand*\GlsSetXdyLanguage[2][\glsdefaultttype]{%
912     \ifglossaryexists{#1}{%
913       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
914     }{%
915       \PackageError{glossaries}{Can't set language type for
916         glossary type '#1' --- no such glossary}{%
917         You have specified a glossary type that doesn't exist}}
918 \else
919   \newcommand*\GlsSetXdyLanguage[2][]{%
920     \glsnoxywarning\GlsSetXdyLanguage}
921 \fi
```

`\@gls@codepage` The xindy codepage setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```
922 \def\@gls@codepage#1#2{}
```

`\GlsSetXdyCodePage` Define command to set the code page.

```
923 \ifglxindy
924   \newcommand*\GlsSetXdyCodePage[1]{%
925     \renewcommand*\@gls@codepage{#1}%
926   }
927   \AtBeginDocument{
928     \ifx\@gls@codepage\@empty
929       \@ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}
930   \fi}
931 \else
932   \newcommand*\GlsSetXdyCodePage[1]{%
```

Suggested by egreg:

```

933 \glsnoxywarning\GlsSetXdyCodePage}
934 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

935 \ifglxindy
936 \ifglx@xindy@glxnumbers
937 \def\@xdylettergroups{(define-letter-group
938 \string"glxnumbers\string"^^J\space\space\space
939 :prefixes (\string"0\string" \string"1\string"
940 \string"2\string" \string"3\string" \string"4\string"
941 \string"5\string" \string"6\string" \string"7\string"
942 \string"8\string" \string"9\string")^^J\space\space\space
943 :before \string"@glxfirstletter\string")}
944 \else
945 \def\@xdylettergroups{}
946 \fi
947 \fi

```

`\GlsAddLetterGroup` Add a new letter group. The first argument is the name of the letter group. The second argument is the xindy code specifying prefixes and ordering.

```

948 \newcommand*\GlsAddLetterGroup[2]{%
949 \expandafter\toks@\expandafter{\@xdylettergroups}%
950 \protected@edef\@xdylettergroups{\the\toks@^^J%
951 (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
952 }%

```

1.5 Loops and conditionals

`\forallglossaries` 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.

```

953 \newcommand*\forallglossaries[3][\@glo@types]{%
954 \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
955 }

```

`\forglsentries` To iterate through all entries in a given glossary use:

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

where *<type>* is the glossary label and *<cmd>* is a control sequence which will be set to the entry label in the current iteration.

```

956 \newcommand*\forglsentries[3][\glsdefaulttype]{%
957 \edef\@glo@list{\csname glolist@#1\endcsname}%
958 \@for#2:=\@glo@list\do{\ifx#2\@empty\else#3\fi}%
959 }

```

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

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

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

```
960 \newcommand*{\forall glossary entries}[3][\@glo@types]{%
961 \expandafter\forall glossaries\expandafter[#1]{\@@this@glo@}{%
962 \forall glossary entries[\@@this@glo@]{#2}{#3}}}
```

`\if glossary exists` To check to see if a glossary exists use:

```
\if glossary exists{<type>}{<true-text>}{<false-text>}
```

where *<type>* is the glossary's label.

```
963 \newcommand{\if glossary exists}[3]{%
964 \ifcsundef{@glo@#1@out}{#3}{#2}%
965 }
```

`\if gloss entry exists` To check to see if a glossary entry has been defined use:

```
\if gloss entry exists{<label>}{<true text>}{<false text>}
```

where *<label>* is the entry's label.

```
966 \newcommand{\if gloss entry exists}[3]{%
967 \ifcsundef{glo@#1@name}{#3}{#2}%
968 }
```

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

```
\if gloss used{<label>}{<true text>}{<false text>}
```

where *<label>* is the entry's label. If true it will do *<true text>* otherwise it will do *<false text>*.

```
969 \newcommand*{\if gloss used}[3]{\ifthenelse{\boolean{glo@#1@flag}}{#2}{#3}}
```

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

`\glsdoif exists` `\glsdoif exists{<label>}{<code>}`

Generate an error if entry specified by *<label>* doesn't exist, otherwise do *<code>*.

```
970 \newcommand{\glsdoif exists}[2]{%
971 \if gloss entry exists{#1}{#2}{%
972 \PackageError{glossaries}{Glossary entry ‘#1’ has not been
973 defined}{You need to define a glossary entry before you
974 can use it.}}%
975 }
```

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

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

```

976 \newcommand{\glsdoifnoexists}[2]{%
977   \ifglstryexists{#1}{%
978     \PackageError{glossaries}{Glossary entry ‘#1’ has already
979       been defined}{#2}%
980 }

```

`\ifglshaschildren` `\ifglshaschildren{<label>}{<true part>}{<false part>}`

```

981 \newcommand{\ifglshaschildren}[3]{%
982   \glsdoifexists{#1}%
983   {%
984     \def\do@glshaschildren{#3}%
985     \expandafter\forlstry\expandafter[\csname glo@#1@type\endcsname]
986     {\glo@label}%
987     {%
988       \letcs\glo@parent{\glo@\glo@label @parent}%
989       \ifthenelse{\equal{#1}{\glo@parent}}{%
990         {%
991           \def\do@glshaschildren{#2}%
992           \@endfortrue
993         }%
994       }%
995     }%
996     \do@glshaschildren
997   }%
998 }

```

`\ifglshasparent` `\ifglshaschildren{<label>}{<true part>}{<false part>}`

```

999 \newcommand{\ifglshasparent}[3]{%
1000   \glsdoifexists{#1}%
1001   {%
1002     \ifcempty{\glo@#1@parent}{#3}{#2}%
1003   }%
1004 }

```

1.6 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`

```

1005 \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`

```
1006 \newcommand*{\newglossary}[5][glg]{%
1007 \ifglossaryexists{#2}{%
1008   \PackageError{glossaries}{Glossary type ‘#2’ already exists}{%
1009     You can’t define a new glossary called ‘#2’ because it already
1010     exists}%
1011 }{%
```

Check if default has been set

```
1012 \ifx\glsdefaulttype\relax
1013   \gdef\glsdefaulttype{#2}%
1014 \fi
```

Add this to the list of glossary types:

```
1015 \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.

```
1016 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store details of this new glossary type:

```
1017 \expandafter\def\csname @glo@type@#2@in\endcsname{#3}%
1018 \expandafter\def\csname @glo@type@#2@out\endcsname{#4}%
1019 \expandafter\def\csname @glo@type@#2@title\endcsname{#5}%
1020 \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`). These may already have been defined if this has been specified as a list of acronyms.

```
1021 \ifcsundef{gls@#2@display}%
1022 {%
1023   \expandafter\gdef\csname gls@#2@display\endcsname{\glsdisplay}%
1024 }%
1025 }%
1026 \ifcsundef{gls@#2@displayfirst}%
1027 {%
1028   \expandafter\gdef\csname gls@#2@displayfirst\endcsname{%
```

```

1029     \glsdisplayfirst
1030   }%
1031 }%
1032 {}%

```

Define sort counter if required:

```

1033 \@gls@defsortcount{#2}%

```

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.)

```

1034 \@ifnextchar[{\@gls@setcounter{#2}}{%
1035   {\@gls@setcounter{#2}[\glscounter]}}%
1036 }

```

`\altnewglossary`

```

1037 \newcommand*{\altnewglossary}[3]{%
1038   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1039 }

```

Only define new glossaries in the preamble:

```

1040 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1041 \@onlypremakeg\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 \TeX , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```

1042 \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`

```

1043 \def\@gls@setcounter#1[#2]{%
1044   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1045   \ifglsxindy
1046     \GlsAddXdyCounters{#2}%
1047   \fi
1048 }

```

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

`\@gls@getcounter`

```

1049 \newcommand*{\@gls@getcounter}[1]{%
1050 \csname @glotype@#1@counter\endcsname}

```

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

```
1051 \glsdefmain
```

1.7 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).

name 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.

```
1052 \define@key{glossentry}{name}{%
1053 \def\@glo@name{#1}%
1054 }
```

description 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.)

```
1055 \define@key{glossentry}{description}{%
1056 \def\@glo@desc{#1}%
1057 }
```

descriptionplural

```
1058 \define@key{glossentry}{descriptionplural}{%
1059 \def\@glo@descplural{#1}%
1060 }
```

sort 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 `\langle name \rangle \langle description \rangle`.

```
1061 \define@key{glossentry}{sort}{%
1062 \def\@glo@sort{#1}}
```

text 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.

```
1063 \define@key{glossentry}{text}{%
1064 \def\@glo@text{#1}%
1065 }
```

plural 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 `\glspluralsuffix` to the value of the text key.

```
1066 \define@key{glossentry}{plural}{%
1067 \def\@glo@plural{#1}%
1068 }
```

first 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.

```
1069 \define@key{glossentry}{first}{%
1070 \def\@glo@first{#1}%
1071 }
```

firstplural 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 `\glspluralsuffix` to the value of the first key.

```
1072 \define@key{glossentry}{firstplural}{%
1073 \def\@glo@firstplural{#1}%
1074 }
```

symbol 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).

```
1075 \define@key{glossentry}{symbol}{%
1076 \def\@glo@symbol{#1}%
1077 }
```

symbolplural

```
1078 \define@key{glossentry}{symbolplural}{%
1079 \def\@glo@symbolplural{#1}%
1080 }
```

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

```
1081 \define@key{glossentry}{type}{%
1082 \def\@glo@type{#1}}
```

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

```
1083 \define@key{glossentry}{counter}{%
1084 \ifcsundef{c@#1}%
1085 {%
1086 \PackageError{glossaries}%
1087 {There is no counter called ‘#1’}%
1088 {%
1089 The counter key should have the name of a valid counter
1090 as its value%
1091 }%
1092 }%
1093 {%
1094 \def\@glo@counter{#1}%
1095 }%
1096 }
```

see The see key specifies a list of cross-references

```
1097 \define@key{glossentry}{see}{%
1098 \def\@glo@see{#1}%
1099 \@glo@seeautonumberlist
1100 }
```

parent The parent key specifies the parent entry, if required.

```
1101 \define@key{glossentry}{parent}{%
1102 \def\@glo@parent{#1}}
```

nonumberlist The nonumberlist key suppresses or activates the number list for the given entry.

```
1103 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1104 \ifcase\nr\relax
1105 \def\@glo@prefix{\glsnonextpages}%
1106 \else
1107 \def\@glo@prefix{\glsnextpages}%
1108 \fi
1109 }
```

Define some generic user keys. (6 ought to be enough!)

user1

```
1110 \define@key{glossentry}{user1}{%
1111 \def\@glo@useri{#1}%
1112 }
```

user2

```
1113 \define@key{glossentry}{user2}{%  
1114   \def\@glo@userii{#1}%  
1115 }
```

user3

```
1116 \define@key{glossentry}{user3}{%  
1117   \def\@glo@useriii{#1}%  
1118 }
```

user4

```
1119 \define@key{glossentry}{user4}{%  
1120   \def\@glo@useriv{#1}%  
1121 }
```

user5

```
1122 \define@key{glossentry}{user5}{%  
1123   \def\@glo@userv{#1}%  
1124 }
```

user6

```
1125 \define@key{glossentry}{user6}{%  
1126   \def\@glo@uservi{#1}%  
1127 }
```

short This key is provided for use by \newacronym. It's not designed for general purpose use, so isn't described in the user manual.

```
1128 \define@key{glossentry}{short}{%  
1129   \def\@glo@short{#1}%  
1130 }
```

shortplural This key is provided for use by \newacronym.

```
1131 \define@key{glossentry}{shortplural}{%  
1132   \def\@glo@shortpl{#1}%  
1133 }
```

long This key is provided for use by \newacronym.

```
1134 \define@key{glossentry}{long}{%  
1135   \def\@glo@long{#1}%  
1136 }
```

longplural This key is provided for use by \newacronym.

```
1137 \define@key{glossentry}{longplural}{%  
1138   \def\@glo@longpl{#1}%  
1139 }
```

```

\@glsnname Define command to generate error if name key is missing.
1140 \newcommand*{\@glsnname}{%
1141   \PackageError{glossaries}{name key required in
1142   \string\newglossaryentry\space for entry '\@glo@label'}{You
1143   haven't specified the entry name}}

```

```

\@glsdefaultplural Define command to set default plural.
1144 \newcommand*{\@glsdefaultplural}{\@glo@text\glspluralsuffix}

```

```

s@missingnumberlist Define a command to generate warning when numberlist not set.
1145 \newcommand*{\@gls@missingnumberlist}[1]{%
1146   ??%
1147   \ifglssavenumberlist
1148     \GlossariesWarning{Missing number list for entry '#1'.
1149     Maybe makeglossaries + rerun required.}%
1150   \else
1151     \PackageError{glossaries}%
1152     {Package option 'savenumberlist=true' required.}%
1153     {%
1154     You must use the 'savenumberlist' package option
1155     to reference location lists.%
1156     }%
1157   \fi
1158 }

```

```

\@glsdefaultsort Define command to set default sort.
1159 \newcommand*{\@glsdefaultsort}{\@glo@name}

```

```

\gls@level Register to increment entry levels.
1160 \newcount\gls@level

```

```

\newglossaryentry Define \newglossaryentry {<label>} {<key-val list>}. There are two required
fields in <key-val list>: name (or parent) and description. (See above.)

```

```

1161 \newrobustcmd{\newglossaryentry}[2]{%
  Check to see if this glossary entry has already been defined:
1162   \glsdoifnoexists{#1}%
1163   {%
    Store label
1164     \def\@glo@label{#1}%
    Set up defaults. If the name or description keys are omitted, an error will be
    generated.
1165     \let\@glo@name\@glsnname
1166     \def\@glo@desc{%
1167       \PackageError{glossaries}
1168       {%

```

```

1169      description key required in \string\newglossaryentry\space
1170      for entry ‘\@glo@label’%
1171  }%
1172  {%
1173      You haven’t specified the entry description%
1174  }%
1175  }%

```

```

1176  \def\@glo@descplural{\@glo@desc}%
1177  \def\@glo@type{\glsdefaulttype}%
1178  \def\@glo@symbol{\relax}%

1179  \def\@glo@symbolplural{\@glo@symbol}%
1180  \def\@glo@text{\@glo@name}%
1181  \let\@glo@plural\@glsdefaultplural

```

Using \let instead of \def to make later comparison avoid expansion issues.
(Thanks to Ulrich Diez for suggesting this.)

```

1182  \let\@glo@first\relax

1183  \let\@glo@firstplural\relax

```

Set the default sort:

```

1184  \let\@glo@sort\@glsdefaultsort

```

Set the default counter:

```

1185  \def\@glo@counter{\@gls@getcounter{\@glo@type}}%

1186  \def\@glo@see{}%

1187  \def\@glo@parent{}%

1188  \def\@glo@prefix{}%

1189  \def\@glo@useri{}%
1190  \def\@glo@userii{}%
1191  \def\@glo@useriii{}%
1192  \def\@glo@useriv{}%
1193  \def\@glo@userv{}%
1194  \def\@glo@uservi{}%

1195  \def\@glo@short{}%
1196  \def\@glo@shortpl{}%
1197  \def\@glo@long{}%
1198  \def\@glo@longpl{}%

```

Add start hook in case another package wants to add extra keys.

```

1199  \@newglossaryentryprehook

```


Extract key-val information from third parameter:

```

1200    \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.

1201    \ifcsundef{glolist@\@glo@type}%
1202    {%
1203        \PackageError{glossaries}%
1204        {Glossary type '\@glo@type' has not been defined}%
1205        {You need to define a new glossary type, before making entries
1206         in it}%
1207    }%
1208    {%
1209        \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
1210        \expandafter\xdef\csname glolist@\@glo@type\endcsname{\@glolist@{#1},}%
1211    }%

Initialise level to 0.

1212    \gls@level=0\relax

Has this entry been assigned a parent?

1213    \ifx\@glo@parent\@empty

Doesn't have a parent. Set \glo@<label>@parent to empty.

1214        \expandafter\gdef\csname glo@#1@parent\endcsname{}%
1215    \else

Has a parent. Check to ensure this entry isn't its own parent.

1216        \ifthenelse{\equal{#1}{\@glo@parent}}{%
1217            {%
1218                \PackageError{glossaries}{Entry '#1' can't be its own parent}{}%
1219                \def\@glo@parent{}%
1220                \expandafter\gdef\csname glo@#1@parent\endcsname{}%
1221            }%
1222        }%

Check the parent exists:

1223        \ifglentryexists{\@glo@parent}%
1224        {%

Parent exists. Set \glo@<label>@parent.

1225            \expandafter\xdef\csname glo@#1@parent\endcsname{\@glo@parent}%

Determine level.

1226            \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
1227            \advance\gls@level by 1\relax

If name hasn't been specified, use same as the parent name

1228            \ifx\@glo@name\@glsnoname
1229                \expandafter\let\expandafter\@glo@name
1230                \csname glo@\@glo@parent @name\endcsname

```

If name and plural haven't been specified, use same as the parent

```

1231         \ifx\@glo@plural\@glsdefaultplural
1232         \expandafter\let\expandafter\@glo@plural
1233             \csname glo@\@glo@parent @plural\endcsname
1234         \fi
1235     \fi
1236 }%
1237 {%

```

Parent doesn't exist, so issue an error message and change this entry to have no parent

```

1238     \PackageError{glossaries}%
1239     {%
1240         Invalid parent '@glo@parent'
1241         for entry '#1' - parent doesn't exist%
1242     }%
1243     {%
1244         Parent entries must be defined before their children%
1245     }%
1246     \def\@glo@parent{%
1247     \expandafter\gdef\csname glo@#1@parent\endcsname{%
1248     }%
1249     }%
1250 \fi

```

Set the level for this entry

```

1251 \expandafter\xdef\csname glo@#1@level\endcsname{\number\gls@level}%

```

Check if first and firstplural have been use. If firstplural hasn't been specified, but first has been specified, then form firstplural by appending \glspluralsuffix to value of first key, otherwise obtain the value from the plural key. This now uses \ifx instead of \if to avoid expansion issues. (Thanks to Ulrich Diez for suggesting this.)

```

1252 \ifx\relax\@glo@firstplural
1253     \ifx\relax\@glo@first
1254         \def\@glo@firstplural{\@glo@plural}%
1255         \def\@glo@first{\@glo@text}%
1256     \else
1257         \def\@glo@firstplural{\@glo@first\glspluralsuffix}%
1258     \fi
1259 \else
1260     \ifx\relax\@glo@first
1261         \def\@glo@first{\@glo@text}%
1262     \fi
1263 \fi

```

Define commands associated with this entry:

```

1264 \expandafter
1265 \protected@xdef\csname glo@#1@text\endcsname{\@glo@text}%
1266 \expandafter

```

```

1267 \protected@xdef\csname glo@#1@plural\endcsname{\@glo@plural}%
1268 \expandafter
1269 \protected@xdef\csname glo@#1@first\endcsname{\@glo@first}%
1270 \expandafter
1271 \protected@xdef\csname glo@#1@firstpl\endcsname{\@glo@firstplural}%
1272 \expandafter
1273 \protected@xdef\csname glo@#1@type\endcsname{\@glo@type}%
1274 \expandafter
1275 \protected@xdef\csname glo@#1@counter\endcsname{\@glo@counter}%
1276 \expandafter
1277 \protected@xdef\csname glo@#1@useri\endcsname{\@glo@useri}%
1278 \expandafter
1279 \protected@xdef\csname glo@#1@userii\endcsname{\@glo@userii}%
1280 \expandafter
1281 \protected@xdef\csname glo@#1@useriii\endcsname{\@glo@useriii}%
1282 \expandafter
1283 \protected@xdef\csname glo@#1@useriv\endcsname{\@glo@useriv}%
1284 \expandafter
1285 \protected@xdef\csname glo@#1@userv\endcsname{\@glo@userv}%
1286 \expandafter
1287 \protected@xdef\csname glo@#1@uservi\endcsname{\@glo@uservi}%
1288 \expandafter
1289 \protected@xdef\csname glo@#1@short\endcsname{\@glo@short}%
1290 \expandafter
1291 \protected@xdef\csname glo@#1@shortpl\endcsname{\@glo@shortpl}%
1292 \expandafter
1293 \protected@xdef\csname glo@#1@long\endcsname{\@glo@long}%
1294 \expandafter
1295 \protected@xdef\csname glo@#1@longpl\endcsname{\@glo@longpl}%
1296 \@gls@sanitizename
1297 \expandafter\protected@xdef\csname glo@#1@name\endcsname{\@glo@name}%

```

Set default numberlist if not defined:

```

1298 \ifcsundef{glo@#1@numberlist}%
1299 {%
1300 \csxdef{glo@#1@numberlist}{\noexpand\@gls@missingnumberlist{\@glo@label}}%
1301 }%
1302 {}%

```

The smaller and smallcaps options set the description to \@glo@first. Need to check for this, otherwise it won't get expanded if the description gets sanitized.

```

1303 \def\@glo@@desc{\@glo@first}%
1304 \ifx\@glo@desc\@glo@@desc
1305 \let\@glo@desc\@glo@first
1306 \fi
1307 \@gls@sanitizedesc
1308 \expandafter\protected@xdef\csname glo@#1@desc\endcsname{\@glo@desc}%
1309 \expandafter\protected@xdef\csname glo@#1@descplural\endcsname{\@glo@descplural}%

```

Set the sort key for this entry:

```

1310 \gls@defsort{\@glo@type}{#1}%
1311 \def\@glo@@symbol{\@glo@text}%
1312 \ifx\@glo@symbol\@glo@@symbol
1313 \let\@glo@symbol\@glo@text
1314 \fi
1315 \gls@sanitizesymbol
1316 \expandafter\protected@xdef\csname glo@#1@symbol\endcsname{\@glo@symbol}%
1317 \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):

```

1318 \expandafter\gdef\csname glo@#1@flagfalse\endcsname{%
1319 \expandafter\global\expandafter
1320 \let\csname ifglo@#1@flag\endcsname\iffalse
1321 }%
1322 \expandafter\gdef\csname glo@#1@flagtrue\endcsname{%
1323 \expandafter\global\expandafter
1324 \let\csname ifglo@#1@flag\endcsname\iftrue
1325 }%
1326 \csname glo@#1@flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

1327 \ifx\@glo@see\@empty
1328 \else
1329 \protected@edef\@do@glsee{%
1330 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
1331 \noexpand\@nil
1332 \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{#1}}%
1333 \do@glsee
1334 \fi
1335 }%

```

Determine and store main part of the entry's index format.

```

1336 \do@glo@storeentry{#1}%

```

Add end hook in case another package wants to add extra keys.

```

1337 \@newglossaryentryposthook
1338 }

```

`\glossaryentryprehook` Allow extra information to be added to glossary entries:

```

1339 \newcommand*{\@newglossaryentryprehook}{}

```

`\glossaryentryposthook` Allow extra information to be added to glossary entries:

```

1340 \newcommand*{\@newglossaryentryposthook}{}

```

`\glsmoveentry` Moves entry whose label is given by first argument to the glossary named in the second argument.

```

1341 \newcommand*{\glsmoveentry}[2]{%

```

```

1342 \edef\glo@type{\csname glo@#1@type\endcsname}%
1343 \def\glo@list{,}%
1344 \forlslentries[\glo@type]{\glo@label}%
1345 {%
1346     \ifthenelse{\equal{\glo@label}{#1}}{\}\{\eappto\glo@list{\glo@label,}}%
1347 }%
1348 \cslet{glolist@\glo@type}{\glo@list}%
1349 \csdef{glo@#1@type}{#2}%
1350 }

```

@glossaryentryfield Indicate what command should be used to display each entry in the glossary. (This enables the glossaries-accsupp package to use `\accsuppglossaryentryfield` instead.)

```

1351 \ifglxindy
1352 \newcommand*{\@glossaryentryfield}{\string\@glossaryentryfield}
1353 \else
1354 \newcommand*{\@glossaryentryfield}{\string\glossaryentryfield}
1355 \fi

```

glossarysubentryfield Indicate what command should be used to display each subentry in the glossary. (This enables the glossaries-accsupp package to use `\accsuppglossarysubentryfield` instead.)

```

1356 \ifglxindy
1357 \newcommand*{\@glossarysubentryfield}{%
1358     \string\@glossarysubentryfield}
1359 \else
1360 \newcommand*{\@glossarysubentryfield}{%
1361     \string\glossarysubentryfield}
1362 \fi

```

\@glo@storeentry Determine the format to write the entry in the glossary output (`.glo`) file. The argument is the entry's label. The result is stored in `\glo@<label>@entry`, where `<label>` is the entry's label. (This doesn't include any formatting or location information.)

```

1363 \newcommand{\@glo@storeentry}[1]{%
    Get the sort string and escape any special characters
1364 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
1365 \@gls@checkmkidxchars\@glo@sort
    Same again for the name string.
1366 \protected@edef\@glo@name{\csname glo@#1@name\endcsname}%
1367 \@gls@checkmkidxchars\@glo@name
    Add the font command. (The backslash needs to be escaped for xindy.)
1368 \ifglxindy
1369 \protected@edef\@glo@name{\string\@glsnamefont{\@glo@name}}%
1370 \else
1371 \protected@edef\@glo@name{\string\glsnamefont{\@glo@name}}%
1372 \fi

```

Get the description string and escape any special characters

```
1373 \protected@edef\@glo@desc{\csname glo@#1@desc\endcsname}%  
1374 \@gls@checkmkidxchars\@glo@desc
```

Same again for the symbol

```
1375 \protected@edef\@glo@symbol{\csname glo@#1@symbol\endcsname}%  
1376 \@gls@checkmkidxchars\@glo@symbol
```

Escape any special characters in the prefix

```
1377 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
1378 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
1379 \ifglxindy
```

Store using xindy syntax.

```
1380 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
1381 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
1382 (\string"\@glo@sort\string" %  
1383 \string"\@glo@prefix\@glossaryentryfield{#1}\@glo@name  
1384 }\@glo@desc}\@glo@symbol}\string") %  
1385 }%  
1386 \else
```

Entry has a parent

```
1387 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
1388 \csname glo@\@glo@parent @index\endcsname  
1389 (\string"\@glo@sort\string" %  
1390 \string"\@glo@prefix\@glossarysubentryfield%  
1391 {\csname glo@#1@level\endcsname}{#1}\@glo@name  
1392 }\@glo@desc}\@glo@symbol}\string") %  
1393 }%  
1394 \fi  
1395 \else
```

Store using makeindex syntax.

```
1396 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
1397 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
1398 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
1399 \@glo@sort\@gls@actualchar\@glo@prefix  
1400 \@glossaryentryfield{#1}\@glo@name}\@glo@desc  
1401 }\@glo@symbol}%  
1402 }%  
1403 \else
```

Entry has a parent

```
1404 \expandafter\protected\edef\csname glo@#1@index\endcsname{%
1405 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
1406 \@glo@sort\@gls@actualchar\@glo@prefix
1407 \@glossarysubentryfield
1408 {\csname glo@#1@level\endcsname}{#1}{\@glo@name}{\@glo@desc
1409 }\@glo@symbol}%
1410 }%
1411 \fi
1412 \fi
1413 }
```

1.8 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`

```
1414 \newcommand*{\glsreset}[1]{%
1415 \glsdoifexists{#1}{%
1416 \expandafter\global\csname glo@#1@flagfalse\endcsname}}
```

As above, but with only a local effect:

`\glslocalreset`

```
1417 \newcommand*{\glslocalreset}[1]{%
1418 \glsdoifexists{#1}{%
1419 \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`

```
1420 \newcommand*{\glsunset}[1]{%
1421 \glsdoifexists{#1}{%
1422 \expandafter\global\csname glo@#1@flagtrue\endcsname}}
```

As above, but with only a local effect:

`\glslocalunset`

```
1423 \newcommand*{\glslocalunset}[1]{%
1424 \glsdoifexists{#1}{%
1425 \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`

```
1426 \newcommand*{\glsresetall}[1][\@glo@types]{%
1427 \forallglsentries[#1]{\@glsentry}{%
1428 \glsreset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalresetall`

```
1429 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
1430 \forallglsentries[#1]{\@glsentry}{%
1431 \glslocalreset{\@glsentry}}}
```

Unset all entries for the named glossaries (supplied in a comma-separated list).

Syntax: `\glsunsetall[<glossary-list>]`

`\glsunsetall`

```
1432 \newcommand*{\glsunsetall}[1][\@glo@types]{%
1433 \forallglsentries[#1]{\@glsentry}{%
1434 \glsunset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalunsetall`

```
1435 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
1436 \forallglsentries[#1]{\@glsentry}{%
1437 \glslocalunset{\@glsentry}}}
```

1.9 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.¹

`\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`

```
1438 \newcommand*{\loadglsentries}[2][\@gls@default]{%
1439 \let\@gls@default\glsdefaulttype
1440 \def\glsdefaulttype{#1}\input{#2}%
1441 \let\glsdefaulttype\@gls@default}
```

`\loadglsentries` can only be used in the preamble:

```
1442 \@onlypreamble{\loadglsentries}
```

¹and any other valid \LaTeX code that can be used in the preamble.

1.10 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`

```
1443 \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.

`\glsdisplayfirst`

```
1444 \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`

```
1445 \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`

```
1446 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
1447 \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`

```
1448 \newcommand*{\defglsdisplayfirst}[2][\glsdefaultttype]{%
1449 \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}}
```

1.10.1 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 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.

```
1450 \define@key{glslink}{counter}{%
1451   \ifcsundef{c@#1}%
1452   {%
1453     \PackageError{glossaries}%
1454     {There is no counter called ‘#1’}%
1455     {%
1456       The counter key should have the name of a valid counter
1457       as its value%
1458     }%
1459   }%
1460   {%
1461     \def\@gls@counter{#1}%
1462   }%
1463 }
```

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.

```
1464 \define@key{glslink}{format}{%
1465 \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.

```
1466 \define@boolkey{glslink}{hyper}[true]{}
```

The local key is a boolean key. If true this indicates that commands such as `\gls` should only do a local reset rather than a global one.

```
1467 \define@boolkey{glslink}{local}[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
```

```
1468 \newrobustcmd*{\glslink}{%
```

```
1469 \@ifstar\@sgls@link\@gls@@link}
```

`\@sgls@link` The starred version of `\glslink` calls the unstarred version with hyperlinks disabled.

```
1470 \newcommand*{\@sgls@link}[1][\@gls@@link[hyper=false,#1]]
```

`\@gls@@link` The unstarred version of `\glslink` checks for the existence of the term. The main part of the business is in `\@gls@link` which shouldn't check if the term is defined as it's called by `\gls` etc which also perform that check.

```
1471 \newcommand*{\@gls@@link}[3][\@gls@link]
```

```
1472 \ifglsentryexists{#2}%
```

```
1473 {%
```

```
1474 \@gls@link[#1]{#2}{#3}%
```

```
1475 }%
```

```
1476 \PackageError{glossaries}{Glossary entry ‘#2’ has not been
```

```
1477 defined}{You need to define a glossary entry before you
```

```
1478 can use it.}%
```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
1479 \glstextformat{#3}%
```

```
1480 }%
```

```
1481 }
```

```
\@gls@link
```

```
1482 \def\@gls@link[#1]#2#3{%
```

Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with `tabularx`).

```
1483 \leavevmode
1484 \def\glslabel{#2}%
1485 \def\@glsnumberformat{glsnumberformat}%
1486 \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
```

If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default

```
1487 \edef\gls@type{\csname glo@#2@type\endcsname}%
1488 \expandafter\DTLifinlist\expandafter
1489 {\gls@type}{\@gls@nohyperlist}%
1490 {%
1491 \KV@glslink@hyperfalse
1492 }%
1493 {%
1494 \KV@glslink@hypertrue
1495 }%
1496 \setkeys{glslink}{#1}%
```

Store the entry’s counter in `\theglsentrycounter`

```
1497 \@gls@saveentrycounter
```

Define sort key if necessary:

```
1498 \@gls@setsort{#2}%
1499 \@do@wrglossary{#2}%
1500 \ifKV@glslink@hyper
1501 \glslink{\glolinkprefix#2}{\glstextformat{#3}}%
1502 \else
1503 \glstextformat{#3}\relax
1504 \fi
1505 }
```

`\glolinkprefix`

```
1506 \newcommand*{\glolinkprefix}{glo:}
```

`\glsentrycounter` Set default value of entry counter

```
1507 \def\glsentrycounter{glscounter}%
```

`\@saveentrycounter` Need to check if using equation counter in align environment:

```
1508 \newcommand*{\@gls@saveentrycounter}{%
1509 \def\@gls@Hcounter{}}%
```

Are we using equation counter?

```
1510 \ifthenelse{\equal{\@gls@counter}{equation}}{%
1511 {
```

If we in align environment, `\xatlevel@` will be defined. (Can’t test for `\@currentenv` as may be inside an inner environment.)

```

1512 \ifcsundef{xatlevel@}%
1513 {%
1514 \edef\theglentrycounter{\expandafter\noexpand
1515 \csname the\@gls@counter\endcsname}%
1516 }%
1517 {%
1518 \ifx\xatlevel@\@empty
1519 \edef\theglentrycounter{\expandafter\noexpand
1520 \csname the\@gls@counter\endcsname}%
1521 \else
1522 \savecounters@
1523 \advance\c@equation by 1\relax
1524 \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

1525 \ifcsundef{theH\@gls@counter}%
1526 {%
1527 \def\@gls@Hcounter{\theglentrycounter}%
1528 }%
1529 {%
1530 \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
1531 }%
1532 \protected@edef\theHglentrycounter{\@gls@Hcounter}%
1533 \restorecounters@
1534 \fi
1535 }%
1536 }%
1537 {%

```

Not using equation counter so no special measures:

```

1538 \edef\theglentrycounter{\expandafter\noexpand
1539 \csname the\@gls@counter\endcsname}%
1540 }%

```

Check if hyperref version of this counter

```

1541 \ifx\@gls@Hcounter\@empty
1542 \ifcsundef{theH\@gls@counter}%
1543 {%
1544 \def\theHglentrycounter{\theglentrycounter}%
1545 }%
1546 {%
1547 \protected@edef\theHglentrycounter{\expandafter\noexpand
1548 \csname theH\@gls@counter\endcsname}%
1549 }%
1550 \fi
1551 }

```

`\@set@glo@numformat` Set the formatting information in the format required by `makeindex`. The first argument is the format specified by the user (via the `format` key), the second argument is the name of the counter used to indicate the location, the third

argument is a control sequence which stores the required format and the fourth argument (new to v3.0) is the hyper-prefix.

```

1552 \def\@set@glo@numformat#1#2#3#4{%
1553   \expandafter\@glo@check@mkidxrangechar#3\@nil
1554   \protected@edef#1{%
1555     \@glo@prefix setentrycounter[#4]{#2}%
1556     \expandafter\string\curname\@glo@suffix\endcurname
1557   }%
1558   \@gls@checkmkidxchars#1%
1559 }

```

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

```

1560 \def\@glo@check@mkidxrangechar#1#2\@nil{%
1561   \if#1(\relax
1562     \def\@glo@prefix{(%
1563     \if\relax#2\relax
1564       \def\@glo@suffix{glsnumberformat}%
1565     \else
1566       \def\@glo@suffix{#2}%
1567     \fi
1568   \else
1569     \if#1)\relax
1570       \def\@glo@prefix{)%
1571     \if\relax#2\relax
1572       \def\@glo@suffix{glsnumberformat}%
1573     \else
1574       \def\@glo@suffix{#2}%
1575     \fi
1576   \else
1577     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
1578   \fi
1579 }

```

`\@gls@escbsdq` Escape backslashes and double quote marks. The argument must be a control sequence.

```

1580 \newcommand*\@gls@escbsdq[1]{%
1581   \def\@gls@checkedmkidx{%
1582     \let\gls@xdystring=#1\relax
1583     \@onelevel@sanitize\gls@xdystring
1584     \edef\do@gls@xdycheckbackslash{%
1585       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
1586       \@backslashchar\@backslashchar\noexpand\null}%
1587     \do@gls@xdycheckbackslash
1588     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
1589   }

```

```

1590 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
1591 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

  Unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage
  (thanks to David Carlisle for the suggestion.)

1592 \@for\@gls@tmp:=\gls@protected@pagefmts\do
1593 {%
1594   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
1595   \@onelevel@sanitize\@gls@sanitized@tmp
1596   \edef\gls@dosubst{%
1597     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
1598     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
1599   }%
1600   \gls@dosubst
1601 }%

  Assign to required control sequence
1602 \let#1=\gls@xdystring
1603 }

```

Catch special characters(argument must be a control sequence):

`\gls@checkmkidxchars`

```

1604 \newcommand{\@gls@checkmkidxchars}[1]{%
1605 \ifglxsindy
1606   \@gls@escbsdq{#1}%
1607 \else
1608   \def\@gls@checkedmkidx{}%
1609   \expandafter\@gls@checkquote#1\@nil""\null
1610   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1611   \def\@gls@checkedmkidx{}%
1612   \expandafter\@gls@checkescquote#1\@nil\""\null
1613   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1614   \def\@gls@checkedmkidx{}%
1615   \expandafter\@gls@checkescactual#1\@nil\?\?\null
1616   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1617   \def\@gls@checkedmkidx{}%
1618   \expandafter\@gls@checkactual#1\@nil??\null
1619   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1620   \def\@gls@checkedmkidx{}%
1621   \expandafter\@gls@checkbar#1\@nil||\null
1622   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1623   \def\@gls@checkedmkidx{}%
1624   \expandafter\@gls@checkesbar#1\@nil\\|\null
1625   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1626   \def\@gls@checkedmkidx{}%
1627   \expandafter\@gls@checklevel#1\@nil!!\null
1628   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1629 \fi
1630 }

```

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

```
1631 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
1632 \newtoks\@gls@tmpb
```

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

```
1633 \def\@gls@checkquote#1"#2"#3\null{%
1634 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1635 \toks@={#1}%
1636 \ifx\null#2\null
1637 \ifx\null#3\null
1638 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1639 \def\@gls@checkquote{\relax}%
1640 \else
1641 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1642 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
1643 \def\@gls@checkquote{\@gls@checkquote#3\null}%
1644 \fi
1645 \else
1646 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1647 \@gls@quotechar\@gls@quotechar}%
1648 \ifx\null#3\null
1649 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
1650 \else
1651 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
1652 \fi
1653 \fi
1654 \@gls@checkquote}
```

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

```
1655 \def\@gls@checkescquote#1\"#2\"#3\null{%
1656 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1657 \toks@={#1}%
1658 \ifx\null#2\null
1659 \ifx\null#3\null
1660 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1661 \def\@gls@checkescquote{\relax}%
1662 \else
1663 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1664 \@gls@quotechar\string\" \@gls@quotechar
1665 \@gls@quotechar\string\" \@gls@quotechar}%
1666 \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
1667 \fi
1668 \else
1669 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1670 \@gls@quotechar\string\" \@gls@quotechar}%

```



```

1671 \ifx\null#3\null
1672   \def\@gls@checkescquote{\@gls@checkescquote#2\""\null}%
1673 \else
1674   \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
1675 \fi
1676 \fi
1677 \@gls@checkescquote}

```

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

```

1678 \def\@gls@checkescactual#1\?#2\?#3\null{%
1679 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1680 \toks@={#1}%
1681 \ifx\null#2\null
1682 \ifx\null#3\null
1683   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1684   \def\@gls@checkescactual{\relax}%
1685 \else
1686   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1687     \@gls@quotechar\string\""\@gls@actualchar
1688     \@gls@quotechar\string\""\@gls@actualchar}%
1689   \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
1690 \fi
1691 \else
1692   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1693     \@gls@quotechar\string\""\@gls@actualchar}%
1694 \ifx\null#3\null
1695   \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
1696 \else
1697   \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
1698 \fi
1699 \fi
1700 \@gls@checkescactual}

```

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

```

1701 \def\@gls@checkescbar#1\|#2\|#3\null{%
1702 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1703 \toks@={#1}%
1704 \ifx\null#2\null
1705 \ifx\null#3\null
1706   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1707   \def\@gls@checkescbar{\relax}%
1708 \else
1709   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1710     \@gls@quotechar\string\""\@gls@encapchar
1711     \@gls@quotechar\string\""\@gls@encapchar}%
1712   \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
1713 \fi
1714 \else
1715   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

```

```

1716 \@gls@quotechar\string\" \@gls@encapchar}%
1717 \ifx\@null#3\@null
1718 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\@null}%
1719 \else
1720 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\@null}%
1721 \fi
1722 \fi
1723 \@gls@checkescbar}

```

\@gls@checkesclevel Similarly for \!:

```

1724 \def\@gls@checkesclevel#1\!#2\!#3\@null{%
1725 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1726 \toks@={#1}%
1727 \ifx\@null#2\@null
1728 \ifx\@null#3\@null
1729 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1730 \def\@gls@checkesclevel{\relax}%
1731 \else
1732 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1733 \@gls@quotechar\string\" \@gls@levelchar
1734 \@gls@quotechar\string\" \@gls@levelchar}%
1735 \def\@gls@checkesclevel{\@gls@checkesclevel#3\@null}%
1736 \fi
1737 \else
1738 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1739 \@gls@quotechar\string\" \@gls@levelchar}%
1740 \ifx\@null#3\@null
1741 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\@null}%
1742 \else
1743 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\@null}%
1744 \fi
1745 \fi
1746 \@gls@checkesclevel}

```

\@gls@checkbar and for |:

```

1747 \def\@gls@checkbar#1|#2|#3\@null{%
1748 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1749 \toks@={#1}%
1750 \ifx\@null#2\@null
1751 \ifx\@null#3\@null
1752 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1753 \def\@gls@checkbar{\relax}%
1754 \else
1755 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1756 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
1757 \def\@gls@checkbar{\@gls@checkbar#3\@null}%
1758 \fi
1759 \else
1760 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

```

```

1761 \@gls@quotechar\@gls@encapchar}%
1762 \ifx\null#3\null
1763 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
1764 \else
1765 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
1766 \fi
1767 \fi
1768 \@gls@checkbar}

```

\@gls@checklevel and for !:

```

1769 \def\@gls@checklevel#1!#2!#3\null{%
1770 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1771 \toks@={#1}%
1772 \ifx\null#2\null
1773 \ifx\null#3\null
1774 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1775 \def\@gls@checklevel{\relax}%
1776 \else
1777 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1778 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
1779 \def\@gls@checklevel{\@gls@checklevel#3\null}%
1780 \fi
1781 \else
1782 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1783 \@gls@quotechar\@gls@levelchar}%
1784 \ifx\null#3\null
1785 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
1786 \else
1787 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
1788 \fi
1789 \fi
1790 \@gls@checklevel}

```

\@gls@checkactual and for ?:

```

1791 \def\@gls@checkactual#1?#2?#3\null{%
1792 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1793 \toks@={#1}%
1794 \ifx\null#2\null
1795 \ifx\null#3\null
1796 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1797 \def\@gls@checkactual{\relax}%
1798 \else
1799 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1800 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
1801 \def\@gls@checkactual{\@gls@checkactual#3\null}%
1802 \fi
1803 \else
1804 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1805 \@gls@quotechar\@gls@actualchar}%

```

```

1806 \ifx\null#3\null
1807   \def\@gls@checkactual{\@gls@checkactual#2??\null}%
1808 \else
1809   \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
1810 \fi
1811 \fi
1812 \@gls@checkactual}

```

\@gls@xdycheckquote As before but for use with xindy

```

1813 \def\@gls@xdycheckquote#1"#2"#3\null{%
1814 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1815 \toks@={#1}%
1816 \ifx\null#2\null
1817   \ifx\null#3\null
1818     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1819     \def\@gls@xdycheckquote{\relax}%
1820 \else
1821     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1822       \string"\string"}%
1823     \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
1824 \fi
1825 \else
1826     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1827       \string"}%
1828   \ifx\null#3\null
1829     \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
1830 \else
1831     \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
1832 \fi
1833 \fi
1834 \@gls@xdycheckquote
1835 }

```

s@xdycheckbackslash Need to escape all backslashes for xindy. Define command that will define

```

\@gls@xdycheckbackslash
1836 \edef\def@gls@xdycheckbackslash{%
1837 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
1838   ##2\@backslashchar##3\noexpand\null{%
1839 \noexpand\@gls@tmpb=\noexpand\expandafter
1840   {\noexpand\@gls@checkedmkidx}%
1841 \noexpand\toks@={##1}%
1842 \noexpand\ifx\noexpand\null##2\noexpand\null
1843 \noexpand\ifx\noexpand\null##3\noexpand\null
1844 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1845   \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
1846 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
1847 \noexpand\else
1848 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1849   \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@

```

```

1850 \backslashchar\backslashchar\backslashchar\backslashchar}%
1851 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1852 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
1853 \noexpand\fi
1854 \noexpand\else
1855 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1856 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
1857 \backslashchar\backslashchar}%
1858 \noexpand\ifx\noexpand\null##3\noexpand\null
1859 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1860 \noexpand\@gls@xdycheckbackslash##2\backslashchar
1861 \backslashchar\noexpand\null}%
1862 \noexpand\else
1863 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1864 \noexpand\@gls@xdycheckbackslash##2\backslashchar
1865 ##3\noexpand\null}%
1866 \noexpand\fi
1867 \noexpand\fi
1868 \noexpand\@gls@xdycheckbackslash
1869 }%
1870 }

```

Now go ahead and define \gls@xdycheckbackslash

```

1871 \def@gls@xdycheckbackslash

```

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

```

1872 \ifcsundef{hyperlink}%
1873 {%
1874 \gdef@glslink#1#2{#2}%
1875 }%
1876 {%
1877 \gdef@glslink#1#2{\hyperlink{#1}{#2}}%
1878 }

```

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

```

1879 \newlength@gls@tmplen
1880 \ifcsundef{hypertarget}%
1881 {%
1882 \gdef@glstarget#1#2{#2}%
1883 }%
1884 {%
1885 \gdef@glstarget#1#2{%
1886 \settoheight@gls@tmplen{#2}%
1887 \raisebox@gls@tmplen{\hypertarget{#1}}{#2}%
1888 }%
1889 }

```

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

`\glsdisablehyper`

```
1890 \newcommand{\glsdisablehyper}{%
1891 \renewcommand*\@glslink[2]{##2}%
1892 \renewcommand*\@glsstarget[2]{##2}}
```

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

`\glsenablehyper`

```
1893 \newcommand{\glsenablehyper}{%
1894 \renewcommand*\@glslink[2]{\hyperlink{##1}{##2}}%
1895 \renewcommand*\@glsstarget[2]{%
1896   \settoheight{\gls@tmplen}{##2}%
1897   \raisebox{\gls@tmplen}{\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`

```
1898 \newrobustcmd*\@gls{\@ifstar\@sgls\@gls}
```

Define the starred form:

`\@sgls`

```
1899 \newcommand*\@sgls[1][]{\@gls[hyper=false,#1]}
```

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

`\@gls`

```
1900 \newcommand*\@gls[2][]{%
1901   \new@ifnextchar{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}%
1902 }
```

\@gls@ Read in the final optional argument:

```
1903 \def\@gls@#1#2[#3]{%
1904   \glsdoifexists{#2}%
1905   {%
1906     \edef\@glo@type{\glsentrytype{#2}}%
1907     Save options in \@gls@link@opts and label in \@gls@link@label
1908     \def\@gls@link@opts{#1}%
1909     \def\@gls@link@label{#2}%
1910     Determine what the link text should be (this is stored in \@glo@text)
1911     \ifglsused{#2}%
1912     {%
1913       \def\@glo@text{%
1914         \csname gls@\@glo@type @display\endcsname
1915         {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
1916       }%
1917     {%
1918       \def\@glo@text{%
1919         \csname gls@\@glo@type @displayfirst\endcsname
1920         {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
1921       }%
1922     }
```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```
1920   \ifglsused{#2}%
1921   {%
1922     \@gls@link[#1]{#2}{\@glo@text}%
1923   }%
1924   {%
1925     \gls@checkisacronymlist\@glo@type
1926     \ifthenelse
1927     {(\boolean{glsisacronymlist}\AND \boolean{glsacrfootnote})\OR \NOT\boolean{glshyperfirst}}
1928     {}%
1929     {\@gls@link[#1,hyper=false]{#2}{\@glo@text}%
1930     }%
1931     {\@gls@link[#1]{#2}{\@glo@text}%
1932     }%
1933   }%
1934   \@gls@link[#1]{#2}{\@glo@text}%
1935   }%
1936 }
```

Indicate that this entry has now been used

```
1937   \ifKV@glslink@local
1938   \glslocalunset{#2}%
1939   \else
1940   \glsunset{#2}%
1941   \fi
1942 }
```

1943 }

`\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`

1944 `\newrobustcmd*{\Gls}{\@ifstar\@sGls\@Gls}`

Define the starred form:

1945 `\newcommand*{\@sGls}[1][\@Gls[hyper=false,#1]]`

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

1946 `\newcommand*{\@Gls}[2][\@Gls@{#1}{#2}]{\@Gls@{#1}{#2}[]}`

1947 `\new@ifnextchar[\@Gls@{#1}{#2}]{\@Gls@{#1}{#2}[]}`

1948 }

`\@Gls@` Read in the final optional argument:

1949 `\def\@Gls@#1#2[#3]{%`

1950 `\glsdoifexists{#2}%`

1951 `{%`

1952 `\edef\@glo@type{\glsentrytype{#2}}%`

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

1953 `\def\@gls@link@opts{#1}%`

1954 `\def\@gls@link@label{#2}%`

1955 `\def\@gls@label{#2}%`

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

1956 `\ifglsused{#2}%`

1957 `{%`

1958 `\protected@edef\@glo@text{%`

1959 `\csname gls@\@glo@type @display\endcsname`

1960 `{\glsentrytext{#2}}{\glsentrydesc{#2}}%`

1961 `{\glsentrysymbol{#2}}{#3}}%`

1962 `}%`

1963 `{%`

1964 `\protected@edef\@glo@text{%`

1965 `\csname gls@\@glo@type @displayfirst\endcsname`

1966 `{\glsentryfirst{#2}}{\glsentrydesc{#2}}%`

1967 `{\glsentrysymbol{#2}}{#3}}%`

1968 `}%`

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

1969 `\ifglsused{#2}%`

1970 `{%`


```

1971 \gls@link[#1]{#2}{%
1972 \expandafter\makefirstuc\expandafter{\@glo@text}}}%
1973 }%
1974 {%
1975 \gls@checkisacronymlist\@glo@type
1976 \ifthenelse
1977 {%
1978 \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
1979 \OR \NOT\boolean{glshyperfirst}}%
1980 }%
1981 {%
1982 \gls@link[#1,hyper=false]{#2}{%
1983 \expandafter\makefirstuc\expandafter{\@glo@text}}}%
1984 }%
1985 {%
1986 \gls@link[#1]{#2}{%
1987 \expandafter\makefirstuc\expandafter{\@glo@text}}}%
1988 }%
1989 }%

```

Indicate that this entry has now been used

```

1990 \ifKV@glslink@local
1991 \glslocalunset{#2}%
1992 \else
1993 \glsunset{#2}%
1994 \fi
1995 }%
1996 }

```

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

\GLS

```

1997 \newrobustcmd*{\GLS}{\@ifstar\@sGLS\@GLS}

```

Define the starred form:

```

1998 \newcommand*{\@sGLS}[1][\@GLS[hyper=false,#1]]

```

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

```

1999 \newcommand*{\@GLS}[2][\@GLS@#1]{%
2000 \new@ifnextchar[\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
2001 }

```

\@GLS@ Read in the final optional argument:

```

2002 \def\@GLS@#1#2[#3]{%
2003 \glsdoifexists{#2}%
2004 {%
2005 \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

2006 \def\@gls@link@opts{#1}%
2007 \def\@gls@link@label{#2}%
    Determine what the link text should be (this is stored in \@glo@text).
2008 \ifglsused{#2}%
2009 {%
2010 \def\@glo@text{%
2011 \csname gls@\@glo@type @display\endcsname
2012 {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}%
2013 }%
2014 }%
2015 {%
2016 \def\@glo@text{%
2017 \csname gls@\@glo@type @displayfirst\endcsname
2018 {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}%
2019 }%
2020 }%

```

Call \@gls@link If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

2021 \ifglsused{#2}%
2022 {%
2023 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2024 }%
2025 {%
2026 \gls@checkisacronymlist\@glo@type
2027 \ifthenelse
2028 {%
2029 \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2030 \OR \NOT\boolean{glshyperfirst}}{%
2031 \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
2032 }%
2033 {%
2034 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2035 }%
2036 }%

```

Indicate that this entry has now been used

```

2037 \ifKV@glslink@local
2038 \glslocalunset{#2}%
2039 \else
2040 \glsunset{#2}%
2041 \fi
2042 }%
2043 }

```

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

\glspl

```

2044 \newrobustcmd*{\glspl}{\@ifstar\@sglspl\@glspl}

```

Define the starred form:

```
2045 \newcommand*{\@glspl}[1][\@glspl[hyper=false,#1]]
```

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

```
2046 \newcommand*{\@glspl}[2][\%
2047   \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[]}%
2048 }
```

`\@glspl@` Read in the final optional argument:

```
2049 \def\@glspl@#1#2[#3]{%
2050   \glsdoifexists{#2}%
2051   {%
2052     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2053   \def\@gls@link@opts{#1}%
2054   \def\@gls@link@label{#2}%
```

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

```
2055   \ifglsused{#2}%
2056   {%
2057     \def\@glo@text{%
2058       \csname gls@\@glo@type @display\endcsname
2059       {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
2060       {\glsentrysymbolplural{#2}}{#3}}%
2061   }%
2062   {%
2063     \def\@glo@text{%
2064       \csname gls@\@glo@type @displayfirst\endcsname
2065       {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
2066       {\glsentrysymbolplural{#2}}{#3}}%
2067   }%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
2068   \ifglsused{#2}%
2069   {%
2070     \@gls@link[#1]{#2}{\@glo@text}%
2071   }%
2072   {%
2073     \gls@checkisacronymlist\@glo@type
2074     \ifthenelse
2075     {%
2076       \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2077       \OR \NOT\boolean{glshyperfirst}%
2078     }%
2079     {%
2080       \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
```

```

2081     }%
2082     {%
2083         \@gls@link[#1]{#2}{\@glo@text}%
2084     }%
2085 }%

```

Indicate that this entry has now been used

```

2086     \ifKV@glslink@local
2087         \glslocalunset{#2}%
2088     \else
2089         \glsunset{#2}%
2090     \fi
2091 }%
2092 }

```

`\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`

```

2093 \newrobustcmd*{\Glspl}{\@ifstar\@sGlspl\@Glspl}

```

Define the starred form:

```

2094 \newcommand*{\@sGlspl}[1][\@Glspl[hyper=false,#1]]{

```

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

```

2095 \newcommand*{\@Glspl}[2][\@Glspl@{#1}{#2}]{\@Glspl@{#1}{#2}[]}
2096 \new@ifnextchar[\@Glspl@{#1}{#2}]{\@Glspl@{#1}{#2}[]}
2097 }

```

`\@Glspl@` Read in the final optional argument:

```

2098 \def\@Glspl@#1#2[#3]{%
2099     \glsdoifexists{#2}%
2100     {%
2101         \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

2102     \def\@gls@link@opts{#1}%
2103     \def\@gls@link@label{#2}%
2104     \def\@gls@label{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`.

```

2105     \ifglsused{#2}%
2106     {%
2107         \protected@edef\@glo@text{%
2108             \csname gls@\@glo@type @display\endcsname
2109             {\glsentryplural{#2}}{\glsentrydescplural{#2}}%

```

```

2110      {\glsentrysymbolplural{#2}}{#3}}%
2111    }%
2112    {%
2113      \protected@edef\@glo@text{%
2114        \csname gls@\@glo@type @displayfirst\endcsname
2115        {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
2116        {\glsentrysymbolplural{#2}}{#3}}%
2117    }%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

2118    \ifglsused{#2}%
2119    {%
2120      \@gls@link[#1]{#2}{%
2121        \expandafter\makefirstuc\expandafter{\@glo@text}}%
2122    }%
2123    {%
2124      \gls@checkisacronymlist\@glo@type
2125      \ifthenelse
2126      {%
2127        \(\boolean{glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2128        \OR \NOT\boolean{glshyperfirst}}%
2129      {%
2130        \@gls@link[#1,hyper=false]{#2}{%
2131          \expandafter\makefirstuc\expandafter{\@glo@text}}%
2132        }%
2133      {%
2134        \@gls@link[#1]{#2}{%
2135          \expandafter\makefirstuc\expandafter{\@glo@text}}%
2136        }%
2137      }%
2138    }%

```

Indicate that this entry has now been used

```

2139    \ifKV@glslink@local
2140    \glslocalunset{#2}%
2141  \else
2142    \glsunset{#2}%
2143  \fi
2144 }%
2145 }

```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

`\GLSp1`

```

2146 \newrobustcmd*{\GLSp1}{\@ifstar\@sGLSp1\@GLSp1}

```

Define the starred form:

```

2147 \newcommand*{\@sGLSp1}[1][\@GLSp1[hyper=false,#1]]{

```

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

```
2148 \newcommand*{\@GLSp1}[2][]{%
2149   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2}[]}%
2150 }
```

\@GLSp1 Read in the final optional argument:

```
2151 \def\@GLSp1@#1#2[#3]{%
2152   \glsdoifexists{#2}%
2153   {%
2154     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2155   \def\@gls@link@opts{#1}%
2156   \def\@gls@link@label{#2}%
```

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

```
2157   \ifglsused{#2}%
2158   {%
2159     \def\@glo@text{%
2160       \csname gls@\@glo@type @display\endcsname
2161       {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
2162       {\glsentrysymbolplural{#2}}{#3}%
2163     }%
2164   }%
2165   {%
2166     \def\@glo@text{%
2167       \csname gls@\@glo@type @displayfirst\endcsname
2168       {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
2169       {\glsentrysymbolplural{#2}}{#3}%
2170     }%
2171   }%
```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```
2172   \ifglsused{#2}%
2173   {%
2174     \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2175   }%
2176   {%
2177     \gls@checkisacronymlist\@glo@type
2178     \ifthenelse
2179     {%
2180       \(\boolean{glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2181       \OR \NOT\boolean{glshyperfirst}%
2182     }%
2183     {%
2184       \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
2185     }%
```

```

2186      {%
2187      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2188      }%
2189      }%

```

Indicate that this entry has now been used

```

2190      \ifKV@glslink@local
2191      \glslocalunset{#2}%
2192      \else
2193      \glsunset{#2}%
2194      \fi
2195      }%
2196 }

```

`\glsdisp` `\glsdisp[<options>]{<label>}{<text>}` This is like `\gls` except that the link text is provided. This differs from `\glslink` in that it uses `\glsdisplay` or `\glsdisplayfirst` and unsets the first use flag.

First determine if we are using the starred form:

```

2197 \newrobustcmd*{\glsdisp}{\@ifstar\sglsdisp\@glsdisp}

```

Define the starred form:

```

\@sgls
2198 \newcommand*{\@sglsdisp}[1][\@glsdisp[hyper=false,#1]]{

```

Defined the un-starred form.

`\@glsdisp`

```

2199 \newcommand*{\@glsdisp}[3][\@glsdoifexists{#2}{%
2200   \glsdoifexists{#2}{%
2201     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

2202   \def\@gls@link@opts{#1}%
2203   \def\@gls@link@label{#2}%

```

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

```

2204   \ifglsused{#2}%
2205   {%
2206     \def\@glo@text{%
2207       \csname gls@\@glo@type @display\endcsname
2208       {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
2209   }%
2210   {%
2211     \def\@glo@text{%
2212       \csname gls@\@glo@type @displayfirst\endcsname
2213       {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
2214   }%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

2215 \ifglsused{#2}%
2216 {%
2217 \@gls@link[#1]{#2}{\@glo@text}%
2218 }%
2219 {%
2220 \gls@checkisacronymlist\@glo@type
2221 \ifthenelse{(\boolean{@glsisacronymlist}\AND
2222 \boolean{glsacrfootnote}) \OR \NOT\boolean{glshyperfirst}}%
2223 {%
2224 \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
2225 }%
2226 {%
2227 \@gls@link[#1]{#2}{\@glo@text}%
2228 }%
2229 }%

```

Indicate that this entry has now been used

```

2230 \ifKV@glslink@local
2231 \glslocalunset{#2}%
2232 \else
2233 \glsunset{#2}%
2234 \fi
2235 }%
2236 }

```

`\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`

```

2237 \newrobustcmd*{\glstext}{\@ifstar\@sglstext\@glstext}

```

Define the starred form:

```

2238 \newcommand*{\@sglstext}[1][\@glstext[hyper=false,#1]]{

```

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

```

2239 \newcommand*{\@glstext}[2][\@glstext@{#1}{#2}]{\@glstext@{#1}{#2}[]}
2240 \new@ifnextchar[\@glstext@{#1}{#2}]{\@glstext@{#1}{#2}[]}

```

Read in the final optional argument:

```

2241 \def\@glstext@#1#2[#3]{%
2242 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

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

```

2243 \protected@edef\@glo@text{\glsentrytext{#2}}%

```

Call `\@gls@link`

```

2244 \@gls@link[#1]{#2}{\@glo@text#3}%

```


2245 }%

2246 }

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

\GLStext

2247 \newrobustcmd*{\GLStext}{\@ifstar\@sGLStext\@GLStext}

Define the starred form:

2248 \newcommand*{\@sGLStext}[1][]{\@GLStext[hyper=false,#1]}

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

2249 \newcommand*{\@GLStext}[2][]{%

2250 \new@ifnextchar[\@GLStext@{#1}{#2}]{\@GLStext@{#1}{#2}[]}}

Read in the final optional argument:

2251 \def\@GLStext@#1#2[#3]{%

2252 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%

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

2253 \protected@edef\@glo@text{\glsentrytext{#2}}%

Call \@gls@link

2254 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%

2255 }%

2256 }

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

\Glstext

2257 \newrobustcmd*{\Glstext}{\@ifstar\@sGlstext\@Glstext}

Define the starred form:

2258 \newcommand*{\@sGlstext}[1][]{\@Glstext[hyper=false,#1]}

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

2259 \newcommand*{\@Glstext}[2][]{%

2260 \new@ifnextchar[\@Glstext@{#1}{#2}]{\@Glstext@{#1}{#2}[]}}

Read in the final optional argument:

2261 \def\@Glstext@#1#2[#3]{%

2262 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%

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

2263 \protected@edef\@glo@text{\glsentrytext{#2}}%

Call \@gls@link

2264 \@gls@link[#1]{#2}{%

2265 \expandafter\makefirstuc\expandafter{\@glo@text#3}}%

2266 }%

2267 }

`\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`

```
2268 \newrobustcmd*{\glsfirst}{\@ifstar\sglsfirst\glsfirst}
```

Define the starred form:

```
2269 \newcommand*{\sglsfirst}[1][]{\@glsfirst[hyper=false,#1]}
```

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

```
2270 \newcommand*{\@glsfirst}[2][]{%
```

```
2271 \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
2272 \def\@glsfirst@#1#2[#3]{%
```

```
2273 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

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

```
2274 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call `\@gls@link`

```
2275 \@gls@link[#1]{#2}{\@glo@text#3}%
```

```
2276 }%
```

```
2277 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
2278 \newrobustcmd*{\Glsfirst}{\@ifstar\sglsfirst\Glsfirst}
```

Define the starred form:

```
2279 \newcommand*{\sglsfirst}[1][]{\@Glsfirst[hyper=false,#1]}
```

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

```
2280 \newcommand*{\@Glsfirst}[2][]{%
```

```
2281 \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
2282 \def\@Glsfirst@#1#2[#3]{%
```

```
2283 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

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

```
2284 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call `\@gls@link`

```
2285 \@gls@link[#1]{#2}{%
```

```
2286 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
```

```
2287 }%
```

```
2288 }
```

`\GLSfirst` behaves like `\Glsfirst` except it displays the text in uppercase.

`\GLSfirst`

```
2289 \newrobustcmd*{\GLSfirst}{\@ifstar\@sGLSfirst\@GLSfirst}
```

Define the starred form:

```
2290 \newcommand*{\@sGLSfirst}[1] [] {\@GLSfirst[hyper=false,#1]}
```

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

```
2291 \newcommand*{\@GLSfirst}[2] [] {%
```

```
2292 \new@ifnextchar [{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2293 \def\@GLSfirst@#1#2[#3] {%
```

```
2294 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2295 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call `\@gls@link`

```
2296 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
```

```
2297 }%
```

```
2298 }
```

`\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`

```
2299 \newrobustcmd*{\glsplural}{\@ifstar\@sglsplural\@glsplural}
```

Define the starred form:

```
2300 \newcommand*{\@sglsplural}[1] [] {\@glsplural[hyper=false,#1]}
```

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

```
2301 \newcommand*{\@glsplural}[2] [] {%
```

```
2302 \new@ifnextchar [{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2303 \def\@glsplural@#1#2[#3] {%
```

```
2304 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2305 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call `\@gls@link`

```
2306 \@gls@link[#1]{#2}{\@glo@text#3}}%
```

```
2307 }%
```

```
2308 }
```

`\Glsplural` behaves like `\glsplural` except that the first letter is converted to uppercase.

`\Glsplural`

```
2309 \newrobustcmd*{\Glsplural}{\@ifstar\@sGlsplural\@Glsplural}
```

Define the starred form:

```
2310 \newcommand*{\@sGlsplural}[1] [] {\@Glsplural[hyper=false,#1]}
```

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

```
2311 \newcommand*{\@Glsplural}[2] [] {%
```

```
2312 \new@ifnextchar [\@Glsplural@{#1}{#2}]{\@Glsplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2313 \def\@Glsplural@#1#2[#3] {%
```

```
2314 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

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

```
2315 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call \@gls@link

```
2316 \@gls@link[#1]{#2}{%
```

```
2317 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
```

```
2318 }%
```

```
2319 }
```

\Glsplural behaves like \glsplural except that the text is converted to uppercase.

\Glsplural

```
2320 \newrobustcmd*{\Glsplural}{\@ifstar\@sGlsplural\@Glsplural}
```

Define the starred form:

```
2321 \newcommand*{\@sGlsplural}[1] [] {\@Glsplural[hyper=false,#1]}
```

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

```
2322 \newcommand*{\@Glsplural}[2] [] {%
```

```
2323 \new@ifnextchar [\@Glsplural@{#1}{#2}]{\@Glsplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2324 \def\@Glsplural@#1#2[#3] {%
```

```
2325 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

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

```
2326 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call \@gls@link

```
2327 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
```

```
2328 }%
```

```
2329 }
```

\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

```
2330 \newrobustcmd*{\glsfirstplural}{\@ifstar\@sglsfirstplural\@glsfirstplural}
```

Define the starred form:

```
2331 \newcommand*{\@sglsfirstplural}[1] [] {\@glsfirstplural[hyper=false,#1]}
    Defined the un-starred form. Need to determine if there is a final optional argument
2332 \newcommand*{\@glsfirstplural}[2] [] {%
2333 \new@ifnextchar [\@glsfirstplural@{#1}{#2}]{\@glsfirstplural@{#1}{#2} []}}
    Read in the final optional argument:
2334 \def\@glsfirstplural@#1#2[#3]{%
2335 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
2336 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
    Call \@gls@link
2337 \@gls@link[#1]{#2}{\@glo@text#3}%
2338 }%
2339 }
```

`\Glsfirstplural` behaves like `\glsfirstplural` except that the first letter is converted to uppercase.

`\Glsfirstplural`

```
2340 \newrobustcmd*{\Glsfirstplural}{\@ifstar\@sGlsfirstplural\@Glsfirstplural}
    Define the starred form:
2341 \newcommand*{\@sGlsfirstplural}[1] [] {\@Glsfirstplural[hyper=false,#1]}
    Defined the un-starred form. Need to determine if there is a final optional argument
2342 \newcommand*{\@Glsfirstplural}[2] [] {%
2343 \new@ifnextchar [\@Glsfirstplural@{#1}{#2}]{\@Glsfirstplural@{#1}{#2} []}}
    Read in the final optional argument:
2344 \def\@Glsfirstplural@#1#2[#3]{%
2345 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
2346 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
    Call \@gls@link
2347 \@gls@link[#1]{#2}{%
2348   \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2349 }%
2350 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
2351 \newrobustcmd*{\GLSfirstplural}{\@ifstar\@sGLSfirstplural\@GLSfirstplural}
```

Define the starred form:

```
2352 \newcommand*{\@sGLSfirstplural}[1] [] {\@GLSfirstplural[hyper=false,#1]}
```

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

```
2353 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
2354 \new@ifnextchar [\@GLSfirstplural@{#1}{#2}]{\@GLSfirstplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2355 \def\@GLSfirstplural@#1#2[#3] {%
```

```
2356 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2357 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
```

Call \@gls@link

```
2358 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
```

```
2359 }%
```

```
2360 }
```

\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

```
2361 \newrobustcmd*{\glsname}{\@ifstar\@sglsname\@glsname}
```

Define the starred form:

```
2362 \newcommand*{\@sglsname}[1] [] {\@glsname[hyper=false,#1]}
```

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

```
2363 \newcommand*{\@glsname}[2] [] {%
```

```
2364 \new@ifnextchar [\@glsname@{#1}{#2}]{\@glsname@{#1}{#2} []}}
```

Read in the final optional argument:

```
2365 \def\@glsname@#1#2[#3] {%
```

```
2366 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2367 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
2368 \@gls@link[#1]{#2}{\@glo@text#3}}%
```

```
2369 }%
```

```
2370 }
```

\Glsname behaves like \glsname except that the first letter is converted to uppercase.

\Glsname

```
2371 \newrobustcmd*{\Glsname}{\@ifstar\@sGlsname\@Glsname}
```

Define the starred form:

```
2372 \newcommand*{\@sGlsname}[1] [] {\@Glsname[hyper=false,#1]}
```

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

```
2373 \newcommand*{\@Glsname}[2] [] {%
2374 \new@ifnextchar [{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2375 \def\@Glsname@#1#2[#3] {%
2376 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2377 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
2378 \@gls@link[#1]{#2}{%
2379 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2380 }%
2381 }
```

\GLSname behaves like \glsname except that the link text is converted to uppercase.

\GLSname

```
2382 \newrobustcmd*{\@GLSname}{\@ifstar\@sGLSname\@GLSname}
```

Define the starred form:

```
2383 \newcommand*{\@sGLSname}[1] [] {\@GLSname[hyper=false,#1]}
```

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

```
2384 \newcommand*{\@GLSname}[2] [] {%
2385 \new@ifnextchar [{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2386 \def\@GLSname@#1#2[#3] {%
2387 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2388 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
2389 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2390 }%
2391 }
```

\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

```
2392 \newrobustcmd*{\glsdesc}{\@ifstar\@sglsdesc\@glsdesc}
```

Define the starred form:

```
2393 \newcommand*{\@sglsdesc}[1] [] {\@glsdesc[hyper=false,#1]}
```

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

```
2394 \newcommand*\@glsdesc}[2][]{%
2395 \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
2396 \def\@glsdesc@#1#2[#3]{%
2397 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2398 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
2399 \@gls@link[#1]{#2}{\@glo@text#3}%
2400 }%
2401 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
2402 \newrobustcmd*\@Glsdesc{\@ifstar\@sGlsdesc\@Glsdesc}
```

Define the starred form:

```
2403 \newcommand*\@sGlsdesc}[1][]{\@Glsdesc[hyper=false,#1]}
```

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

```
2404 \newcommand*\@Glsdesc}[2][]{%
2405 \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
2406 \def\@Glsdesc@#1#2[#3]{%
2407 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2408 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
2409 \@gls@link[#1]{#2}{%
2410 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2411 }%
2412 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
2413 \newrobustcmd*\@GLSdesc{\@ifstar\@sGLSdesc\@GLSdesc}
```

Define the starred form:

```
2414 \newcommand*\@sGLSdesc}[1][]{\@GLSdesc[hyper=false,#1]}
```


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

```
2415 \newcommand*{\@GLSdesc}[2] [] {%
2416 \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
2417 \def\@GLSdesc@#1#2[#3] {%
2418 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2419 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
2420 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2421 }%
2422 }
```

\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

```
2423 \newrobustcmd*{\glsdescplural}{\@ifstar\sglsdescplural\@glsdescplural}
```

Define the starred form:

```
2424 \newcommand*{\@sglsdescplural}[1] [] {\@glsdescplural[hyper=false,#1]}
```

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

```
2425 \newcommand*{\@glsdescplural}[2] [] {%
2426 \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2427 \def\@glsdescplural@#1#2[#3] {%
2428 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2429 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
2430 \@gls@link[#1]{#2}{\@glo@text#3}}%
2431 }%
2432 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
2433 \newrobustcmd*{\Glsdescplural}{\@ifstar\@sGlsdescplural\@Glsdescplural}
```

Define the starred form:

```
2434 \newcommand*{\@sGlsdescplural}[1] [] {\@Glsdescplural[hyper=false,#1]}
```

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

```
2435 \newcommand*{\@Glsdescplural}[2] [] {%
2436 \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}]{\@Glsdescplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2437 \def\@Glsdescplural@#1#2[#3] {%
2438 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2439 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
2440 \@gls@link[#1]{#2}{%
2441 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2442 }%
2443 }
```

\GLSdescplural behaves like \glsdescplural except that the link text is converted to uppercase.

\GLSdescplural

```
2444 \newrobustcmd*{\GLSdescplural}{\@ifstar\@sGLSdescplural\@GLSdescplural}
```

Define the starred form:

```
2445 \newcommand*{\@sGLSdescplural}[1] [] {\@GLSdescplural[hyper=false,#1]}
```

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

```
2446 \newcommand*{\@GLSdescplural}[2] [] {%
2447 \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}]{\@GLSdescplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2448 \def\@GLSdescplural@#1#2[#3] {%
2449 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2450 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
2451 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2452 }%
2453 }
```

\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

```
2454 \newrobustcmd*{\glssymbol}{\@ifstar\@sglssymbol\@glssymbol}
```

Define the starred form:

```
2455 \newcommand*{\@sglssymbol}[1] [] {\@glssymbol[hyper=false,#1]}
```

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

```
2456 \newcommand*{\@glssymbol}[2] [] {%
2457 \new@ifnextchar [{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
2458 \def\@glssymbol@#1#2[#3] {%
2459 \glsoifexists{#2}{\edef\@glo@type{\glentrytype{#2}}%
```

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

```
2460 \protected@edef\@glo@text{\glentrysymbol{#2}}%
```

Call \@gls@link

```
2461 \@gls@link[#1]{#2}{\@glo@text#3}%
2462 }%
2463 }
```

\Glsymbol behaves like \glssymbol except that the first letter is converted to uppercase.

\Glsymbol

```
2464 \newrobustcmd*{\Glsymbol}{\@ifstar\@sGlsymbol\@Glsymbol}
```

Define the starred form:

```
2465 \newcommand*{\@sGlsymbol}[1] [] {\@Glsymbol[hyper=false,#1]}
```

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

```
2466 \newcommand*{\@Glsymbol}[2] [] {%
2467 \new@ifnextchar [{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
2468 \def\@Glsymbol@#1#2[#3] {%
2469 \glsoifexists{#2}{\edef\@glo@type{\glentrytype{#2}}%
```

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

```
2470 \protected@edef\@glo@text{\glentrysymbol{#2}}%
```

Call \@gls@link

```
2471 \@gls@link[#1]{#2}{%
2472 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2473 }%
2474 }
```

\GLSsymbol behaves like \glssymbol except that the link text is converted to uppercase.

\GLSsymbol

```
2475 \newrobustcmd*{\GLSsymbol}{\@ifstar\@sGLSsymbol\@GLSsymbol}
```

Define the starred form:

```
2476 \newcommand*{\@sGLSsymbol}[1] [] {\@GLSsymbol[hyper=false,#1]}
```

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

```
2477 \newcommand*{\@GLSSymbol}[2] [] {%
2478 \new@ifnextchar [{\@GLSSymbol@{#1}{#2}}{\@GLSSymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
2479 \def\@GLSSymbol@#1#2[#3] {%
2480 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2481 \protected@edef\@glo@text{\glsentrysymbol{#2}}%
```

Call \@gls@link

```
2482 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2483 }%
2484 }
```

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

\glsymbolplural

```
2485 \newrobustcmd*{\glsymbolplural}{\@ifstar\sglsymbolplural\@glsymbolplural}
```

Define the starred form:

```
2486 \newcommand*{\@sglsymbolplural}[1] [] {\@glsymbolplural[hyper=false,#1]}
```

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

```
2487 \newcommand*{\@glsymbolplural}[2] [] {%
2488 \new@ifnextchar [{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2489 \def\@glsymbolplural@#1#2[#3] {%
2490 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2491 \protected@edef\@glo@text{\glsentrysymbolplural{#2}}%
```

Call \@gls@link

```
2492 \@gls@link[#1]{#2}{\@glo@text#3}}%
2493 }%
2494 }
```

\Glsymbolplural behaves like \glsymbolplural except that the first letter is converted to uppercase.

\Glsymbolplural

```
2495 \newrobustcmd*{\Glsymbolplural}{\@ifstar\@sGlsymbolplural\@Glsymbolplural}
```

Define the starred form:

```
2496 \newcommand*{\@sGlsymbolplural}[1] [] {\@Glsymbolplural[hyper=false,#1]}
```

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

```
2497 \newcommand*{\@Glssymbolplural}[2] [] {%
2498 \new@ifnextchar [{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2499 \def\@Glssymbolplural@#1#2[#3] {%
2500 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}%
```

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

```
2501 \protected@edef\@glo@text{\glstrysymbolplural{#2}}%
```

Call \@gls@link

```
2502 \@gls@link[#1]{#2}{%
2503   \expandafter\makefirstuc\expandafter{\@glo@text#3}}%
2504 }%
2505 }
```

\GLSsymbolplural behaves like \glsymbolplural except that the link text is converted to uppercase.

\GLSsymbolplural

```
2506 \newrobustcmd*{\GLSsymbolplural}{\@ifstar\@sGLSsymbolplural\@GLSsymbolplural}
```

Define the starred form:

```
2507 \newcommand*{\@sGLSsymbolplural}[1] [] {\@GLSsymbolplural[hyper=false,#1]}
```

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

```
2508 \newcommand*{\@GLSsymbolplural}[2] [] {%
2509 \new@ifnextchar [{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2510 \def\@GLSsymbolplural@#1#2[#3] {%
2511 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}%
```

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

```
2512 \protected@edef\@glo@text{\glstrysymbolplural{#2}}%
```

Call \@gls@link

```
2513 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2514 }%
2515 }
```

\glsuseri behaves like \gls except it always uses the value given by the user1 key and it doesn't mark the entry as used.

\glsuseri

```
2516 \newrobustcmd*{\glsuseri}{\@ifstar\@sglsuseri\@glsuseri}
```

Define the starred form:

```
2517 \newcommand*{\@sglsuseri}[1] [] {\@glsuseri[hyper=false,#1]}
```

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

```
2518 \newcommand*{\@glsuseri}[2] [] {%
2519 \new@ifnextchar [\@glsuseri@{#1}{#2}]{\@glsuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
2520 \def\@glsuseri@#1#2[#3] {%
2521 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2522 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
2523 \@gls@link[#1]{#2}{\@glo@text#3}%
2524 }%
2525 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
2526 \newrobustcmd*{\Glsuseri}{\@ifstar\@sGlsuseri\@Glsuseri}
```

Define the starred form:

```
2527 \newcommand*{\@sGlsuseri}[1] [] {\@Glsuseri[hyper=false,#1]}
```

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

```
2528 \newcommand*{\@Glsuseri}[2] [] {%
2529 \new@ifnextchar [\@Glsuseri@{#1}{#2}]{\@Glsuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
2530 \def\@Glsuseri@#1#2[#3] {%
2531 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2532 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
2533 \@gls@link[#1]{#2}{%
2534 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2535 }%
2536 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
2537 \newrobustcmd*{\GLSuseri}{\@ifstar\@sGLSuseri\@GLSuseri}
```

Define the starred form:

```
2538 \newcommand*{\@sGLSuseri}[1] [] {\@GLSuseri[hyper=false,#1]}
```

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

```
2539 \newcommand*{\@GLSuseri}[2] [] {%
2540 \new@ifnextchar [\@GLSuseri@{#1}{#2}]{\@GLSuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
2541 \def\@GLSuseri@#1#2[#3] {%
2542 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2543 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
2544 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2545 }%
2546 }
```

\glsuserii behaves like \gls except it always uses the value given by the user2 key and it doesn't mark the entry as used.

\glsuserii

```
2547 \newrobustcmd*{\glsuserii}{\@ifstar\@sglsuserii\@glsuserii}
```

Define the starred form:

```
2548 \newcommand*{\@sglsuserii}[1] [] {\@glsuserii[hyper=false,#1]}
```

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

```
2549 \newcommand*{\@glsuserii}[2] [] {%
2550 \new@ifnextchar [\@glsuserii@{#1}{#2}]{\@glsuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2551 \def\@glsuserii@#1#2[#3] {%
2552 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2553 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
2554 \@gls@link[#1]{#2}{\@glo@text#3}}%
2555 }%
2556 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
2557 \newrobustcmd*{\Glsuserii}{\@ifstar\@sGlsuserii\@Glsuserii}
```

Define the starred form:

```
2558 \newcommand*{\@sGlsuserii}[1] [] {\@Glsuserii[hyper=false,#1]}
```

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

```
2559 \newcommand*{\@Glsuserii}[2] [] {%
2560 \new@ifnextchar [{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2561 \def\@Glsuserii@#1#2[#3] {%
2562 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2563 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
2564 \@gls@link[#1]{#2}{%
2565 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2566 }%
2567 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
2568 \newrobustcmd*{\GLSuserii}{\@ifstar\@sGLSuserii\@GLSuserii}
```

Define the starred form:

```
2569 \newcommand*{\@sGLSuserii}[1] [] {\@GLSuserii[hyper=false,#1]}
```

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

```
2570 \newcommand*{\@GLSuserii}[2] [] {%
2571 \new@ifnextchar [{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2572 \def\@GLSuserii@#1#2[#3] {%
2573 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2574 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
2575 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2576 }%
2577 }
```

\glsuseriii behaves like \gls except it always uses the value given by the user3 key and it doesn't mark the entry as used.

\glsuseriii

```
2578 \newrobustcmd*{\glsuseriii}{\@ifstar\@sglsuseriii\@glsuseriii}
```

Define the starred form:

```
2579 \newcommand*{\@sglsuseriii}[1] [] {\@glsuseriii[hyper=false,#1]}
```


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

```
2580 \newcommand*{\@glsuseriii}[2] [] {%
2581 \new@ifnextchar [{\@glsuseriii@{#1}{#2}}]{\@glsuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2582 \def\@glsuseriii@#1#2[#3]{%
2583 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

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

```
2584 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
2585 \@gls@link[#1]{#2}{\@glo@text#3}%
2586 }%
2587 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
2588 \newrobustcmd*{\Glsuseriii}{\@ifstar\@sGlsuseriii\@Glsuseriii}
```

Define the starred form:

```
2589 \newcommand*{\@sGlsuseriii}[1] [] {\@Glsuseriii[hyper=false,#1]}
```

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

```
2590 \newcommand*{\@Glsuseriii}[2] [] {%
2591 \new@ifnextchar [{\@Glsuseriii@{#1}{#2}}]{\@Glsuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2592 \def\@Glsuseriii@#1#2[#3]{%
2593 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

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

```
2594 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
2595 \@gls@link[#1]{#2}{%
2596 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2597 }%
2598 }
```

\GLSuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
2599 \newrobustcmd*{\GLSuseriii}{\@ifstar\@sGLSuseriii\@GLSuseriii}
```

Define the starred form:

```
2600 \newcommand*{\@sGLSuseriii}[1] [] {\@GLSuseriii[hyper=false,#1]}
```

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

```
2601 \newcommand*{\@GLSuseriii}[2] [] {%
2602 \new@ifnextchar [\@GLSuseriii@{#1}{#2}]{\@GLSuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2603 \def\@GLSuseriii@#1#2[#3] {%
2604 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2605 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
2606 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2607 }%
2608 }
```

\glsuseriv behaves like \gls except it always uses the value given by the user4 key and it doesn't mark the entry as used.

\glsuseriv

```
2609 \newrobustcmd*{\glsuseriv}{\@ifstar\@sglsuseriv\@glsuseriv}
```

Define the starred form:

```
2610 \newcommand*{\@sglsuseriv}[1] [] {\@glsuseriv[hyper=false,#1]}
```

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

```
2611 \newcommand*{\@glsuseriv}[2] [] {%
2612 \new@ifnextchar [\@glsuseriv@{#1}{#2}]{\@glsuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
2613 \def\@glsuseriv@#1#2[#3] {%
2614 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2615 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
2616 \@gls@link[#1]{#2}{\@glo@text#3}%
2617 }%
2618 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
2619 \newrobustcmd*{\Glsuseriv}{\@ifstar\@sGlsuseriv\@Glsuseriv}
```

Define the starred form:

```
2620 \newcommand*{\@sGlsuseriv}[1] [] {\@Glsuseriv[hyper=false,#1]}
```

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

```
2621 \newcommand*{\@Glsuseriv}[2] [] {%
2622 \new@ifnextchar [{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
2623 \def\@Glsuseriv@#1#2[#3] {%
2624 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2625 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
2626 \@gls@link[#1]{#2}{%
2627 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2628 }%
2629 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
2630 \newrobustcmd*{\GLSuseriv}{\@ifstar\@sGLSuseriv\@GLSuseriv}
```

Define the starred form:

```
2631 \newcommand*{\@sGLSuseriv}[1] [] {\@GLSuseriv[hyper=false,#1]}
```

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

```
2632 \newcommand*{\@GLSuseriv}[2] [] {%
2633 \new@ifnextchar [{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
2634 \def\@GLSuseriv@#1#2[#3] {%
2635 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2636 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
2637 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2638 }%
2639 }
```

\glsuserv behaves like \gls except it always uses the value given by the user5 key and it doesn't mark the entry as used.

\glsuserv

```
2640 \newrobustcmd*{\glsuserv}{\@ifstar\@sglsuserv\@glsuserv}
```

Define the starred form:

```
2641 \newcommand*{\@sglsuserv}[1] [] {\@glsuserv[hyper=false,#1]}
```

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

```
2642 \newcommand*{\@glsuserv}[2] [] {%
2643 \new@ifnextchar [\@glsuserv@{#1}{#2}]{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2644 \def\@glsuserv@#1#2[#3] {%
2645 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2646 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2647 \@gls@link[#1]{#2}{\@glo@text#3}%
2648 }%
2649 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
2650 \newrobustcmd*{\Glsuserv}{\@ifstar\@sGlsuserv\@Glsuserv}
```

Define the starred form:

```
2651 \newcommand*{\@sGlsuserv}[1] [] {\@Glsuserv[hyper=false,#1]}
```

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

```
2652 \newcommand*{\@Glsuserv}[2] [] {%
2653 \new@ifnextchar [\@Glsuserv@{#1}{#2}]{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2654 \def\@Glsuserv@#1#2[#3] {%
2655 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2656 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2657 \@gls@link[#1]{#2}{%
2658 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2659 }%
2660 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
2661 \newrobustcmd*{\GLSuserv}{\@ifstar\@sGLSuserv\@GLSuserv}
```

Define the starred form:

```
2662 \newcommand*{\@sGLSuserv}[1] [] {\@GLSuserv[hyper=false,#1]}
```

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

```
2663 \newcommand*{\@GLSuserv}[2] [] {%
2664 \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2665 \def\@GLSuserv@#1#2[#3] {%
2666 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2667 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2668 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2669 }%
2670 }
```

\glsuservi behaves like \gls except it always uses the value given by the user6 key and it doesn't mark the entry as used.

\glsuservi

```
2671 \newrobustcmd*{\glsuservi}{\@ifstar\sglsuservi\@glsuservi}
```

Define the starred form:

```
2672 \newcommand*{\@sglsuservi}[1] [] {\@glsuservi[hyper=false,#1]}
```

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

```
2673 \newcommand*{\@glsuservi}[2] [] {%
2674 \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2675 \def\@glsuservi@#1#2[#3] {%
2676 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2677 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2678 \@gls@link[#1]{#2}{\@glo@text#3}}%
2679 }%
2680 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
2681 \newrobustcmd*{\Glsuservi}{\@ifstar\sglsuservi\@Glsuservi}
```

Define the starred form:

```
2682 \newcommand*{\@sglsuservi}[1] [] {\@Glsuservi[hyper=false,#1]}
```

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

```
2683 \newcommand*{\@Glsuservi}[2] [] {%
2684 \new@ifnextchar [{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}}
```

Read in the final optional argument:

```
2685 \def\@Glsuservi@#1#2[#3] {%
2686 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2687 \protected@edef\@glo@text{\glsentryuservi{#2}}%
```

Call \@gls@link

```
2688 \@gls@link[#1]{#2}{%
2689 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2690 }%
2691 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
2692 \newrobustcmd*{\GLSuservi}{\@ifstar\@sGLSuservi\@GLSuservi}
```

Define the starred form:

```
2693 \newcommand*{\@sGLSuservi}[1] [] {\@GLSuservi[hyper=false,#1]}
```

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

```
2694 \newcommand*{\@GLSuservi}[2] [] {%
2695 \new@ifnextchar [{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}}
```

Read in the final optional argument:

```
2696 \def\@GLSuservi@#1#2[#3] {%
2697 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2698 \protected@edef\@glo@text{\glsentryuservi{#2}}%
```

Call \@gls@link

```
2699 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2700 }%
2701 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
2702 \newrobustcmd*{\acrshort}{\@ifstar\s@acrshort\ns@acrshort}
```

Define the starred form:

```
2703 \newcommand*{\s@acrshort}[2] [] {%
2704 \new@ifnextchar [{\@acrshort{hyper=false,#1}{#2}}%
2705 {\@acrshort{hyper=false,#1}{#2} []}%
2706 }
```

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

```
2707 \newcommand*{\ns@acrshort}[2] [] {%
2708   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}%
2709 }
```

Read in the final optional argument:

```
2710 \def\@acrshort#1#2[#3] {%
2711   \glsdoifexists{#2}%
2712   {%
2713     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2714     \protected@edef\@glo@text{\glsentryshort{#2}}%
```

Call \@gls@link

```
2715     \@gls@link[#1]{#2}{\acronymfont{\@glo@text}#3}%
2716   }%
2717 }
```

\Acrshort

```
2718 \newrobustcmd*{\Acrshort}{\@ifstar\s@Acrshort\ns@Acrshort}
```

Define the starred form:

```
2719 \newcommand*{\s@Acrshort}[2] [] {%
2720   \new@ifnextchar[{\@Acrshort{hyper=false,#1}{#2}}%
2721   {\@Acrshort{hyper=false,#1}{#2} []}%
2722 }
```

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

```
2723 \newcommand*{\ns@Acrshort}[2] [] {%
2724   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}%
2725 }
```

Read in the final optional argument:

```
2726 \def\@Acrshort#1#2[#3] {%
2727   \glsdoifexists{#2}%
2728   {%
2729     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2730     \protected@edef\@glo@text{\glsentryshort{#2}}%
```

Call \@gls@link

```
2731     \@gls@link[#1]{#2}%
2732   {%
2733     \acronymfont{\expandafter\makefirstuc\expandafter{\@glo@text}}#3%
2734   }%
2735 }%
2736 }
```

\ACRshort

```
2737 \newrobustcmd*{\ACRshort}{\@ifstar\s@ACRshort\ns@ACRshort}
```

Define the starred form:

```
2738 \newcommand*{\s@ACRshort}[2] [] {%
2739   \new@ifnextchar[{\@ACRshort{hyper=false,#1}{#2}}%
2740   {\@ACRshort{hyper=false,#1}{#2} []}%
2741 }
```

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

```
2742 \newcommand*{\ns@ACRshort}[2] [] {%
2743   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}%
2744 }
```

Read in the final optional argument:

```
2745 \def\@ACRshort#1#2[#3] {%
2746   \glsdoifexists{#2}%
2747   {%
2748     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2749   \protected@edef\@glo@text{\glsentryshort{#2}}%
```

Call \@gls@link

```
2750   \@gls@link[#1]{#2}{\acronymfont{\MakeUppercase{\@glo@text#3}}}%
2751   }%
2752 }
```

Short plural:

\acrshortpl

```
2753 \newrobustcmd*{\acrshortpl}{\@ifstar\s@acrshortpl\ns@acrshortpl}
```

Define the starred form:

```
2754 \newcommand*{\s@acrshortpl}[2] [] {%
2755   \new@ifnextchar[{\@acrshortpl{hyper=false,#1}{#2}}%
2756   {\@acrshortpl{hyper=false,#1}{#2} []}%
2757 }
```

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

```
2758 \newcommand*{\ns@acrshortpl}[2] [] {%
2759   \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
2760 }
```

Read in the final optional argument:

```
2761 \def\@acrshortpl#1#2[#3] {%
2762   \glsdoifexists{#2}%
2763   {%
2764     \edef\@glo@type{\glsentrytype{#2}}%
```


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

```
2765 \protected@edef\@glo@text{\glentryshortpl{#2}}%
```

Call \@gls@link

```
2766 \@gls@link[#1]{#2}{\acronymfont{\@glo@text}{#3}}%
```

```
2767 }%
```

```
2768 }
```

\Acrshortpl

```
2769 \newrobustcmd*{\Acrshortpl}{\@ifstar\s@Acrshortpl\ns@Acrshortpl}
```

Define the starred form:

```
2770 \newcommand*{\s@Acrshortpl}[2][]{%}
```

```
2771 \new@ifnextchar[{\@Acrshortpl{hyper=false,#1}{#2}}%
```

```
2772 {\@Acrshortpl{hyper=false,#1}{#2}[]}%
```

```
2773 }
```

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

```
2774 \newcommand*{\ns@Acrshortpl}[2][]{%}
```

```
2775 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2}[]}%
```

```
2776 }
```

Read in the final optional argument:

```
2777 \def\@Acrshortpl#1#2[#3]{%
```

```
2778 \glsdoifexists{#2}}%
```

```
2779 {%
```

```
2780 \edef\@glo@type{\glentrytype{#2}}%
```

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

```
2781 \protected@edef\@glo@text{\glentryshortpl{#2}}%
```

Call \@gls@link

```
2782 \@gls@link[#1]{#2}}%
```

```
2783 {%
```

```
2784 \acronymfont{\expandafter\makefirstuc\expandafter{\@glo@text}}{#3}}%
```

```
2785 }%
```

```
2786 }%
```

```
2787 }
```

\ACRshortpl

```
2788 \newrobustcmd*{\ACRshortpl}{\@ifstar\s@ACRshortpl\ns@ACRshortpl}
```

Define the starred form:

```
2789 \newcommand*{\s@ACRshortpl}[2][]{%}
```

```
2790 \new@ifnextchar[{\@ACRshortpl{hyper=false,#1}{#2}}%
```

```
2791 {\@ACRshortpl{hyper=false,#1}{#2}[]}%
```

```
2792 }
```

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

```
2793 \newcommand*{\ns@ACRshortpl}[2] [] {%
2794   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
2795 }
```

Read in the final optional argument:

```
2796 \def\@ACRshortpl#1#2[#3] {%
2797   \glsdoifexists{#2}%
2798   {%
2799     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2800   \protected@edef\@glo@text{\glsentryshortpl{#2}}%
```

Call \@gls@link

```
2801   \@gls@link[#1]{#2}{\acronymfont{\MakeUppercase{\@glo@text#3}}}%
2802   }%
2803 }
```

\acrlong

```
2804 \newrobustcmd*{\acrlong}{\@ifstar\s@acrlong\ns@acrlong}
```

Define the starred form:

```
2805 \newcommand*{\s@acrlong}[2] [] {%
2806   \new@ifnextchar[{\@acrlong{hyper=false,#1}{#2}}%
2807   {\@acrlong{hyper=false,#1}{#2} []}%
2808 }
```

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

```
2809 \newcommand*{\ns@acrlong}[2] [] {%
2810   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
2811 }
```

Read in the final optional argument:

```
2812 \def\@acrlong#1#2[#3] {%
2813   \glsdoifexists{#2}%
2814   {%
2815     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2816   \protected@edef\@glo@text{\glsentrylong{#2}}%
```

Call \@gls@link

```
2817   \@gls@link[#1]{#2}{\@glo@text#3}%
2818   }%
2819 }
```

\Acrlong

```
2820 \newrobustcmd*{\Acrlong}{\@ifstar\s@Acrlong\ns@Acrlong}
```

Define the starred form:

```
2821 \newcommand*{\s@Acrlong}[2] [] {%
2822   \new@ifnextchar[{\@Acrlong{hyper=false,#1}{#2}}}%
2823   {\@Acrlong{hyper=false,#1}{#2} []}%
2824 }
```

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

```
2825 \newcommand*{\ns@Acrlong}[2] [] {%
2826   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}%
2827 }
```

Read in the final optional argument:

```
2828 \def\@Acrlong#1#2[#3] {%
2829   \glsdoifexists{#2}%
2830   {%
2831     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2832   \protected@edef\@glo@text{\glsentrylong{#2}}%
```

Call \@gls@link

```
2833   \@gls@link[#1]{#2}%
2834   {%
2835     \expandafter\makefirstuc\expandafter{\@glo@text}#3%
2836   }%
2837 }%
2838 }
```

\ACRlong

```
2839 \newrobustcmd*{\ACRlong}{\@ifstar\s@ACRlong\ns@ACRlong}
```

Define the starred form:

```
2840 \newcommand*{\s@ACRlong}[2] [] {%
2841   \new@ifnextchar[{\@ACRlong{hyper=false,#1}{#2}}}%
2842   {\@ACRlong{hyper=false,#1}{#2} []}%
2843 }
```

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

```
2844 \newcommand*{\ns@ACRlong}[2] [] {%
2845   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2} []}%
2846 }
```

Read in the final optional argument:

```
2847 \def\@ACRlong#1#2[#3] {%
2848   \glsdoifexists{#2}%
2849   {%
2850     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2851   \protected@edef\@glo@text{\glsentrylong{#2}}%
```

Call \@gls@link

```
2852 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%  
2853 }%  
2854 }
```

Short plural:

\acrlongpl

```
2855 \newrobustcmd*{\acrlongpl}{\@ifstar\s@acrlongpl\@ns@acrlongpl}
```

Define the starred form:

```
2856 \newcommand*{\s@acrlongpl}[2] [] {%  
2857 \new@ifnextchar[{\@acrlongpl{hyper=false,#1}{#2}}%  
2858 {\@acrlongpl{hyper=false,#1}{#2} []}%  
2859 }
```

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

```
2860 \newcommand*{\ns@acrlongpl}[2] [] {%  
2861 \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2} []}%  
2862 }
```

Read in the final optional argument:

```
2863 \def\@acrlongpl#1#2[#3] {%  
2864 \glsdoifexists{#2}%  
2865 {%  
2866 \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2867 \protected@edef\@glo@text{\glsentrylongpl{#2}}%
```

Call \@gls@link

```
2868 \@gls@link[#1]{#2}{\@glo@text#3}%  
2869 }%  
2870 }
```

\Acrlongpl

```
2871 \newrobustcmd*{\Acrlongpl}{\@ifstar\s@Acrlongpl\@ns@Acrlongpl}
```

Define the starred form:

```
2872 \newcommand*{\s@Acrlongpl}[2] [] {%  
2873 \new@ifnextchar[{\@Acrlongpl{hyper=false,#1}{#2}}%  
2874 {\@Acrlongpl{hyper=false,#1}{#2} []}%  
2875 }
```

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

```
2876 \newcommand*{\ns@Acrlongpl}[2] [] {%  
2877 \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%  
2878 }
```

Read in the final optional argument:

```
2879 \def\@Acrlongpl#1#2[#3]{%
2880   \glsdoifexists{#2}%
2881   {%
2882     \edef\@glo@type{\glsentrytype{#2}}%
2883     \protected@edef\@glo@text{\glsentrylongpl{#2}}%
2884     \Call\@gls@link
2885     {%
2886       \expandafter\makefirstuc\expandafter{\@glo@text}#3%
2887     }%
2888   }%
2889 }
```

\ACRlongpl

```
2890 \newrobustcmd*{\ACRlongpl}{\@ifstar\s@ACRlongpl\ns@ACRlongpl}
```

Define the starred form:

```
2891 \newcommand*{\s@ACRlongpl}[2][ ]{%
2892   \new@ifnextchar[{\@ACRlongpl{hyper=false,#1}{#2}}%
2893   {\@ACRlongpl{hyper=false,#1}{#2}[ ]}%
2894 }
```

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

```
2895 \newcommand*{\ns@ACRlongpl}[2][ ]{%
2896   \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[ ]}%
2897 }
```

Read in the final optional argument:

```
2898 \def\@ACRlongpl#1#2[#3]{%
2899   \glsdoifexists{#2}%
2900   {%
2901     \edef\@glo@type{\glsentrytype{#2}}%
2902     \protected@edef\@glo@text{\glsentrylongpl{#2}}%
2903     \Call\@gls@link
2904     {\@gls@link{#1}{#2}{\MakeUppercase{\@glo@text#3}}%
2905   }
```

1.10.2 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`

```
2906 \newcommand*{\glsentryname}[1]{\csname glo@#1@name\endcsname}
```

`\Glsentryname`

```
2907 \newcommand*{\Glsentryname}[1]{%
2908 \protected@edef\@glo@text{\csname glo@#1@name\endcsname}%
2909 \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`

```
2910 \newcommand*{\glsentrydesc}[1]{\csname glo@#1@desc\endcsname}
```

`\Glsentrydesc`

```
2911 \newcommand*{\Glsentrydesc}[1]{%
2912 \protected@edef\@glo@text{\csname glo@#1@desc\endcsname}%
2913 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

`\glsentrydescplural`

```
2914 \newcommand*{\glsentrydescplural}[1]{%
2915 \csname glo@#1@descplural\endcsname}
```

`\Glsentrydescplural`

```
2916 \newcommand*{\Glsentrydescplural}[1]{%
2917 \protected@edef\@glo@text{\csname glo@#1@descplural\endcsname}%
2918 \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`

```
2919 \newcommand*{\glsentrytext}[1]{\csname glo@#1@text\endcsname}
```

`\Glsentrytext`

```
2920 \newcommand*{\Glsentrytext}[1]{%
2921 \protected@edef\@glo@text{\csname glo@#1@text\endcsname}%
2922 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form:

`\glsentryplural`

```
2923 \newcommand*{\glsentryplural}[1]{\csname glo@#1@plural\endcsname}
```

`\Glsentryplural`

```
2924 \newcommand*{\Glsentryplural}[1]{%
2925 \protected@edef\@glo@text{\csname glo@#1@plural\endcsname}%
2926 \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`

```
2927 \newcommand*{\glsentrysymbol}[1]{\csname glo@#1@symbol\endcsname}
```

`\Glsentrysymbol`

```
2928 \newcommand*{\Glsentrysymbol}[1]{%
2929 \protected@edef\@glo@text{\csname glo@#1@symbol\endcsname}%
2930 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

`\glsentrysymbolplural`

```
2931 \newcommand*{\glsentrysymbolplural}[1]{%
2932 \csname glo@#1@symbolplural\endcsname}
```

`\Glsentrysymbolplural`

```
2933 \newcommand*{\Glsentrysymbolplural}[1]{%
2934 \protected@edef\@glo@text{\csname glo@#1@symbolplural\endcsname}%
2935 \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`

```
2936 \newcommand*{\glsentryfirst}[1]{\csname glo@#1@first\endcsname}
```

`\Glsentryfirst`

```
2937 \newcommand*{\Glsentryfirst}[1]{%
2938 \protected@edef\@glo@text{\csname glo@#1@first\endcsname}%
2939 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form (as specified by the `firstplural` key when the entry was defined).

`\glsentryfirstplural`

```
2940 \newcommand*{\glsentryfirstplural}[1]{%
2941 \csname glo@#1@firstpl\endcsname}
```

Glsentryfirstplural

```
2942 \newcommand*{\Glsentryfirstplural}[1]{%
2943 \protected@edef\@glo@text{\csname glo@#1@firstpl\endcsname}%
2944 \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

```
2945 \newcommand*{\glsentrytype}[1]{\csname glo@#1@type\endcsname}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
2946 \newcommand*{\glsentrysort}[1]{\csname glo@#1@sort\endcsname}
```

\glsentryuseri Get the first user key (as specified by the user1 when the entry was defined).
The argument is the label associated with the entry.

```
2947 \newcommand*{\glsentryuseri}[1]{\csname glo@#1@useri\endcsname}
```

\Glsentryuseri

```
2948 \newcommand*{\Glsentryuseri}[1]{%
2949 \protected@edef\@glo@text{\csname glo@#1@useri\endcsname}%
2950 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuserii Get the second user key (as specified by the user2 when the entry was defined).
The argument is the label associated with the entry.

```
2951 \newcommand*{\glsentryuserii}[1]{\csname glo@#1@userii\endcsname}
```

\Glsentryuserii

```
2952 \newcommand*{\Glsentryuserii}[1]{%
2953 \protected@edef\@glo@text{\csname glo@#1@userii\endcsname}%
2954 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriii Get the third user key (as specified by the user3 when the entry was defined).
The argument is the label associated with the entry.

```
2955 \newcommand*{\glsentryuseriii}[1]{\csname glo@#1@useriii\endcsname}
```

\Glsentryuseriii

```
2956 \newcommand*{\Glsentryuseriii}[1]{%
2957 \protected@edef\@glo@text{\csname glo@#1@useriii\endcsname}%
2958 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriv Get the fourth user key (as specified by the user4 when the entry was defined).
The argument is the label associated with the entry.

```
2959 \newcommand*{\glsentryuseriv}[1]{\csname glo@#1@useriv\endcsname}
```


\Glsentryuseriv

```
2960 \newcommand*{\Glsentryuseriv}[1]{%
2961 \protected@edef\@glo@text{\csname glo@#1@useriv\endcsname}%
2962 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriv Get the fifth user key (as specified by the user5 when the entry was defined).
The argument is the label associated with the entry.

```
2963 \newcommand*{\glsentryuseriv}[1]{\csname glo@#1@useriv\endcsname}
```

\Glsentryuseriv

```
2964 \newcommand*{\Glsentryuseriv}[1]{%
2965 \protected@edef\@glo@text{\csname glo@#1@useriv\endcsname}%
2966 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuservi Get the sixth user key (as specified by the user6 when the entry was defined).
The argument is the label associated with the entry.

```
2967 \newcommand*{\glsentryuservi}[1]{\csname glo@#1@uservi\endcsname}
```

\Glsentryuservi

```
2968 \newcommand*{\Glsentryuservi}[1]{%
2969 \protected@edef\@glo@text{\csname glo@#1@uservi\endcsname}%
2970 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryshort Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
2971 \newcommand*{\glsentryshort}[1]{\csname glo@#1@short\endcsname}
```

\Glsentryshort

```
2972 \newcommand*{\Glsentryshort}[1]{%
2973 \protected@edef\@glo@text{\csname glo@#1@short\endcsname}%
2974 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryshortpl Get the short plural key (as specified by the shortplural the entry was defined).
The argument is the label associated with the entry.

```
2975 \newcommand*{\glsentryshortpl}[1]{\csname glo@#1@shortpl\endcsname}
```

\Glsentryshortpl

```
2976 \newcommand*{\Glsentryshortpl}[1]{%
2977 \protected@edef\@glo@text{\csname glo@#1@shortpl\endcsname}%
2978 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentrylong Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
2979 \newcommand*{\glsentrylong}[1]{\csname glo@#1@long\endcsname}
```

```

\Glsentrylong
2980 \newcommand*{\Glsentrylong}[1]{%
2981 \protected@edef\@glo@text{\csname glo@#1@long\endcsname}%
2982 \expandafter\makefirstuc\expandafter{\@glo@text}}

\glsentrylongpl  Get the long plural key (as specified by the longplural the entry was defined).
                  The argument is the label associated with the entry.
2983 \newcommand*{\glsentrylongpl}[1]{\csname glo@#1@longpl\endcsname}

\Glsentrylongpl
2984 \newcommand*{\Glsentrylongpl}[1]{%
2985 \protected@edef\@glo@text{\csname glo@#1@longpl\endcsname}%
2986 \expandafter\makefirstuc\expandafter{\@glo@text}}

Short cut macros to access full form:

\glsentryfull
2987 \newcommand*{\glsentryfull}[1]{%
2988 \glsentrylong{#1}\space(\glsentryshort{#1})%
2989 }

\Glsentryfull
2990 \newcommand*{\Glsentryfull}[1]{%
2991 \Glsentrylong{#1}\space(\glsentryshort{#1})%
2992 }

\glsentryfullpl
2993 \newcommand*{\glsentryfullpl}[1]{%
2994 \glsentrylongpl{#1}\space(\glsentryshortpl{#1})%
2995 }

\Glsentryfullpl
2996 \newcommand*{\Glsentryfullpl}[1]{%
2997 \Glsentrylongpl{#1}\space(\glsentryshortpl{#1})%
2998 }

\glsentrynumberlist  Displays the number list as is.
2999 \newcommand*{\glsentrynumberlist}[1]{%
3000 \glsdoifexists{#1}%
3001 {%
3002 \csname glo@#1@numberlist\endcsname
3003 }%
3004 }

\glsdisplaynumberlist  Formats the number list for the given entry label. Doesn't work with hyperref.
3005 \@ifpackageloaded{hyperref}
3006 {%
3007 \newcommand*{\glsdisplaynumberlist}[1]{%

```

```

3008 \GlossariesWarning
3009 {%
3010 \string\glsdisplaynumberlist\space
3011 doesn't work with hyperref.^JUsing
3012 \string\glsentrynumberlist\space instead%
3013 }%
3014 \glsentrynumberlist{#1}%
3015 }%
3016 }%
3017 {%
3018 \newcommand*{\glsdisplaynumberlist}[1]{%
3019 \glsdoifexists{#1}%
3020 {%
3021 \bgroup
3022 \def\@glo@label{#1}%
3023 \let\@org@glsglnumberformat\glsglnumberformat
3024 \def\glsglnumberformat##1{##1}%
3025 \protected@edef\the@numberlist{\csname glo@\@glo@label @numberlist\endcsname}%
3026 \def\@glsgl@numlist@sep{}%
3027 \def\@glsgl@numlist@nextsep{}%
3028 \def\@glsgl@numlist@lastsep{}%
3029 \def\@glsgl@thislist{}%
3030 \def\@glsgl@donext@def{}%
3031 \renewcommand\do[1]{%
3032 \protected@edef\@glsgl@thislist{%
3033 \@glsgl@thislist
3034 \noexpand\@glsgl@numlist@sep
3035 ##1%
3036 }%
3037 \let\@glsgl@numlist@sep\@glsgl@numlist@nextsep
3038 \def\@glsgl@numlist@nextsep{\glsgl@numlist@sep}%
3039 \@glsgl@donext@def
3040 \def\@glsgl@donext@def{%
3041 \def\@glsgl@numlist@lastsep{\glsgl@numlist@lastsep}%
3042 }%
3043 }%
3044 \expandafter \glsgl@numlist@parser \expandafter{\the@numberlist}%
3045 \let\@glsgl@numlist@sep\@glsgl@numlist@lastsep
3046 \@glsgl@thislist
3047 \egroup
3048 }%
3049 }
3050 }

```

\glsgl@numlist@sep

```

3051 \newcommand*{\glsgl@numlist@sep}{, }

```

\glsgl@numlist@lastsep

```

3052 \newcommand*{\glsgl@numlist@lastsep}{ \& }

```

`\glshyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\glslink` or `\glsadd` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```
3053 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
3054 \def\@glo@label{#2}%
3055 \@glslink{\glo@linkprefix#2}{#1}}
```

1.11 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
3056 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
3057 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
3058 \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`

```
3059 \newrobustcmd*{\glsadd}[2][ ]{%
3060   \glsdoifexists{#2}%
3061   {%
3062     \def\@glsnumberformat{glsnumberformat}%
3063     \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
3064     \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
3065   \@gls@saveentrycounter
3066   \@do@wrglossary{#2}%
3067   }%
3068 }
```

`\glsaddall[<option list>]`

Add all terms defined for the listed glossaries (without displaying any text). If types key is omitted, apply to all glossary types.

`\glsaddall`

```
3069 \newrobustcmd*{\glsaddall}[1][ ]{%
3070   \edef\@glo@type{\@glo@types}%
3071   \setkeys{glossadd}{#1}%
```

```

3072 \forallglsentries[\@glo@type]{\@glo@entry}{%
3073 \glsadd[#1]{\@glo@entry}}%
3074 }

```

1.12 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 `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.`. This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```

3075 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```

3076 \edef\glsclosebrace{\expandafter\@gobble\string\} }

```

`\glsquote` Define command that makes it easier to write quote marks to a file in the event that the double quote character has been made active.

```

3077 \edef\glsquote#1{\string"#1\string"}

```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```

3078 \ifglsxindy
3079   \newcommand*{\@glsfirstletter}{A}
3080 \fi

```

`stLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```

3081 \ifglsxindy
3082   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
3083     \renewcommand*{\@glsfirstletter}{#1}}
3084 \else
3085   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
3086     \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
3087 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```

3088 \newcommand*{\@glsminrange}{2}

```

`\GlsSetXdyMinRangeLength` Set the minimum range length. The value must either be none or a positive integer. The glossaries package doesn't check if the argument is valid, that is left to xindy.

```

3089 \ifglxindy
3090   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
3091     \renewcommand*{\@glsminrange}{#1}}
3092 \else
3093   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
3094     \glsnoxywarning\GlsSetXdyMinRangeLength}
3095 \fi

```

`\writeist`

```

3096 \ifglxindy

```

Code to use if xindy is required.

```

3097 \def\writeist{%

```

Update attributes list

```

3098   \@gls@addpredefinedattributes

```

Open the file.

```

3099   \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

3100   \write\glswrite{;; xindy style file created by the glossaries
3101     package}%
3102   \write\glswrite{;; for document '\jobname' on
3103     \the\year-\the\month-\the\day}%

```

Specify the required styles

```

3104   \write\glswrite{^^J; required styles^^J}
3105   \@for\@xdystyle:=\@xdyrequiredstyles\do{%
3106     \ifx\@xdystyle\@empty
3107     \else
3108       \protected@write\glswrite{{(require
3109         \string"\@xdystyle.xdy\string")}}%
3110     \fi
3111   }%

```

List the allowed attributes (possible values used by the format key)

```

3112   \write\glswrite{^^J%
3113     ; list of allowed attributes (number formats)^^J}%
3114   \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```

3115   \write\glswrite{^^J; user defined alphabets^^J}%
3116   \write\glswrite{\@xdyuseralphabets}%

```

Define location classes.

```
3117 \write\glswrite{^^J; location class definitions^^J}%
```

As from version 3.0, locations are now specified as $\{\langle Hprefix \rangle\}\{\langle number \rangle\}$, so need to add all possible combinations of location types.

```
3118 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
3119 \protected@write\glswrite{}\{(define-location-class
3120 \string"\@gls@classI\string"^^J\space\space\space
3121 (
3122 :sep "{"
3123 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
3124 :sep "}"
3125 )
3126 ^^J\space\space\space
3127 :min-range-length \@glsminrange^^J%
3128 )
3129 }%
```

Nested iteration over all classes:

```
3130 {%
3131 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
3132 \protected@write\glswrite{}\{(define-location-class
3133 \string"\@gls@classII-\@gls@classI\string"
3134 ^^J\space\space\space
3135 (
3136 :sep "{"
3137 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
3138 :sep "}"
3139 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
3140 :sep "}"
3141 )
3142 ^^J\space\space\space
3143 :min-range-length \@glsminrange^^J%
3144 )
3145 }%
3146 }%
3147 }%
3148 }%
```

User defined location classes (needs checking for new location format).

```
3149 \write\glswrite{^^J; user defined location classes}%
3150 \write\glswrite{\@xdyuserlocationdefs}%
```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for \backslash glsseeformat which xindy won't recognise.)

```
3151 \write\glswrite{^^J; define cross-reference class^^J}%
3152 \write\glswrite{(define-crossref-class \string"see\string"
3153 :unverified )}%
```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```
3154 \write\glswrite{(markup-crossref-list
3155      :class \string"see\string"^^J\space\space\space
3156      :open \string"\string\glsseeformat\string"
3157      :close \string"{}\string")}%
```

List the order to sort the classes.

```
3158 \write\glswrite{^^J; define the order of the location classes}%
3159 \write\glswrite{(define-location-class-order
3160      (\@xdylocationclassorder))}%
```

Specify what to write to the start and end of the glossary file.

```
3161 \write\glswrite{^^J; define the glossary markup^^J}%
3162 \write\glswrite{(markup-index^^J\space\space\space
3163      :open \string"\string
3164      \glossarysection[\string\glossarytoctitle]{\string
3165      \glossarytitle}\string\glossarypreamble}%
```

Add all the xindy-only macro definitions (needed to prevent errors in the event that the user changes from xindy to makeindex)

```
3166 \@for\@this@ctr:=\@xdycounters\do{%
3167     {%
3168         \@for\@this@attr:=\@xdyattributelist\do{%
3169             \protected@write\glswrite{{}\string\providecommand*%
3170                 \expandafter\string
3171                 \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
3172                 {%
3173                     \string\setentrycounter
3174                     [\expandafter\@gobble\string\#1]{\@this@ctr}%
3175                     \expandafter\string
3176                     \csname\@this@attr\endcsname
3177                     {\expandafter\@gobble\string\#2}%
3178                 }%
3179             }%
3180         }%
3181     }%
3182 }%
```

Add the end part of the open tag and the rest of the markup-index information:

```
3183 \write\glswrite{%
3184     \string\begin
3185     {theglossary}\string\glossaryheader\string~n\string" ^^J\space
3186     \space\space:close \string"\expandafter\@gobble
3187     \string%\string~n\string
3188     \end{theglossary}\string\glossarypostamble
3189     \string~n\string" ^^J\space\space\space
3190     :tree)}}%
```


Specify what to put between letter groups

```
3191 \write\glswrite{(markup-letter-group-list
3192 :sep \string"\string\glsgroupskip\string~n\string")}%
```

Specify what to put between entries

```
3193 \write\glswrite{(markup-indexentry
3194 :open \string"\string\relax \string\glresetentrylist
3195 \string~n\string")}%
```

Specify how to format entries

```
3196 \write\glswrite{(markup-locclass-list :open
3197 \string"\glsoopenbrace\string\glossaryentrynumbers
3198 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
3199 :sep \string", \string"
3200 :close \string"\glsclosebrace\glsclosebrace\string")}%
```

Specify how to separate location numbers

```
3201 \write\glswrite{(markup-locref-list
3202 :sep \string"\string\delimN\space\string")}%
```

Specify how to indicate location ranges

```
3203 \write\glswrite{(markup-range
3204 :sep \string"\string\delimR\space\string")}%
```

Specify 2-page and 3-page suffixes, if defined. First, the values must be sanitized to write them explicitly.

```
3205 \@onelevel@sanitize\gls@suffixF
3206 \@onelevel@sanitize\gls@suffixFF

3207 \ifx\gls@suffixF\@empty
3208 \else
3209 \write\glswrite{(markup-range
3210 :close "\gls@suffixF" :length 1 :ignore-end)}%
3211 \fi
3212 \ifx\gls@suffixFF\@empty
3213 \else
3214 \write\glswrite{(markup-range
3215 :close "\gls@suffixFF" :length 2 :ignore-end)}%
3216 \fi
```

Specify how to format locations.

```
3217 \write\glswrite{^^J; define format to use for locations^^J}%
3218 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
3219 \write\glswrite{^^J; define letter group list format^^J}%
3220 \write\glswrite{(markup-letter-group-list
3221 :sep \string"\string\glsgroupskip\string~n\string")}%
```

Define letter group headings.

```
3222 \write\glswrite{^^J; letter group headings^^J}%
3223 \write\glswrite{(markup-letter-group
```

```

3224      :open-head \string"\string\glsgroupheading
3225      \glsopenbrace\string"^^J\space\space\space
3226      :close-head \string"\glsclosebrace\string"))}%

```

Define additional letter groups.

```

3227      \write\glswrite{^^J; additional letter groups^^J}%
3228      \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

3229      \write\glswrite{^^J; additional sort rules^^J}
3230      \write\glswrite{\@xdysortrules}%

```

Close the style file

```

3231      \closeout\glswrite

```

Suppress any further calls.

```

3232      \let\writeist\relax
3233    }
3234 \else

```

Code to use if makeindex is required.

```

3235      \edef\@gls@actualchar{\string?}
3236      \edef\@gls@encapchar{\string|}
3237      \edef\@gls@levelchar{\string!}
3238      \edef\@gls@quotechar{\string"}
3239      \def\writeist{\relax
3240      \openout\glswrite=\istfilename
3241      \write\glswrite{\expandafter\@gobble\string\% makeindex style file
3242      created by the glossaries package}
3243      \write\glswrite{\expandafter\@gobble\string\% for document
3244      '\jobname' on \the\year-\the\month-\the\day}
3245      \write\glswrite{actual '\@gls@actualchar'}
3246      \write\glswrite{encap '\@gls@encapchar'}
3247      \write\glswrite{level '\@gls@levelchar'}
3248      \write\glswrite{quote '\@gls@quotechar'}
3249      \write\glswrite{keyword \string"\string\glossaryentry\string"}
3250      \write\glswrite{preamble \string"\string\glossarysection[\string
3251      \glossarytoctitle]{\string\glossarytitle}\string
3252      \glossarypreamble\string\n\string\begin{theglossary}\string
3253      \glossaryheader\string\n\string"}
3254      \write\glswrite{postamble \string"\string\%\string\n\string
3255      \end{theglossary}\string\glossarypostamble\string\n
3256      \string"}
3257      \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
3258      \string"}
3259      \write\glswrite{item_0 \string"\string\%\string\n\string"}
3260      \write\glswrite{item_1 \string"\string\%\string\n\string"}
3261      \write\glswrite{item_2 \string"\string\%\string\n\string"}
3262      \write\glswrite{item_01 \string"\string\%\string\n\string"}
3263      \write\glswrite{item_x1
3264      \string"\string\relax \string\glsresetentrylist\string\n

```

```

3265     \string"}
3266 \write\glswrite{item_12 \string"\string%\string\n\string"}
3267 \write\glswrite{item_x2
3268     \string"\string\\relax \string\\glsresetentrylist\string\n
3269     \string"}

3270 \write\glswrite{delim_0 \string"\string\{\string
3271     \\glossaryentrynumbers\string\{\string\\relax \string"}
3272 \write\glswrite{delim_1 \string"\string\{\string
3273     \\glossaryentrynumbers\string\{\string\\relax \string"}
3274 \write\glswrite{delim_2 \string"\string\{\string
3275     \\glossaryentrynumbers\string\{\string\\relax \string"}
3276 \write\glswrite{delim_t \string"\string\}\string\}\string"}
3277 \write\glswrite{delim_n \string"\string\\delimN \string"}
3278 \write\glswrite{delim_r \string"\string\\delimR \string"}
3279 \write\glswrite{headings_flag 1}
3280 \write\glswrite{heading_prefix
3281     \string"\string\\glsgroupheading\string\{\string"}
3282 \write\glswrite{heading_suffix
3283     \string"\string\}\string\\relax
3284     \string\\glsresetentrylist \string"}
3285 \write\glswrite{symhead_positive \string"glssymbols\string"}
3286 \write\glswrite{numhead_positive \string"glslnumbers\string"}
3287 \write\glswrite{page_compositor \string"glscpositor\string"}
3288 \@gls@escbsdq\gls@suffixF
3289 \@gls@escbsdq\gls@suffixFF
3290 \ifx\gls@suffixF\@empty
3291 \else
3292     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
3293 \fi
3294 \ifx\gls@suffixFF\@empty
3295 \else
3296     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
3297 \fi
3298 \closeout\glswrite
3299 \let\writeist\relax
3300 }
3301 \fi

```

The command `\noist` will suppress the creation of the `.ist` file. Obviously you need to use this command before `\writeist` to have any effect.

`\noist`

```

3302 \newcommand{\noist}{%
    Update attributes list
3303     \@gls@addpredefinedattributes
3304     \let\writeist\relax
3305 }

```

`\@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`

```
3306 \newcommand*{\@makeglossary}[1]{%
3307   \ifglossaryexists{#1}%
3308   {%
```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```
3309   \ifglssavewrites
3310     \expandafter\newtoks\csname glo@#1@filetok\endcsname
3311   \else
3312     \expandafter\newwrite\csname glo@#1@file\endcsname
3313     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
3314   \fi
3315   \@gls@renewglossary
3316   \writeist
3317 }%
3318 {%
3319   \PackageError{glossaries}%
3320   {Glossary type ‘#1’ not defined}%
3321   {New glossaries must be defined before using \string\makeglossary}%
3322 }%
3323 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```
3324 \newcommand*{\@glsopenfile}[2]{%
3325   \immediate\openout#1=\jobname.\csname @glo#2@out\endcsname
3326   \PackageInfo{glossaries}{Writing glossary file
3327     \jobname.\csname @glo#2@out\endcsname}%
3328 }
```

`\@nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
3329 \newcommand*{\warn@nomakeglossaries}{%
3330   \GlossariesWarningNoLine{\string\makeglossaries\space
3331     hasn't been used,^^Jthe glossaries will not be updated}%
3332 }
```

`\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`

```
3333 \newcommand*{\makeglossaries}{%
```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```
3334 \protected@write\@auxout{}\string\@istfilename{\istfilename}}%
```

```
3335 \protected@write\@auxout{}\string\@glsorder{\glsorder}}%
```

Iterate through each glossary type and activate it.

```
3336 \@for\@glo@type:=\@glo@types\do{%
```

```
3337 \ifthenelse{\equal{\@glo@type}{}}{\}%
```

```
3338 \makeglossary{\@glo@type}}%
```

```
3339 }%
```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```
3340 \renewcommand*\newglossary[4][]{%
```

```
3341 \PackageError{glossaries}{New glossaries
```

```
3342 must be created before \string\makeglossaries}{You need
```

```
3343 to move \string\makeglossaries\space after all your
```

```
3344 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
3345 \let\@makeglossary\relax
```

```
3346 \let\makeglossary\relax
```

```
3347 \let\makeglossaries\relax
```

Disable all commands that have no effect after `\makeglossaries`

```
3348 \@disable@onlypremakeg
```

Suppress warning about no `\makeglossaries`

```
3349 \let\warn@nomakeglossaries\relax
```

Declare list parser for `\glsdisplaynumberlist`

```
3350 \ifglssavenumberlist
```

```
3351 \edef\@gls@doddeflistparser{\noexpand\DeclareListParser
```

```
3352 {\noexpand\glsnumlistparser}{\delimN}}%
```

```
3353 \@gls@doddeflistparser
```

```
3354 \fi
```

```
3355 }
```

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`

```
3356 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
3357 \AtEndDocument{%
```

```

3358 \warn@nomakeglossaries
3359 \warn@noprintglossary
3360 }

```

1.13 Writing information to associated files

`\glswrite` The write used for style file also used for all other output files if `savewrites=true`.

```
3361 \newwrite\glswrite
```

`\istfile` Deprecated.

```
3362 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if `savewrites=true`.

```

3363 \AtEndDocument{%
3364   \glswritefiles
3365 }

```

`\glswritefiles` Only write the files if `savewrites=true`

```

3366 \ifglssavewrites
3367   \newcommand*{\glswritefiles}{%

```

Iterate through all the glossaries

```
3368   \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```

3369     \ifcsundef{glo@\@glo@type @filetok}%
3370     {%
3371       \def\gls@tmp{}%
3372     }%
3373     {%
3374       \edef\gls@tmp{\expandafter\the
3375         \csname glo@\@glo@type @filetok\endcsname}%
3376     }%
3377     \ifx\gls@tmp\@empty
3378       \ifx\@glo@type\glsdefaulttype
3379         \GlossariesWarningNoLine{Glossary '\@glo@type' has no
3380           entries.^^JRemember to use package option 'nomain' if
3381 you
3382           don't want to^^Juse the main glossary}%
3383       \else
3384         \GlossariesWarningNoLine{Glossary '\@glo@type' has no
3385           entries}%
3386       \fi
3387     \else
3388       \@glsopenfile{\glswrite}{\@glo@type}%
3389       \immediate\write\glswrite{%
3390         \expandafter\the
3391         \csname glo@\@glo@type @filetok\endcsname}%
3392       \immediate\closeout\glswrite

```

```

3393     \fi
3394   }%
3395 }
3396 \else
3397   \let\glswritefiles\relax
3398 \fi

```

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`

```

3399 \renewcommand*{\glossary}[1][\glsdefaulttype]{%
3400   \@glossary[#1]%
3401 }

```

Define internal `\@glossary` to ignore its argument. This gets redefined in `\@makeglossary`. This is defined to just `\index` as memoir changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.)

`\@glossary`

```

3402 \def\@glossary[#1]{\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`

```

3403 \newcommand{\@gls@renewglossary}{%
3404   \gdef\@glossary[##1]{\@bsphack\begin{group}\@wrglossary{##1}}%
3405   \let\@gls@renewglossary\empty
3406 }

```

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`

```

3407 \renewcommand*{\@wrglossary}[2]{%
3408   \ifglssavewrites
3409     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
3410     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
3411       \expandafter{\@gls@tmp^^J}%
3412   \else
3413     \ifcsdef{glo@#1@file}%
3414       {%
3415         \expandafter\protected@write\csname glo@#1@file\endcsname{%
3416           \gls@disablepagerefexpansion}{#2}%

```

```

3417 }%
3418 {%
3419 \GlossariesWarning{No file defined for glossary ‘#1’}%
3420 }%
3421 \fi
3422 \endgroup\@esphack
3423 }

```

\@do@wrglossary

```

3424 \newcommand*{\@do@wrglossary}[1]{%
3425 \ifglindexonlyfirst
3426 \ifglused{#1}{\@do@wrglossary{#1}}%
3427 \else
3428 \@do@wrglossary{#1}%
3429 \fi
3430 }

```

@protected@pagefmts List of page formats to be protected against expansion.

```

3431 \newcommand{\gls@protected@pagefmts}{%
3432 \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
3433 }

```

blepagerefexpansion

```

3434 \newcommand*{\gls@disablepagerefexpansion}{%
3435 \@for\@gls@this:=\gls@protected@pagefmts\do
3436 {%
3437 \expandafter\let\@gls@this\relax
3438 }%
3439 }

```

\gls@alphpage

```

3440 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

\gls@Alphpage

```

3441 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

```

\gls@numberpage

```

3442 \newcommand*{\gls@numberpage}{\number\c@page}

```

\gls@romanpage

```

3443 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

```

\gls@Romanpage

```

3444 \newcommand*{\gls@Romanpage}{\@Roman\c@page}

```

\@do@wrglossary Write the glossary entry in the appropriate format. (Need to set \@gls@numberformat and \@gls@counter prior to use.) The argument is the entry’s label.

```

3445 \newcommand*{\@do@wrglossary}[1]{%
3446 \begingroup

```


First a bit of hackery to prevent premature expansion of \c@page. Store original definitions:

```

3447 \let\orgthe\the
3448 \let\orgnumber\number
3449 \let\orgromannumeral\romannumeral
3450 \let\orgalph\@alph
3451 \let\orgAlph\@Alph
3452 \let\orgRoman\@Roman

```

Redefine:

```

3453 \def\the##1{%
3454   \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
3455 \def\number##1{%
3456   \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
3457 \def\romannumeral##1{%
3458   \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
3459 \def\@Roman##1{%
3460   \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
3461 \def\@alph##1{%
3462   \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
3463 \def\@Alph##1{%
3464   \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Prevent expansion:

```

3465 \gls@disablepagerefexpansion

```

Now store location in \@glslocref:

```

3466 \protected@xdef\@glslocref{\theHglentrycounter}%
3467 \endgroup

```

Escape any special characters

```

3468 \@gls@checkmkidxchars\@glslocref

```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

3469 \expandafter\ifx\theHglentrycounter\theHglentrycounter
3470 \def\@glo@counterprefix{%
3471 \else
3472 \protected@edef\@glsHlocref{\theHglentrycounter}%
3473 \@gls@checkmkidxchars\@glsHlocref
3474 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
3475   {\@glslocref}{\@glsHlocref}%
3476 }%
3477 \@do@gls@getcounterprefix
3478 \fi

```

Determine whether to use xindy or makeindex syntax

```

3479 \ifglxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

3480 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
3481 \def\@glo@range{}%
3482 \expandafter\if\@glo@prefix(\relax
3483 \def\@glo@range{:open-range}%
3484 \else
3485 \expandafter\if\@glo@prefix)\relax
3486 \def\@glo@range{:close-range}%
3487 \fi
3488 \fi

```

Write to the glossary file using xindy syntax.

```

3489 \glossary[\csname glo@#1@type\endcsname]{%
3490 (indexentry :tkey (\csname glo@#1@index\endcsname)

3491 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
3492 :attr \string"\@gls@counter\@glo@suffix\string"
3493 \@glo@range
3494 )
3495 }%
3496 \else

```

Convert the format information into the format required for makeindex

```

3497 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
3498 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

3499 \glossary[\csname glo@#1@type\endcsname]{%
3500 \string\glossaryentry{\csname glo@#1@index\endcsname
3501 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
3502 \fi
3503 }

```

`\ls@getcounterprefix` Get the prefix that needs to be prepended to counter in order to get the hyper counter. (For example, with the standard article class and hyperref, `\theequation` needs to be prefixed with `\section num` to get the equivalent `\theHequation`.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```

3504 \newcommand*\@gls@getcounterprefix[2]{%
3505 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
3506 \ifx\@gls@thisloc\@gls@thisHloc
3507 \def\@glo@counterprefix{}%
3508 \else
3509 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
3510 \def\@glo@tmp{##2}%
3511 \ifx\@glo@tmp\@empty
3512 \def\@glo@counterprefix{}%
3513 \else
3514 \def\@glo@counterprefix{##1}%
3515 \fi
3516 }%

```

```

3517 \@gls@get@counterprefix#2.#1\end@getprefix
3518 \fi
3519 }

```

1.14 Glossary Entry Cross-References

`\@do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry's label, the second must be in the form `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such as “see” and `\langle list \rangle` is a list of labels.

```

3520 \newcommand{\@do@seeglossary}[2]{%
3521 \def\@gls@xref{#2}%
3522 \@onelevel@sanitize\@gls@xref
3523 \@gls@checkmkidxchars\@gls@xref
3524 \ifglxsindy
3525 \glossary[\csname glo@#1@type\endcsname]{%
3526 (indexentry
3527 :tkey (\csname glo@#1@index\endcsname)
3528 :xref (\string"\@gls@xref\string")
3529 :attr \string"see\string"
3530 )
3531 }%
3532 \else
3533 \glossary[\csname glo@#1@type\endcsname]{%
3534 \string\glossaryentry{\csname glo@#1@index\endcsname
3535 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
3536 \fi
3537 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

3538 \def\@gls@fixbraces#1#2#3\@nil{%
3539 \ifx#2[\relax
3540 \def#1{#2#3}%
3541 \else
3542 \def#1{{#2#3}}%
3543 \fi
3544 }

```

`\glssee` `\glssee{\langle label \rangle}{\langle cross-ref list \rangle}`

```

3545 \newcommand*\glssee[3][\seename]{%
3546 \@do@seeglossary{#2}{#1}{#3}}
3547 \newcommand*\@glssee[3][\seename]{%
3548 \glssee[#1]{#3}{#2}}

```

`\glsseeformat` The first argument specifies what tag to use (e.g. “see”), the second argument is a comma-separated list of labels. The final argument (the location) is ignored.

```

3549 \newcommand*\glsseeformat[3][\seename]{\emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{\langle list \rangle}` formats list of entry labels.

```

3550 \newcommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```
3551 \let\@gls@dolast\relax
```

Don't display separator on the first iteration of the loop

```
3552 \let\@gls@donext\relax
```

Iterate through the labels

```
3553 \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
3554 \ifx\@xfor@nextelement\@nnil
```

```
3555 \@gls@dolast
```

```
3556 \else
```

```
3557 \@gls@donext
```

```
3558 \fi
```

display the entry for this label

```
3559 \glsseeitem{\@gls@thislabel}%
```

Update separators

```
3560 \let\@gls@dolast\glsseelastsep
```

```
3561 \let\@gls@donext\glsseesep
```

```
3562 }%
```

```
3563 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
3564 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
3565 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
3566 \newcommand*{\glsseeitem}[1]{\glshyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glsentrytext` instead of `\glsentryname`. (To avoid problems with the name key being sanitized.)

```
3567 \newcommand*{\glsseeitemformat}[1]{\glsentrytext{#1}}
```

1.15 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.

`\gls@save@numberlist` Provide command to store number list.

```
3568 \newcommand*{\gls@save@numberlist}[1]{%
3569   \ifglssavenumberlist
3570     \toks@{#1}%
3571     \edef\@do@writeaux@info{%
3572       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
3573     }%
3574     \@onelevel@sanitize\@do@writeaux@info
3575     \protected@write\@auxout{}\{\@do@writeaux@info}%
3576   \fi
3577 }
```

`\warn@noprintglossary` Warn the user if they have forgotten `\printglossaries` or `\printglossary`.
(Will be suppressed if there is at least one occurrence of `\printglossary`.
There is no check to ensure that there is a `\printglossary` for each defined
glossary.)

```
3578 \def\warn@noprintglossary{%
3579   \GlossariesWarningNoLine{No \string\printglossary\space
3580     or \string\printglossaries\space
3581     found.^^JThis document will not have a glossary}%
3582 }
```

`\printglossary` The TOC title needs to be processed in a different manner to the main title in
case the translator and hyperref packages are both being used.

```
3583 \ifcsundef{printglossary}{}%
3584 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
3585   \GlossariesWarning{Overriding \string\printglossary}%
3586   \undef\printglossary
3587 }
```

`\printglossary` has an optional argument. The default value is to set the glossary
type to the main glossary.

```
3588 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
```

Set up defaults.

```
3589   \def\@glo@type{\glsdefaulttype}%
3590   \def\glossarytitle{\csname @glo@type @title\endcsname}%

3591   \def\glossarytoctitle{\glossarytitle}%
3592   \let\org@glossarytitle\glossarytitle
3593   \def\@glossarystyle{}%
3594   \def\gls@dotoc@title{\glssettoctitle{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the
optional argument)

```
3595   \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
3596   \bgroup
```

Determine settings specified in the optional argument.

```
3597 \setkeys{printgloss}{#1}%
```

If title has been set, but toctitle hasn't, make toctitle the same as given title (rather than the title used when the glossary was defined)

```
3598 \ifx\glossarytitle\org@glossarytitle
```

```
3599 \else
```

```
3600 \expandafter\let\csname @glotype@\@glo@type @title\endcsname
```

```
3601 \glossarytitle
```

```
3602 \fi
```

Allow a high-level user command to indicate the current glossary

```
3603 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
3604 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

```
3605 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
3606 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
3607 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
3608 \gls@dotocitle
```

Set the glossary style

```
3609 \@glossarystyle
```

added a way to fetch the current entry label:

```
3610 \let\gls@org@glossaryentryfield\glossaryentryfield
```

```
3611 \let\gls@org@glossarysubentryfield\glossarysubentryfield
```

```
3612 \renewcommand{\glossaryentryfield}[1]{%
```

```
3613 \gdef\glscurrententrylabel{##1}%
```

```
3614 \gls@org@glossaryentryfield{##1}%
```

```
3615 }%
```

```
3616 \renewcommand{\glossarysubentryfield}[2]{%
```

```
3617 \gdef\glscurrententrylabel{##2}%
```

```
3618 \gls@org@glossarysubentryfield{##1}{##2}%
```

```
3619 }%
```

Some macros may end up being expanded into internals in the glossary, so need to make @ a letter.

```
3620 \makeatletter
```

Input the glossary file, if it exists.

```
3621 \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
3622 \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
```

```

3623 {}%
3624 {\null}}%

  If xindy is being used, need to write the language dependent information to
  the .aux file for makeglossaries.

3625 \ifglxindy
3626 \ifcsundef{@xdy@\@glo@type @language}%
3627 {%
3628 \edef\@do@auxoutstuff{%
3629 \noexpand\AtEndDocument{%
3630 \noexpand\immediate\noexpand\write\@auxout{%
3631 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
3632 }%
3633 }%
3634 }%
3635 {%
3636 \edef\@do@auxoutstuff{%
3637 \noexpand\AtEndDocument{%
3638 \noexpand\immediate\noexpand\write\@auxout{%
3639 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
3640 @language\endcsname}}%
3641 }%
3642 }%
3643 }%
3644 \@do@auxoutstuff
3645 \edef\@do@auxoutstuff{%
3646 \noexpand\AtEndDocument{%
3647 \noexpand\immediate\noexpand\write\@auxout{%
3648 \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
3649 }%
3650 }%
3651 \@do@auxoutstuff
3652 \fi
3653 \egroup

  Reset \glossaryentrynumbers
3654 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

  Suppress warning about no \printglossary
3655 \global\let\warn@noprintglossary\relax
3656 }

```

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`

```
3657 \newcommand*{\printglossaries}{%
3658   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}}%
3659 }
```

The keys that can be used in the optional argument to `\printglossary` are as follows: The `type` key sets the glossary type.

```
3660 \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`.

```
3661 \define@key{printgloss}{title}{%
3662   \def\glossarytitle{#1}%
3663   \let\gls@dotoc@title\relax
3664 }
```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```
3665 \define@key{printgloss}{toctitle}{%
3666   \def\glossarytoctitle{#1}%
3667   \let\gls@dotoc@title\relax
3668 }
```

The `style` key sets the glossary style (but only for the given glossary).

```
3669 \define@key{printgloss}{style}{%
3670   \ifcsundef{\glsstyle@#1}%
3671   {%
3672     \PackageError{glossaries}%
3673       {Glossary style ‘#1’ undefined}{}%
3674   }%
3675   {%
3676     \def\@glossarystyle{\csname @glsstyle@#1\endcsname}%
3677   }%
3678 }
```

The `numberedsection` key determines if this glossary should be in a numbered section.

```
3679 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
3680   false,nolabel,autolabel}[nolabel]{%
3681     \ifcase\nr\relax
3682       \renewcommand*{\@glossarysecstar}{*}%
3683       \renewcommand*{\@glossaryseclabel}{}%
3684     \or
3685       \renewcommand*{\@glossarysecstar}{}%
3686       \renewcommand*{\@glossaryseclabel}{}%
3687     \or
3688       \renewcommand*{\@glossarysecstar}{*}%
3689       \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
3690   \fi}
```

The `nonumberlist` key determines if this glossary should have a number list.

```
3691 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
```



```

3692 \ifglsonumberlist
3693   \def\glossaryentrynumbers##1{%
3694 \else
3695   \def\glossaryentrynumbers##1{##1}%
3696 \fi}

```

`\@glsonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsonextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

3697 \newcommand*\@glsonextpages{%
3698   \gdef\glossaryentrynumbers##1{%
3699     \glsresetentrylist
3700   }%
3701 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

3702 \newcommand*\@glsnextpages{%
3703   \gdef\glossaryentrynumbers##1{%
3704     ##1\glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

3705 \newcommand*\glsresetentrylist{%
3706   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsonextpages` Outside of `\printglossary` this does nothing.

```

3707 \newcommand*\glsonextpages{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

3708 \newcommand*\glsnextpages{}

```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

3709 \ifglsentrycounter
3710   \ifx\@gls@counterwithin\@empty
3711     \newcounter{glossaryentry}
3712   \else
3713     \newcounter{glossaryentry}[\@gls@counterwithin]
3714   \fi
3715   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
3716 \fi

```

glossarysubentry If the subentrycounter package option has been used, define a counter to number each level 1 entry.

```

3717 \ifglssubentrycounter
3718   \ifglsentrycounter
3719     \newcounter{glossarysubentry}[glossaryentry]
3720   \else
3721     \newcounter{glossarysubentry}
3722   \fi
3723   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
3724 \fi

```

resetsubentrycounter Resets the glossarysubentry counter.

```

3725 \ifglssubentrycounter
3726   \newcommand*{\glsresetsubentrycounter}{%
3727     \setcounter{glossarysubentry}{0}%
3728   }
3729 \else
3730   \newcommand*{\glsresetsubentrycounter}{}
3731 \fi

```

resetsubentrycounter Resets the glossareentry counter.

```

3732 \ifglsentrycounter
3733   \newcommand*{\glsresetentrycounter}{%
3734     \setcounter{glossaryentry}{0}%
3735   }
3736 \else
3737   \newcommand*{\glsresetentrycounter}{}
3738 \fi

```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```

3739 \ifglsentrycounter
3740   \newcommand*{\glsstepentry}[1]{%
3741     \refstepcounter{glossaryentry}%
3742     \label{glentry-#1}%
3743   }
3744 \else
3745   \newcommand*{\glsstepentry}[1]{}
3746 \fi

```

\glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

3747 \ifglssubentrycounter
3748   \newcommand*{\glsstepsubentry}[1]{%
3749     \def\currentglssubentry{#1}%
3750     \refstepcounter{glossarysubentry}%
3751     \label{glentry-#1}%
3752   }

```

```

3753 \else
3754   \newcommand*{\glsstepsubentry}[1]{
3755 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

3756 \ifglentrycounter
3757   \newcommand*{\glsrefentry}[1]{\ref{glentry-#1}}
3758 \else
3759   \ifglssubentrycounter
3760     \newcommand*{\glsrefentry}[1]{\ref{glentry-#1}}
3761   \else
3762     \newcommand*{\glsrefentry}[1]{\gls{#1}}
3763   \fi
3764 \fi

```

`glentrycounterlabel` Defines how to display the glossaryentry counter.

```

3765 \ifglentrycounter
3766   \newcommand*{\glentrycounterlabel}{\theglossaryentry.\space}
3767 \else
3768   \newcommand*{\glentrycounterlabel}{}
3769 \fi

```

`glsubentrycounterlabel` Defines how to display the glossarysubentry counter.

```

3770 \ifglssubentrycounter
3771   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
3772 \else
3773   \newcommand*{\glssubentrycounterlabel}{}
3774 \fi

```

`\glentryitem` Step and display glossaryentry counter, if appropriate.

```

3775 \ifglentrycounter
3776   \newcommand*{\glentryitem}[1]{%
3777     \glsstepentry{#1}\glentrycounterlabel
3778   }
3779 \else
3780   \newcommand*{\glentryitem}[1]{\glsresetsubentrycounter}
3781 \fi

```

`\glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

3782 \ifglssubentrycounter
3783   \newcommand*{\glssubentryitem}[1]{%
3784     \glsstepsubentry{#1}\glssubentrycounterlabel
3785   }
3786 \else
3787   \newcommand*{\glssubentryitem}[1]{}
3788 \fi

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

3789 \ifcsundef{theglossary}%
3790 {%
3791   \newenvironment{theglossary}{}{}%
3792 }%
3793 {%
3794   \GlossariesWarning{overriding ‘theglossary’ environment}%
3795   \renewenvironment{theglossary}{}{}%
3796 }

```

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`

```

3797 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

3798 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#1}{#2}}

```

`\glossaryentryfield` `\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>`.

```

3799 \newcommand*{\glossaryentryfield}[5]{%
3800 \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

`\glossaryentryfield` `\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore `<symbol>`. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```

3801 \newcommand*{\glossarysubentryfield}[6]{%
3802 \glstarget{#2}{\strut}#4. #6\par}

```

Within each glossary, the entries form distinct groups which are determined by the first character of the sort key. When using `makeindex`, there will be a maximum of 28 groups: symbols, numbers, and the 26 alphabetical groups A, ..., Z. If you use `xindy` the groups will depend on whatever alphabet is used. This is determined by the language or custom alphabets can be created

in the xindy style file. 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`

```
3803 \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 redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
3804 \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 `\glsgrouptitle` and `\glsgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgrouptitle{<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.

`\glsgrouptitle`

```
3805 \newcommand*{\glsgrouptitle}[1]{%
3806   \ifcsundef{#1groupname}{#1}{\csname #1groupname\endcsname}%
3807 }

```

`\glsgrouplabel{<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 `\glsgrouptitle`, you will also need to redefine `\glsgrouplabel`.

`\glsgetgrouplabel`

```
3808 \newcommand*{\glsgetgrouplabel}[1]{%
3809 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
3810 \ifthenelse{\equal{#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`

```
3811 \newcommand*{\setentrycounter}[2][{}]{%
3812   \def\@glo@counterprefix{#1}%
3813   \ifx\@glo@counterprefix\@empty
3814     \def\@glo@counterprefix{.}%
3815   \else
3816     \def\@glo@counterprefix{.#1.}%
3817   \fi
3818   \def\glsentrycounter{#2}%
3819 }
```

The current glossary style can be set using `\glossarystyle{<style>}`.

`\glossarystyle`

```
3820 \newcommand*{\glossarystyle}[1]{%
3821   \ifcsundef{@glsstyle@#1}%
3822   {%
3823     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
3824   }%
3825   {%
3826     \csname @glsstyle@#1\endcsname
3827   }%
3828 }
```

`\newglossarystyle` New glossary styles can be defined using:

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

The `<definition>` argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [subsection 1.18](#) 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.

```
3829 \newcommand{\newglossarystyle}[2]{%
3830   \ifcsundef{@glsstyle@#1}%
3831   {%
3832     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
3833   }%
3834   {%
3835     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
3836   }
```

```

3836 }%
3837 }

```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```

3838 \newcommand{\renewglossarystyle}[2]{%
3839   \ifcsundef{@glsstyle@#1}%
3840   {%
3841     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
3842   }%
3843   {%
3844     \csdef{@glsstyle@#1}{#2}%
3845   }%
3846 }

```

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`

```

3847 \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 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 package.

`\glshypernumber`

```

3848 \ifcsundef{hyperlink}%
3849 {%
3850   \def\glshypernumber#1{#1}%
3851 }%
3852 {%
3853   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}\@nil}
3854 }

```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

3855 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
3856   \ifx\#1\%
3857   \else
3858     \@delimR#1\delimR\delimR\%
3859   \fi
3860   \ifx\#2\%
3861   \else
3862     #2%
3863   \fi
3864   \ifx\#3\%
3865   \else
3866     \@glshypernumber#3\@nil
3867   \fi
3868 }

```

`\@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`

```

3869 \def\@delimR#1\delimR #2\delimR #3\{%
3870 \ifx\#2\%
3871   \@delimN{#1}%
3872 \else
3873   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
3874 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

3875 \def\@delimN#1{\@delimN#1\delimN \delimN\}
3876 \def\@delimN#1\delimN #2\delimN#3\{%
3877 \ifx\#3\%
3878   \@gls@numberlink{#1}%
3879 \else
3880   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
3881 \fi
3882 }

```

The following code is modified from hyperref's `\HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

3883 \def\@gls@numberlink#1{%
3884 \begingroup
3885 \toks@={}%
3886 \@gls@removespaces#1 \@nil
3887 \endgroup}

3888 \def\@gls@removespaces#1 #2\@nil{%
3889 \toks@=\expandafter{\the\toks@#1}%

```



```

3890 \ifx\|#2\%
3891   \edef\x{\the\toks@}%
3892   \ifx\x\empty
3893   \else

3894     \hyperlink{\glentrycounter\@glo@counterprefix\the\toks@}%
3895             {\the\toks@}%
3896   \fi
3897 \else
3898   \@gls@ReturnAfterFi{%
3899     \@gls@removespaces#2\@nil
3900   }%
3901 \fi
3902 }
3903 \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`

```
3904 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}
```

`\hypersf`

```
3905 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}
```

`\hypertt`

```
3906 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}
```

`\hyperbf`

```
3907 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}
```

`\hypermd`

```
3908 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}
```

`\hyperit`

```
3909 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
3910 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
3911 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
3912 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
3913 \newcommand*\hyperemph[1]{\emph{\glshypernumber{#1}}}
```

1.16 Acronyms

If the acronym package option is used, a new glossary called acronym is created

```
3914 \ifglsacronym
3915   \newglossary[alg]{acronym}{acr}{acn}{\acronymname}
      and \acronymtype is set to the name of this new glossary.
3916   \renewcommand*{\acronymtype}{acronym}
3917 \fi
```

```
\oldacronym \oldacronym[\langle label \rangle]{\langle abbrev \rangle}{\langle long \rangle}{\langle key-val list \rangle}
```

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[\langle key-val list \rangle]{\langle label \rangle}{\langle abbrev \rangle}{\langle long \rangle}` and it additionally defines the command `\langle label \rangle` which is equivalent to `\gls{\langle label \rangle}` (thus `\langle label \rangle` must only contain alphabetical characters). If `\langle label \rangle` is omitted, `\langle abbrev \rangle` is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\langle label \rangle` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\langle label \rangle[\langle insert \rangle]` but you can't do `\langle label \rangle[\langle key-val list \rangle]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}['s]`.

Note that it is up to the user to load if desired.

```
3918 \newcommand{\oldacronym}[4][\gls@label]{%
3919   \def\gls@label{#2}%
3920   \newacronym[#4]{#1}{#2}{#3}%
3921   \ifcsundef{xspace}%
3922   {%
3923     \expandafter\edef\csname#1\endcsname{%
3924       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
3925   }%
3926 }%
3927 {%
3928   \expandafter\edef\csname#1\endcsname{%
3929     \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
3930       \noexpand\gls{#1}\noexpand\xspace}%
3931   }%
3932 }%
3933 }
```

```
\newacronym[\langle key-val list \rangle]{\langle label \rangle}{\langle abbrev \rangle}{\langle long \rangle}
```

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.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

`\newacronym`

```
3934 \newcommand{\newacronym}[4] [] {}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix`

Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the smallcaps option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, ABCS looks as though the "s" is part of the acronym, but ABCS looks as though the "s" is a plural suffix. Since the entire text abcs is set in `\textsc`, `\textup` is needed to cancel it out.

```
3935 \newcommand*{\acrpluralsuffix}{\glspluralsuffix}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
3936 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
3937 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
3938 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
3939 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
3940 \newrobustcmd*{\acrfull}{%
```

```
3941 \ifstar\s@acrfull\ns@acrfull
```

```
3942 }
```

```
3943 \newcommand*\s@acrfull[2] [] {%
```

```
3944 \new@ifnextchar[{\@acrfull{hyper=false,#1}{#2}}%
```

```
3945 \@acrfull{hyper=false,#1}{#2} [] }%
```

```
3946 }
```

```
3947 \newcommand*\ns@acrfull[2] [] {%
```

```
3948 \new@ifnextchar[{\@acrfull{#1}{#2}}%
```

```
3949 \@acrfull{#1}{#2} [] }%
```

```
3950 }
```

Low-level macro:

```
3951 \def\@acrfull#1#2[#3]{%
3952   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
3953 }
```

```
\acrlinkfullformat  Format for full links like \acrfull.  Syntax: \acrlinkfullformat{<long
                    cs>}{<short cs>}{<options>}{<label>}{<insert>}
3954 \newcommand{\acrlinkfullformat}[5]{%
3955   \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%
3956 }
```

\acrfullformat Default full form is <long> (<short>).

```
3957 \newcommand{\acrfullformat}[2]{#1\space(#2)}
```

Default format for full acronym

\Acrfull

```
3958 \newrobustcmd*\Acrfull{%
3959   \@ifstar\s@Acrfull\ns@Acrfull
3960 }

3961 \newcommand*\s@Acrfull[2][]{%
3962   \new@ifnextchar[{\@Acrfull{hyper=false,#1}{#2}}%
3963     {\@Acrfull{hyper=false,#1}{#2}[]}%
3964 }
3965 \newcommand*\ns@Acrfull[2][]{%
3966   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
3967     {\@Acrfull{#1}{#2}[]}%
3968 }
```

Low-level macro:

```
3969 \def\@Acrfull#1#2[#3]{%
3970   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
3971 }
```

\ACRfull

```
3972 \newrobustcmd*\ACRfull{%
3973   \@ifstar\s@ACRfull\ns@ACRfull
3974 }

3975 \newcommand*\s@ACRfull[2][]{%
3976   \new@ifnextchar[{\@ACRfull{hyper=false,#1}{#2}}%
3977     {\@ACRfull{hyper=false,#1}{#2}[]}%
3978 }
3979 \newcommand*\ns@ACRfull[2][]{%
3980   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
3981     {\@ACRfull{#1}{#2}[]}%
3982 }
```

Low-level macro:

```
3983 \def\@ACRfull#1#2[#3]{%
3984   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
3985 }
```

Plural:

\acrfullpl

```
3986 \newrobustcmd*\acrfullpl{%
3987   \@ifstar\s@acrfullpl\ns@acrfullpl
3988 }

3989 \newcommand*\s@acrfullpl[2][{}]{%
3990   \new@ifnextchar[{\@acrfullpl{hyper=false,#1}{#2}}%
3991     {\@acrfullpl{hyper=false,#1}{#2}[]}%
3992 }
3993 \newcommand*\ns@acrfullpl[2][{}]{%
3994   \new@ifnextchar[{\@acrfullpl{#1}{#2}}%
3995     {\@acrfullpl{#1}{#2}[]}%
3996 }
```

Low-level macro:

```
3997 \def\@acrfullpl#1#2[#3]{%
3998   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
3999 }
```

\Acrfullpl

```
4000 \newrobustcmd*\Acrfullpl{%
4001   \@ifstar\s@Acrfullpl\ns@Acrfullpl
4002 }

4003 \newcommand*\s@Acrfullpl[2][{}]{%
4004   \new@ifnextchar[{\@Acrfullpl{hyper=false,#1}{#2}}%
4005     {\@Acrfullpl{hyper=false,#1}{#2}[]}%
4006 }
4007 \newcommand*\ns@Acrfullpl[2][{}]{%
4008   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%
4009     {\@Acrfullpl{#1}{#2}[]}%
4010 }
```

Low-level macro:

```
4011 \def\@Acrfullpl#1#2[#3]{%
4012   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
4013 }
```

\ACRfullpl

```
4014 \newrobustcmd*\ACRfullpl{%
4015   \@ifstar\s@ACRfullpl\ns@ACRfullpl
4016 }
```

```

4017 \newcommand*\s@ACRfullpl[2] [] {%
4018   \new@ifnextchar[{\@ACRfullpl{hyper=false,#1}{#2}}{%
4019     {\@ACRfullpl{hyper=false,#1}{#2} []}%
4020 }
4021 \newcommand*\ns@ACRfullpl[2] [] {%
4022   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}{%
4023     {\@ACRfullpl{#1}{#2} []}%
4024 }

Low-level macro:
4025 \def\@ACRfullpl#1#2[#3] {%
4026   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
4027 }

```

1.17 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
4028 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
4029 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

`\acrnameformat` The styles that allow an additional description use `\acrnameformat{<short>}{<long>}` to determine what information is displayed in the name.

```
4030 \newcommand*\acrnameformat[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
4031 \newtoks\glskeylisttok
```

`\glslabeltok`

```
4032 \newtoks\glslabeltok
```

`\glsshorttok`

```
4033 \newtoks\glsshorttok
```

`\glslongtok`

```
4034 \newtoks\glslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
4035 \newcommand*\newacronymhook{}
```

`AcronymDisplayStyle` Sets the default acronym display style for given glossary.

```

4036 \newcommand*\SetDefaultAcronymDisplayStyle[1] {%
4037   \defglsdisplay[#1]{##1##4}%
4038   \defglsdisplayfirst[#1]{##1##4}%
4039 }

```

`DefaultNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```

4040 \newcommand*\DefaultNewAcronymDef{%
4041   \edef\@do@newglossaryentry{%
4042     \noexpand\newglossaryentry{\the\glslabeltok}%
4043     {%
4044       type=\acronymtype,%
4045       name={\the\glsshorttok},%
4046       sort={\the\glsshorttok},%
4047       text={\the\glsshorttok},%
4048       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
4049       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4050       firstplural={\acrfullformat{\noexpand\@glo@longpl}%
4051                     {\noexpand\@glo@shortpl}},%
4052       short={\the\glsshorttok},%
4053       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4054       long={\the\glslongtok},%
4055       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4056       description={\the\glslongtok},%
4057       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%

```

Remaining options specified by the user:

```

4058     \the\glskeylisttok
4059   }%
4060 }%
4061 \@do@newglossaryentry
4062 }

```

`DefaultAcronymStyle` Set up the default acronym style:

```

4063 \newcommand*\SetDefaultAcronymStyle{%
  Set the display style:
4064   \@for\@gls@type:=\@glsacronymlists\do{%
4065     \SetDefaultAcronymDisplayStyle{\@gls@type}%
4066   }%

```

Set up the definition of `\newacronym`:

```

4067 \renewcommand{\newacronym}[4][\]{%

```

If user is just using the main glossary and hasn't identified it as a list of acronyms, then update. (This is done to ensure backwards compatibility with versions prior to 2.04).

```

4068   \ifx\@glsacronymlists\@empty
4069     \def\@glo@type{\acronymtype}%
4070     \setkeys{glossentry}{##1}%
4071     \DeclareAcronymList{\@glo@type}%
4072     \SetDefaultAcronymDisplayStyle{\@glo@type}%
4073   \fi
4074   \glskeylisttok{##1}%

```

```

4075 \glslabeltok{##2}%
4076 \glsshorttok{##3}%
4077 \glslongtok{##4}%
4078 \newacronymhook
4079 \DefaultNewAcronymDef
4080 }%
4081 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
4082 }

```

`\acrfootnote` Used by the footnote acronym styles.

```

4083 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}

```

`\acrlinkfootnote`

```

4084 \newcommand*{\acrlinkfootnote}[3]{%
4085 \footnote{\glslink{#1}{#2}{#3}}%
4086 }

```

`\acrnoflinkfootnote`

```

4087 \newcommand*{\acrnoflinkfootnote}[3]{%
4088 \footnote{#3}%
4089 }

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

4090 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
4091 \defglsdisplayfirst[#1]{%
4092 \firstacronymfont{##1}##4%
4093 \expandafter\protect\expandafter\acrfootnote\expandafter
4094 {\@gls@link@opts}{\@gls@link@label}{##3}%
4095 }%
4096 \defglsdisplay[#1]{\acronymfont{##1}##4}%
4097 }

```

`otnoteNewAcronymDef`

```

4098 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
4099 \edef\@do@newglossaryentry{%
4100 \noexpand\newglossaryentry{\the\glslabeltok}%
4101 {%
4102 type=\acronymtype,%
4103 name={\noexpand\acronymfont{\the\glsshorttok}},%
4104 sort={\the\glsshorttok},%
4105 text={\the\glsshorttok},%
4106 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4107 short={\the\glsshorttok},%
4108 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4109 long={\the\glslongtok},%
4110 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4111 symbol={\the\glslongtok},%

```



```

4112         symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4113         \the\glskeylisttok
4114     }%
4115 }%
4116 \do@newglossaryentry
4117 }

```

footnoteAcronymStyle If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

4118 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
4119   \renewcommand{\newacronym}[4][\]{%
4120     \ifx\@glsacronymlists\@empty
4121       \def\@glo@type{\acronymtype}%
4122       \setkeys{glossentry}{##1}%
4123       \DeclareAcronymList{\@glo@type}%
4124       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
4125     \fi
4126     \glskeylisttok{##1}%
4127     \glslabeltok{##2}%
4128     \glsshorttok{##3}%
4129     \glslongtok{##4}%
4130     \newacronymhook
4131     \DescriptionFootnoteNewAcronymDef
4132   }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

4133   \@for\@gls@type:=\@glsacronymlists\do{%
4134     \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
4135   }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

4136   \ifglsacrsmallcaps
4137     \renewcommand*\acronymfont[1]{\textsc{##1}}%
4138     \renewcommand*\acrpluralsuffix{%
4139       \textup{\glspluralsuffix}}%
4140   \else
4141     \ifglsacrsmaller
4142       \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
4143     \fi
4144   \fi

```

Check for package option clash

```

4145   \ifglsacrdua
4146     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
4147     can’t both be set}{}%

```

```

4148 \fi
4149 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

4150 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
4151 \defglstdisplay[#1]{##1##4}%
4152 \defglstdisplayfirst[#1]{##1##4}%
4153 }

```

ionDUANewAcronymDef

```

4154 \newcommand*{\DescriptionDUANewAcronymDef}{%
4155 \edef\@do@newglossaryentry{%
4156 \noexpand\newglossaryentry{\the\glslabeltok}%
4157 {%
4158 type=\acronymtype,%
4159 name={\the\glslongtok},%
4160 sort={\the\glslongtok},%
4161 text={\the\glslongtok},%
4162 plural={\the\glslongtok\noexpand\acrpluralsuffix},%
4163 short={\the\glsshorttok},%
4164 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4165 long={\the\glslongtok},%
4166 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4167 symbol={\the\glsshorttok},%
4168 symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4169 \the\glskeylisttok
4170 }%
4171 }%
4172 \@do@newglossaryentry
4173 }

```

tionDUAAcronymStyle Description, don't use acronym and no footnote. Note that the short form is stored in the symbol key, so if the short form needs to be displayed in the glossary, use a style the displays the symbol.

```

4174 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
4175 \ifglssacrsmallcaps
4176 \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
4177 can't both be set}{}%
4178 \else
4179 \ifglssacrsmaller
4180 \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
4181 can't both be set}{}%
4182 \fi
4183 \fi
4184 \renewcommand{\newacronym}[4][[]]{%
4185 \ifx\@glsacronymlists\@empty
4186 \def\@glo@type{\acronymtype}%
4187 \setkeys{glossentry}{##1}%

```

```

4188     \DeclareAcronymList{\@glo@type}%
4189     \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
4190     \fi
4191     \glskeylisttok{##1}%
4192     \glslabeltok{##2}%
4193     \glsshorttok{##3}%
4194     \glslongtok{##4}%
4195     \newacronymhook
4196     \DescriptionDUANewAcronymDef
4197 }%

Set display.
4198 \@for\@gls@type:=\@glsacronymlists\do{%
4199     \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
4200 }%
4201 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

4202 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
4203     \defglsdisplayfirst[#1]{%
4204         ##1##4\glsdoparenifnotempty{##3}{\protect\firstacronymfont}}%
4205     \defglsdisplay[#1]{\acronymfont{##1}##4}%
4206 }

```

DescriptionNewAcronymDef

```

4207 \newcommand*{\DescriptionNewAcronymDef}{%
4208     \edef\@do@newglossaryentry{%
4209         \noexpand\newglossaryentry{\the\glslabeltok}%
4210         {%
4211             type=\acronymtype,%
4212             name={\noexpand
4213                 \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
4214             sort={\the\glsshorttok},%
4215             first={\the\glslongtok},%
4216             firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4217             text={\the\glsshorttok},%
4218             plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4219             short={\the\glsshorttok},%
4220             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4221             long={\the\glslongtok},%
4222             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4223             symbol={\noexpand\@glo@text},%
4224             symbolplural={\noexpand\@glo@plural},%
4225             \the\glskeylisttok}%
4226     }%
4227     \@do@newglossaryentry
4228 }

```

`riptionAcronymStyle` Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using `\acrnameformat` to allow the user to override the way the name is displayed in the list of acronyms.

```

4229 \newcommand*{\SetDescriptionAcronymStyle}{%
4230   \renewcommand{\newacronym}[4][\]{%
4231     \ifx\@glsacronymlists\@empty
4232       \def\@glo@type{\acronymtype}%
4233       \setkeys{glossentry}{##1}%
4234       \DeclareAcronymList{\@glo@type}%
4235       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
4236     \fi
4237     \glskeylisttok{##1}%
4238     \glslabeltok{##2}%
4239     \glsshorttok{##3}%
4240     \gslongtok{##4}%
4241     \newacronymhook
4242     \DescriptionNewAcronymDef
4243   }%

```

Set display.

```

4244 \@for\@gls@type:=\@glsacronymlists\do{%
4245   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
4246 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

4247 \ifglsacrsmallcaps
4248   \renewcommand{\acronymfont}[1]{\textsc{##1}}
4249   \renewcommand*{\acrpluralsuffix}{%
4250     \textup{\glspluralsuffix}}%
4251 \else
4252   \ifglsacrsmaller
4253     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
4254   \fi
4255 \fi
4256 }%

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

4257 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
4258   \defglsdisplayfirst[#1]{%
4259     \firstacronymfont{##1}##4%
4260     \expandafter\protect\expandafter\acrfootnote\expandafter
4261       {\@gls@link@opts}{\@gls@link@label}{##2}%
4262   }%
4263   \defglsdisplay[#1]{\acronymfont{##1}##4}%
4264 }

```

otnoteNewAcronymDef

```

4265 \newcommand*{\FootnoteNewAcronymDef}{%
4266   \edef\@do@newglossaryentry{%
4267     \noexpand\newglossaryentry{\the\glslabeltok}%
4268     {%
4269       type=\acronymtype,%
4270       name={\noexpand\acronymfont{\the\glsshorttok}},%
4271       sort={\the\glsshorttok},%
4272       text={\the\glsshorttok},%
4273       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4274       short={\the\glsshorttok},%
4275       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4276       long={\the\glslongtok},%
4277       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4278       description={\the\glslongtok},%
4279       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4280       \the\glskeylisttok
4281     }%
4282   }%
4283   \@do@newglossaryentry
4284 }
```

otnoteAcronymStyle 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.

```

4285 \newcommand*{\SetFootnoteAcronymStyle}{%
4286   \renewcommand{\newacronym}[4][\]{%
4287     \ifx\@glsacronymlists\@empty
4288       \def\@glo@type{\acronymtype}%
4289       \setkeys{glossentry}{##1}%
4290       \DeclareAcronymList{\@glo@type}%
4291       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
4292     \fi
4293     \glskeylisttok{##1}%
4294     \glslabeltok{##2}%
4295     \glsshorttok{##3}%
4296     \glslongtok{##4}%
4297     \newacronymhook
4298     \FootnoteNewAcronymDef
4299   }%
```

Set display

```

4300   \@for\@gls@type:=\@glsacronymlists\do{%
4301     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
4302   }%
```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

4303 \ifglssacrsmallcaps
4304   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
4305   \renewcommand*{\acrpluralsuffix}{%
4306     \textup{\glspluralsuffix}}%
4307 \else
4308   \ifglssacrsmaller
4309     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
4310   \fi
4311 \fi

  Check for option clash
4312 \ifglssacrdua
4313   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
4314     can’t both be set}{}%
4315 \fi
4316 }%

```

glsdoparenifnotempty Do a space followed by the argument if the argument doesn’t expand to empty or `\relax`. If argument isn’t empty (or `\relax`), apply the macro to it given in the second argument.

```

4317 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
4318   \protected@edef\gls@tmp{#1}%
4319   \ifdefempty\gls@tmp
4320   {}%
4321   {%
4322     \if\gls@tmp\relax
4323     \else
4324       \space (#2{#1})%
4325     \fi
4326   }%
4327 }

```

AcronymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

4328 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
4329   \defglssdisplayfirst[#1]{##1##4\glsdoparenifnotempty{##3}{\protect\firstacronymfont}}
4330   \defglssdisplay[#1]{\acronymfont{##1}##4}%
4331 }

```

\SmallNewAcronymDef

```

4332 \newcommand*{\SmallNewAcronymDef}{%
4333   \edef\@do@newglossaryentry{%
4334     \noexpand\newglossaryentry{\the\glslabeltok}%
4335     {%
4336       type=\acronymtype,%
4337       name={\noexpand\acronymfont{\the\glsshorttok}},%
4338       sort={\the\glsshorttok},%
4339       text={\noexpand\@glo@symbol},%

```

Default to the short plural.

```
4340 plural={\noexpand\@glo@shortpl},%
4341 first={\the\glslongtok},%
```

Default to the long plural.

```
4342 firstplural={\noexpand\@glo@longpl},%
4343 short={\the\glsshorttok},%
4344 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4345 long={\the\glslongtok},%
4346 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4347 description={\noexpand\@glo@first},%
4348 descriptionplural={\noexpand\@glo@firstplural},%
4349 symbol={\the\glsshorttok},%
```

Default to the short plural.

```
4350 symbolplural={\noexpand\@glo@shortpl},%
4351 \the\glskeylisttok
4352 }%
4353 }%
4354 \@do@newglossaryentry
4355 }
```

`\SetSmallAcronymStyle` Neither footnote nor description required, but smallcaps or smaller specified.

Use the symbol key to store the short form and first to store the long form.

```
4356 \newcommand*\SetSmallAcronymStyle{%
4357 \renewcommand{\newacronym}[4][\]{%
4358 \ifx\@glsacronymlists\@empty
4359 \def\@glo@type{\acronymtype}%
4360 \setkeys{glossentry}{##1}%
4361 \DeclareAcronymList{\@glo@type}%
4362 \SetSmallAcronymDisplayStyle{\@glo@type}%
4363 \fi
4364 \glskeylisttok{##1}%
4365 \glslabeltok{##2}%
4366 \glsshorttok{##3}%
4367 \glslongtok{##4}%
4368 \newacronymhook
4369 \SmallNewAcronymDef
4370 }%
```

Change the display since first only contains long form.

```
4371 \for\@gls@type:=\@glsacronymlists\do{%
4372 \SetSmallAcronymDisplayStyle{\@gls@type}%
4373 }%
```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```
4374 \ifglsacrsmallcaps
4375 \renewcommand*\acronymfont[1]{\textsc{##1}}
```

```

4376 \renewcommand*{\acrpluralsuffix}{%
4377 \textup{\glspuralsuffix}}%
4378 \else
4379 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
4380 \fi

    check for option clash
4381 \ifglssacrdua
4382 \ifglssacrsmallcaps
4383 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
4384 can’t both be set}{}%
4385 \else
4386 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
4387 can’t both be set}{}%
4388 \fi
4389 \fi
4390 }%

```

\SetDUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

4391 \newcommand*{\SetDUADisplayStyle}[1]{%
4392 \defglssdisplay[#1]{##1##4}%
4393 \defglssdisplayfirst[#1]{##1##4}%
4394 }

```

\DUANewAcronymDef

```

4395 \newcommand*{\DUANewAcronymDef}{%
4396 \edef\@do@newglossaryentry{%
4397 \noexpand\newglossaryentry{\the\glslabeltok}%
4398 {%
4399 type=\acronymtype,%
4400 name={\the\glsshorttok},%
4401 text={\the\glslongtok},%
4402 plural={\the\glslongtok\noexpand\acrpluralsuffix},%
4403 short={\the\glsshorttok},%
4404 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4405 long={\the\glslongtok},%
4406 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4407 description={\the\glslongtok},%
4408 symbol={\the\glsshorttok},%
4409 symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4410 \the\glskeylisttok
4411 }%
4412 }%
4413 \@do@newglossaryentry
4414 }

```

\SetDUASStyle Always expand acronyms.

```

4415 \newcommand*{\SetDUASStyle}{%
4416 \renewcommand{\newacronym}[4][\]{%

```



```

4417 \ifx\@glsacronymlists\@empty
4418 \def\@glo@type{\acronymtype}%
4419 \setkeys{glossentry}{##1}%
4420 \DeclareAcronymList{\@glo@type}%
4421 \SetDUADisplayStyle{\@glo@type}%
4422 \fi
4423 \glskeylisttok{##1}%
4424 \glslabeltok{##2}%
4425 \glsshorttok{##3}%
4426 \glslongtok{##4}%
4427 \newacronymhook
4428 \DUANewAcronymDef
4429 }%
    Set the display
4430 \@for\@gls@type:=\@glsacronymlists\do{%
4431 \SetDUADisplayStyle{\@gls@type}%
4432 }%
4433 }

```

\SetAcronymStyle

```

4434 \newcommand*\SetAcronymStyle{%
4435 \SetDefaultAcronymStyle
4436 \ifglsacrdescription
4437 \ifglsacrfootnote
4438 \SetDescriptionFootnoteAcronymStyle
4439 \else
4440 \ifglsacrdua
4441 \SetDescriptionDUAAcronymStyle
4442 \else
4443 \SetDescriptionAcronymStyle
4444 \fi
4445 \fi
4446 \else
4447 \ifglsacrfootnote
4448 \SetFootnoteAcronymStyle
4449 \else
4450 \ifthenelse{\boolean{glsacrsmalldcaps}\OR
4451 \boolean{glsacrsmalld}}{
4452 {%
4453 \SetSmallAcronymStyle
4454 }%
4455 {%
4456 \ifglsacrdua
4457 \SetDUASyle
4458 \fi
4459 }%
4460 \fi
4461 \fi
4462 }

```

Set the acronym style according to the package options

```
4463 \SetAcronymStyle
```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\SetCustomDisplayStyle` Sets the acronym display style.

```
4464 \newcommand*\SetCustomDisplayStyle[1]{%
4465   \def\glsdisplay[#1]{##1##4}%
4466   \def\glsdisplayfirst[#1]{##1##4}%
4467 }
```

`\CustomAcronymFields`

```
4468 \newcommand*\CustomAcronymFields{%
4469   name={\the\glsshorttok},%
4470   description={\the\glslongtok},%
4471   first={\noexpand\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
4472   firstplural={\noexpand\acrfullformat
4473     {\the\glslongtok\noexpand\acrpluralsuffix}{\the\glsshorttok}}%
4474   text={\the\glsshorttok},%
4475   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
4476 }
```

`\CustomNewAcronymDef`

```
4477 \newcommand*\CustomNewAcronymDef{%
4478   \protected@edef\@do@newglossaryentry{%
4479     \noexpand\newglossaryentry{\the\glslabeltok}%
4480     {%
4481       type=\acronymtype,%
4482       short={\the\glsshorttok},%
4483       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4484       long={\the\glslongtok},%
4485       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4486       user1={\the\glsshorttok},%
4487       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
4488       user3={\the\glslongtok},%
4489       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
4490       \CustomAcronymFields,%
4491       \the\glskeylisttok
4492     }%
4493   }%
4494   \@do@newglossaryentry
4495 }
```

`\SetCustomStyle`

```

4496 \newcommand*{\SetCustomStyle}{%
4497   \renewcommand{\newacronym}[4][]{%
4498     \ifx\@glsacronymlists\@empty
4499       \def\@glo@type{\acronymtype}%
4500       \setkeys{glossentry}{##1}%
4501       \DeclareAcronymList{\@glo@type}%
4502       \SetCustomDisplayStyle{\@glo@type}%
4503     \fi
4504     \glskeylisttok{##1}%
4505     \glslabeltok{##2}%
4506     \glsshorttok{##3}%
4507     \gslongtok{##4}%
4508     \newacronymhook
4509     \CustomNewAcronymDef
4510   }%

  Set the display
4511   \@for\@gls@type:=\@glsacronymlists\do{%
4512     \SetCustomDisplayStyle{\@gls@type}%
4513   }%
4514 }

```

`\defineAcronymSynonyms`

```

4515 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

`\acs`

```

4516   \let\acs\acrshort

```

First letter uppercase short form

`\Acs`

```

4517   \let\Acs\Acrshort

```

Plural short form

`\acsp`

```

4518   \let\acsp\acrshortpl

```

First letter uppercase plural short form

`\Acsp`

```

4519   \let\Acsp\Acrshortpl

```

Long form

`\acl`

```

4520   \let\acl\aclong

```

Plural long form

`\aclp`
 4521 `\let\aclp\acrlongpl`
 First letter upper case long form

`\Acl`
 4522 `\let\Acl\Acrlong`
 First letter upper case plural long form

`\Aclp`
 4523 `\let\Aclp\Acrlongpl`
 Full form

`\acf`
 4524 `\let\acf\acrfull`
 Plural full form

`\acfp`
 4525 `\let\acfp\acrfullpl`
 First letter upper case full form

`\Acf`
 4526 `\let\Acf\Acrfull`
 First letter upper case plural full form

`\Acfp`
 4527 `\let\Acfp\Acrfullpl`
 Standard form

`\ac`
 4528 `\let\ac\gls`
 First upper case standard form

`\Ac`
 4529 `\let\Ac\Gls`
 Standard plural form

`\acp`
 4530 `\let\acp\glspl`
 Standard first letter upper case plural form

`\Acp`
 4531 `\let\Acp\Glspl`

```
4532 }
```

Define synonyms if required

```
4533 \ifglsacrshortcuts
4534   \DefineAcronymSynonyms
4535 \fi
```

1.18 Predefined Glossary Styles

The glossaries bundle comes with some predefined glossary styles. These need to be loaded now for the style option to use them.

First, the glossary hyper-navigation commands need to be loaded.

```
4536 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
4537 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long package` option is used.

```
4538 \@gls@loadlong
```

The styles that use the `supertabular` environment. These are not loaded if the `nosuper` package option is used or if the package isn't installed.

```
4539 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
4540 \@gls@loadtree
```

The default glossary style is set according to the `style` package option, but can be overridden by `\glossarystyle`. The required style must be defined at this point.

```
4541 \ifx\@glossary@default@style\relax
4542 \else
4543   \glossarystyle{\@glossary@default@style}
4544 \fi
```

1.19 Debugging Commands

`\showgloparent` `\showgloparent{\<label>}`

```
4545 \newcommand*{\showgloparent}[1]{%
4546   \expandafter\show\csname glo@#1@parent\endcsname
4547 }
```

`\showglolevel` `\showglolevel{\<label>}`

```

4548 \newcommand*{\showglolevel}[1]{%
4549   \expandafter\show\csname glo@#1@level\endcsname
4550 }

```

\showglolevel \showglolevel{<label>}

```

4551 \newcommand*{\showglolevel}[1]{%
4552   \expandafter\show\csname glo@#1@level\endcsname
4553 }

```

\showgloplural \showgloplural{<label>}

```

4554 \newcommand*{\showgloplural}[1]{%
4555   \expandafter\show\csname glo@#1@plural\endcsname
4556 }

```

\showglofirst \showglofirst{<label>}

```

4557 \newcommand*{\showglofirst}[1]{%
4558   \expandafter\show\csname glo@#1@first\endcsname
4559 }

```

\showglofirstpl \showglofirstpl{<label>}

```

4560 \newcommand*{\showglofirstpl}[1]{%
4561   \expandafter\show\csname glo@#1@firstpl\endcsname
4562 }

```

\showgloftype \showgloftype{<label>}

```

4563 \newcommand*{\showgloftype}[1]{%
4564   \expandafter\show\csname glo@#1@type\endcsname
4565 }

```

\showglocounter \showglocounter{<label>}

```

4566 \newcommand*{\showglocounter}[1]{%
4567   \expandafter\show\csname glo@#1@counter\endcsname
4568 }

```

\showglouserii \showglouserii{<label>}

```
4569 \newcommand*{\showglouserii}[1]{%
4570   \expandafter\show\csname glo@#1@userii\endcsname
4571 }
```

\showglouseriii \showglouseriii{<label>}

```
4572 \newcommand*{\showglouseriii}[1]{%
4573   \expandafter\show\csname glo@#1@useriii\endcsname
4574 }
```

\showglouseriv \showglouseriv{<label>}

```
4575 \newcommand*{\showglouseriv}[1]{%
4576   \expandafter\show\csname glo@#1@useriv\endcsname
4577 }
```

\showglouserv \showglouserv{<label>}

```
4578 \newcommand*{\showglouserv}[1]{%
4579   \expandafter\show\csname glo@#1@userv\endcsname
4580 }
```

\showglouservi \showglouservi{<label>}

```
4581 \newcommand*{\showglouservi}[1]{%
4582   \expandafter\show\csname glo@#1@uservi\endcsname
4583 }
```

\showgloname \showgloname{<label>}

```
4584 \newcommand*{\showgloname}[1]{%
4585   \expandafter\show\csname glo@#1@name\endcsname
4586 }
```

\showgloname \showgloname{<label>}

```
4587 \newcommand*{\showgloname}[1]{%
4588   \expandafter\show\csname glo@#1@name\endcsname
4589 }
```

`\showglodesc` `\showglodesc{<label>}`

```
4590 \newcommand*{\showglodesc}[1]{%
4591   \expandafter\show\csname glo@#1@desc\endcsname
4592 }
```

`\showglodescplural` `\showglodescplural{<label>}`

```
4593 \newcommand*{\showglodescplural}[1]{%
4594   \expandafter\show\csname glo@#1@descplural\endcsname
4595 }
```

`\showglosort` `\showglosort{<label>}`

```
4596 \newcommand*{\showglosort}[1]{%
4597   \expandafter\show\csname glo@#1@sort\endcsname
4598 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
4599 \newcommand*{\showglosymbol}[1]{%
4600   \expandafter\show\csname glo@#1@symbol\endcsname
4601 }
```

`\showglosymbolplural` `\showglosymbolplural{<label>}`

```
4602 \newcommand*{\showglosymbolplural}[1]{%
4603   \expandafter\show\csname glo@#1@symbolplural\endcsname
4604 }
```

`\showgloindex` `\showgloindex{<label>}`

```
4605 \newcommand*{\showgloindex}[1]{%
4606   \expandafter\show\csname glo@#1@index\endcsname
4607 }
```

`\showgloflag` `\showgloflag{<label>}`

```
4608 \newcommand*{\showgloflag}[1]{%
4609   \expandafter\show\csname ifglo@#1@flag\endcsname
4610 }
```


`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
4611 \newcommand*{\showacronymlists}{%  
4612   \show\@glsacronymlists  
4613 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
4614 \newcommand*{\showglossaries}{%  
4615   \show\@glo@types  
4616 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
4617 \newcommand*{\showglossaryin}[1]{%  
4618   \expandafter\show\csname @glo@type@#1@in\endcsname  
4619 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
4620 \newcommand*{\showglossaryout}[1]{%  
4621   \expandafter\show\csname @glo@type@#1@out\endcsname  
4622 }
```

`\showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
4623 \newcommand*{\showglossarytitle}[1]{%  
4624   \expandafter\show\csname @glo@type@#1@title\endcsname  
4625 }
```

`\showglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
4626 \newcommand*{\showglossarycounter}[1]{%  
4627   \expandafter\show\csname @glo@type@#1@counter\endcsname  
4628 }
```

`\showglossaryentries` `\showglossaryentries{\glossary-label}`

Show the list of entry labels for the given glossary.

```
4629 \newcommand*{\showglossaryentries}[1]{%
4630   \expandafter\show\csname glolist@#1\endcsname
4631 }
```

1.20 Compatibility with version 2.07 and below

In order to fix some bugs in v3.0, it was necessary to change the way information is written to the `glo` file, which also meant a change in the format of the Xindy style file. The compatibility option is meant for documents that use a customised Xindy style file with `\noist`. With the compatibility option, hopefully xindy will still be able to process the old document, but the bugs will remain. The issues in versions 2.07 and below:

- With xindy, the counter used by the entry was hard-coded into the Xindy style file. This meant that you couldn't use the counter to swap counters.
- With both xindy and makeindex, if used with hyperref and `\theH{counter}` was different to `\thecounter`, the link in the location number would be undefined.

```
4632 \csname ifglscpatible-2.07\endcsname
4633   \RequirePackage{glossaries-compatible-207}
4634 \fi
```

2 Mfirstuc Documented Code

```
4635 \NeedsTeXFormat{LaTeX2e}
4636 \ProvidesPackage{mfirstuc}[2012/05/21 v1.06 (NLCT)]
```

Requires etoolbox:

```
4637 \RequirePackage{etoolbox}
```

`\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, unless the group is empty. 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`.

```
4638 \newif\if@glscs
4639 \newtoks\@glsmfirst
4640 \newtoks\@glsmrest
4641 \def\makefirstuc#1{%
4642   \def\gls@argi{#1}%
4643   \ifx\gls@argi\@empty
```

If the argument is empty, do nothing.

```

4644 \else

4645 \def\@gls@tmp{\ #1}%
4646 \@onelevel@sanitize\@gls@tmp
4647 \expandafter\@gls@checkcs\@gls@tmp\relax\relax
4648 \if@glscs
4649 \@gls@getbody #1{}\@nil
4650 \ifx\@gls@rest\@empty
4651 \glsmakefirstuc{#1}%
4652 \else
4653 \expandafter\@gls@split\@gls@rest\@nil
4654 \ifx\@gls@first\@empty
4655 \glsmakefirstuc{#1}%
4656 \else
4657 \expandafter\@glsmfirst\expandafter{\@gls@first}%
4658 \expandafter\@glsmrest\expandafter{\@gls@rest}%
4659 \edef\@gls@domfirstuc{\noexpand\@gls@body
4660 {\noexpand\glsmakefirstuc\the\@glsmfirst}%
4661 \the\@glsmrest}%
4662 \@gls@domfirstuc
4663 \fi
4664 \fi
4665 \else
4666 \glsmakefirstuc{#1}%
4667 \fi
4668 \fi
4669 }

```

Put first argument in \@gls@first and second argument in \@gls@rest:

```

4670 \def\@gls@split#1#2\@nil{%
4671 \def\@gls@first{#1}\def\@gls@rest{#2}%
4672 }

4673 \def\@gls@checkcs#1 #2#3\relax{%
4674 \def\@gls@argi{#1}\def\@gls@argii{#2}%
4675 \ifx\@gls@argi\@gls@argii
4676 \@glscstrue
4677 \else
4678 \@glscsfalse
4679 \fi
4680 }

```

Make first thing upper case:

```

4681 \def\@gls@makefirstuc#1{\MakeUppercase #1}

```

\glsmakefirstuc Provide a user command to make it easier to customise.

```

4682 \newcommand*\glsmakefirstuc}[1]{\@gls@makefirstuc{#1}}

```

Get the first grouped argument and stores in \@gls@body.

```
4683 \def\@gls@getbody#1#\def\@gls@body{#1}\@gls@gobbletonil}
```

Scoup up everything to \@nil and store in \@gls@rest:

```
4684 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}
```

`\xmakefirstuc` Expand argument once before applying `\makefirstuc` (added v1.01).

```
4685 \newcommand*\xmakefirstuc}[1]{%
```

```
4686 \expandafter\makefirstuc\expandafter{#1}}
```

`\capitalisewords` Capitalise each word in the argument. Words are considered to be separated by plain spaces (i.e. non-breakable spaces won't be considered a word break).

```
4687 \newcommand*\capitalisewords}[1]{%
```

```
4688 \def\gls@add@space{ }%
```

```
4689 \mfu@capitalisewords#1 \@nil\mfu@endcap
```

```
4690 %\gls@add@space\makefirstuc{##1}\def\gls@add@space{ }%
```

```
4691 }
```

```
4692 \def\mfu@capitalisewords#1 #2\mfu@endcap{%
```

```
4693 \def\mfu@cap@first{#1}%
```

```
4694 \def\mfu@cap@second{#2}%
```

```
4695 \gls@add@space
```

```
4696 \makefirstuc{#1}%
```

```
4697 \def\gls@add@space{ }%
```

```
4698 \ifx\mfu@cap@second\@nnil
```

```
4699 \let\next@mfu@cap\mfu@noop
```

```
4700 \else
```

```
4701 \let\next@mfu@cap\mfu@capitalisewords
```

```
4702 \fi
```

```
4703 \next@mfu@cap#2\mfu@endcap
```

```
4704 }
```

```
4705 \def\mfu@noop#1\mfu@endcap{ }
```

`\xcapitalisewords` Short-cut command:

```
4706 \newcommand*\xcapitalisewords}[1]{%
```

```
4707 \expandafter\capitalisewords\expandafter{#1}%
```

```
4708 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
4709 \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 1.15.](#)) `\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

```
4710 \newcommand*\glsnavhyperlink}[3][\@glo@type]{%
4711   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
4712   \@glslink{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⟩. If ⟨type⟩ is omitted, \@glo@type is used which is set by \printglossary to the current glossary label.

\glsnavhypertarget

```
4713 \newcommand*\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
4714   \protected@write\@auxout{}\string\gls@hypergroup{#1}{#2}}%
  Add the target.
4715   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
4716   \expandafter\let
4717   \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
  Iterate through list and terminate loop if this group is found.
4718   \@for\@gls@elem:=\@gls@list\do{%
4719     \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
  Check if list terminated prematurely.
4720   \if@endfor
4721   \else
    This group was not included in the list, so issue a warning.
4722     \GlossariesWarningNoLine{Navigation panel
4723       for glossary type ‘#1’^^Jmissing group ‘#2’}%
4724     \gdef\gls@hypergroup@rerun{%
4725       \GlossariesWarningNoLine{Navigation panel
4726         has changed. Rerun LaTeX}}%
4727     \fi
4728 }
```

\gls@hypergroup@rerun Give a warning at the end if re-run required

```
4729 \let\gls@hypergroup@rerun\relax
4730 \AtEndDocument{\gls@hypergroup@rerun}
```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergrouplist@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```

4731 \newcommand*{\@gls@hypergroup}[2]{%
4732 \@ifundefined{\@gls@hypergrouplist@#1}{%
4733   \expandafter\xdef\csname \@gls@hypergrouplist@#1\endcsname{#2}%
4734 }{%
4735   \expandafter\let\expandafter\@gls@tmp
4736     \csname \@gls@hypergrouplist@#1\endcsname
4737   \expandafter\xdef\csname \@gls@hypergrouplist@#1\endcsname{%
4738     \@gls@tmp,#2}%
4739 }%
4740 }
```

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`

```

4741 \newcommand*{\glsnavigation}{%
4742 \def\@gls@between{}%
4743 \@ifundefined{\@gls@hypergrouplist@\@glo@type}{%
4744   \def\@gls@list{}%
4745 }{%
4746   \expandafter\let\expandafter\@gls@list
4747     \csname \@gls@hypergrouplist@\@glo@type\endcsname
4748 }%
4749 \@for\@gls@tmp:=\@gls@list\do{%
4750   \@gls@between
4751   \glsnavhyperlink{\@gls@tmp}{\glsgetgrouptitle{\@gls@tmp}}%
4752   \let\@gls@between\glshypernavsep%
4753 }%
4754 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```

4755 \newcommand*{\glshypernavsep}{\space\textbar\space}
```

The `\glssymbolnav` produces a simple navigation set of links for just the symbol and number groups. This used to be used at the start of `\glsnavigation`. This command is no longer needed.

`\glssymbolnav`

```

4756 \newcommand*{\glssymbolnav}{%
4757 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
```

```

4758 \glshypernavsep
4759 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
4760 \glshypernavsep
4761 }

```

3.2 In-line Style (glossary-inline.sty)

This defines an in-line style where the entries are comma-separated with just the name and description displayed.

```

4762 \ProvidesPackage{glossary-inline}[2012/09/21 v3.03 (NLCT)]

```

inline Define the inline style.

```

4763 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then
    changed by \glossaryentryfield)
4764 \renewenvironment{theglossary}%
4765     {%
4766         \def\gls@inlinesep{}%
4767         \def\gls@inlinesubsep{}%
4768         \def\gls@inlinepostchild{}%
4769     }%
4770     {\glspostinline}%

```

No header:

```

4771 \renewcommand*{\glossaryheader}{}%
    No group headings (if heading is required, add \glsinlinedopostchild to
    start definition in case heading follows a child entry):
4772 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

4773 \renewcommand{\glossaryentryfield}[5]{%
4774     \glsinlinedopostchild
4775     \gls@inlinesep
4776     \def\glo@desc{##3}%
4777     \def\@no@post@desc{\nopostdesc}%
4778     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
4779     \ifx\glo@desc\@no@post@desc
4780         \glsinlineemptydescformat{##4}{##5}%
4781     \else
4782         \ifstrempy{##3}%
4783             {\glsinlineemptydescformat{##4}{##5}}%
4784             {\glsinlinedescformat{##3}{##4}{##5}}%
4785     \fi
4786     \ifglshaschildren{##1}%
4787     {%
4788         \glsresetsubentrycounter
4789         \glsinlineparentchildseparator
4790         \def\gls@inlinesubsep{}%

```

```

4791     \def\gls@inlinepostchild{\glsinlinepostchild}%
4792   }%
4793   {}%
4794   \def\gls@inlinesep{\glsinlineseparator}%
4795   }%

```

Sub-entries display description:

```

4796   \renewcommand{\glossarysubentryfield}[6]{%
4797     \gls@inlinesubsep%
4798     \glsinlinesubnameformat{##2}{##3}%
4799     \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
4800   \def\gls@inlinesubsep{\glsinlinesubseparator}%
4801   }%

```

Nothing special between groups:

```

4802   \renewcommand*{\glsgroupskip}{}%
4803 }

```

`\glsinlinedopostchild`

```

4804 \newcommand*{\glsinlinedopostchild}{%
4805   \gls@inlinepostchild
4806   \def\gls@inlinepostchild{}%
4807 }

```

`\glsinlineseparator` Separator to use between entries.

```

4808 \newcommand*{\glsinlineseparator}{;\space}

```

`\glsinlinesubseparator` Separator to use between sub-entries.

```

4809 \newcommand*{\glsinlinesubseparator}{,\space}

```

`\glsinlineparentchildseparator` Separator to use between parent and children.

```

4810 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

`\glsinlinepostchild` Hook to use between child and next entry

```

4811 \newcommand*{\glsinlinepostchild}{}

```

`\glspostinline` Terminator for inline glossary.

```

4812 \newcommand*{\glspostinline}{\glspostdescription\space}

```

`\glsinlinenameformat` Formats the name of the entry (first argument label, second argument name):

```

4813 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

`\glsinlinedescformat` Formats the entry's description, symbol and location list:

```

4814 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

`\glsinlineemptydescformat` Formats the entry's symbol and location list when the description is empty:

```

4815 \newcommand*{\glsinlineemptydescformat}[2]{}

```


`inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
4816 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
4817 \newcommand*{\glsinlinesubdescformat}[3]{#1}
```

3.3 List Style (`glossary-list.sty`)

The style file 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.

```
4818 \ProvidesPackage{glossary-list}[2012/11/11 v3.04 (NLCT)]
```

`list` 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. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```
4819 \newglossarystyle{list}{%
```

Use description environment:

```
4820 \renewenvironment{theglossary}{%
```

```
4821 {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
4822 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
4823 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
4824 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4825 \item[\glsentryitem{##1}\glstarget{##1}{##2}]
```

```
4826 ##3\glspostdescription\space ##5}%
```

Sub-entries continue on the same line:

```
4827 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
4828 \glssubentryitem{##2}%
```

```
4829 \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
```

```
4830 % \end{macrocode}
```

```
4831 % Add vertical space between groups:
```

```
4832 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
```

```
4833 % \begin{macrocode}
```

```
4834 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
```

```
4835 }
```

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

```
4836 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
4837 \glossarystyle{list}%
```

Each group has a heading:

```
4838 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##1}]}}
```

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

```
4839 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
4840 \glossarystyle{list}%
```

Add navigation links at the start of the environment:

```
4841 \renewcommand*{\glossaryheader}{%
```

```
4842 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
4843 \renewcommand*{\glsgroupheading}[1]{%
```

```
4844 \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
4845 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
4846 \glossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
4847 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4848 \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
```

Version 3.04 changed `\newline` to the following paragraph break stuff (thanks to Daniel Gebhardt for supplying the fix) to prevent a page break occurring at this point.

```
4849 \mbox{} \par \nobreak \@afterheading
```

```
4850 ##3 \glspostdescription \space ##5}%
```

Sub-entries start a new paragraph:

```
4851 \renewcommand{\glossarysubentryfield}[6]{%
```

```
4852 \par
```

```
4853 \glssubentryitem{##2}%
```

```
4854 \glstarget{##2}{\strut}##4 \glspostdescription \space ##6}%
```

```
4855 }
```

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

```
4856 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
4857 \glossarystyle{altlist}%
```

Each group has a heading:

```
4858 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##1}]}
```

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

```
4859 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
4860 \glossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
4861 \renewcommand*{\glossaryheader}{%
```

```
4862 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
4863 \renewcommand*{\glsgroupheading}[1]{%
```

```
4864 \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}
```

listdotted 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. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
4865 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
4866 \glossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
4867 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4868 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
4869 \glentryitem{##1}\glstarget{##1}{##2}%
```

```
4870 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
```

Sub entries have the same format as main entries:

```
4871 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
4872 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
4873 \glssubentryitem{##2}%
```

```
4874 \glstarget{##2}{##3}%
```

```
4875 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
```

```
4876 }
```

\glslistdottedwidth

```
4877 \newlength\glslistdottedwidth
```

```
4878 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
4879 \newglossarystyle{sublistdotted}{%
    Base it on the listdotted style:
4880   \glossarystyle{listdotted}%
    Main (level 0) entries just display the name:
4881   \renewcommand*{\glossaryentryfield}[5]{%
4882     \item[\glentryitem{##1}\glstarget{##1}{##2}]}%
4883 }
```

3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
4884 \ProvidesPackage{glossary-long}[2012/09/21 v3.03 (NLCT)]
    Requires the package:
4885 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. (There's a chance that the user may specify `nolong` and then load later, in which case `\glsdescwidth` may have already been defined by . The same goes for `\glspagelistwidth`.)

```
4886 \@ifundefined{glsdescwidth}{%
4887   \newlength\glsdescwidth
4888   \setlength{\glsdescwidth}{0.6\hsize}
4889 }
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
4890 \@ifundefined{glspagelistwidth}{%
4891   \newlength\glspagelistwidth
4892   \setlength{\glspagelistwidth}{0.1\hsize}
4893 }
```

`long` The long glossary style command which uses the `longtable` environment:

```
4894 \newglossarystyle{long}{%
    Use longtable with two columns:
4895   \renewenvironment{theglossary}%
4896     {\begin{longtable}\lp{\glsdescwidth}}%
4897     {\end{longtable}}%
    Do nothing at the start of the environment:
4898   \renewcommand*{\glossaryheader}{}%
    No heading between groups:
4899   \renewcommand*{\glsgroupheading}[1]{}%
}
```

Main (level 0) entries displayed in a row:

```
4900 \renewcommand*{\glossaryentryfield}[5]{%
4901     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
```

Sub entries displayed on the following row without the name:

```
4902 \renewcommand*{\glossarysubentryfield}[6]{%
4903     &
4904     \glssubentryitem{##2}%
4905     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
```

Blank row between groups:

```
4906 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \\fi}%
4907 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
4908 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
4909 \glossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
4910 \renewenvironment{theglossary}{%
4911     \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
4912 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
4913 }
```

longheader The longheader style is like the long style but with a header:

```
4914 \newglossarystyle{longheader}{%
```

Base it on the glostylelong style:

```
4915 \glossarystyle{long}%
```

Set the table's header:

```
4916 \renewcommand*{\glossaryheader}{%
4917     \bfseries \entryname & \bfseries \descriptionname\\\endhead}%
4918 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
4919 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
4920 \glossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
4921 \renewcommand*{\glossaryheader}{%
4922     \hline\bfseries \entryname & \bfseries \descriptionname\\\hline
4923     \endhead
4924     \hline\endfoot}%
4925 }
```

long3col The long3col style is like long but with 3 columns

```
4926 \newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
4927 \renewenvironment{theglossary}%
```

```
4928 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
```

```
4929 {\end{longtable}}%
```

No table header:

```
4930 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
4931 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
4932 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4933 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
4934 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
4935 &
```

```
4936 \glssubentryitem{##2}%
```

```
4937 \glstarget{##2}{\strut}##4 & ##6\\}%
```

Blank row between groups:

```
4938 \renewcommand*{\glsgroupskip}{\ifglsgnpgroupskip\else & &\\fi}%
```

```
4939 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
4940 \newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
4941 \glossarystyle{long3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
4942 \renewenvironment{theglossary}%
```

```
4943 {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}}%
```

```
4944 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
4945 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
4946 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
4947 \newglossarystyle{long3colheader}{%
```

Base it on the glostylelong3col style:

```
4948 \glossarystyle{long3col}%
```

Set the table's header:

```

4949 \renewcommand*{\glossaryheader}{%
4950 \bfseries\entryname&\bfseries\descriptionname&
4951 \bfseries\pagelistname\\\endhead}%
4952 }

```

`long3colheaderborder` The `long3colheaderborder` style is like the above but with a border

```

4953 \newglossarystyle{long3colheaderborder}{%

```

Base it on the `glostylelong3colborder` style:

```

4954 \glossarystyle{long3colborder}%

```

Set the table's header and add horizontal line at table's foot:

```

4955 \renewcommand*{\glossaryheader}{%
4956 \hline
4957 \bfseries\entryname&\bfseries\descriptionname&
4958 \bfseries\pagelistname\\\hline\endhead
4959 \hline\endfoot}%
4960 }

```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```

4961 \newglossarystyle{long4col}{%

```

Use a `longtable` with 4 columns:

```

4962 \renewenvironment{theglossary}%
4963 {\begin{longtable}{llll}}%
4964 {\end{longtable}}%

```

No table header:

```

4965 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

4966 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```

4967 \renewcommand*{\glossaryentryfield}[5]{%
4968 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%

```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

4969 \renewcommand*{\glossarysubentryfield}[6]{%
4970 &
4971 \glssubentryitem{##2}%
4972 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%

```

Blank row between groups:

```

4973 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & \\\fi}%
4974 }

```

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

```
4975 \newglossarystyle{long4colheader}{%  
    Base it on the glostylelong4col style:  
4976   \glossarystyle{long4col}%  
    Table has a header:  
4977   \renewcommand*{\glossaryheader}{%  
4978     \bfseries\entryname&\bfseries\descriptionname&  
4979     \bfseries \symbolname&  
4980     \bfseries\pagelistname\\\endhead}%  
4981 }
```

`long4colborder` The `long4colborder` style is like `long4col` but with a border.

```
4982 \newglossarystyle{long4colborder}{%  
    Base it on the glostylelong4col style:  
4983   \glossarystyle{long4col}%  
    Use a longtable with 4 columns surrounded by vertical lines:  
4984   \renewenvironment{theglossary}%  
4985     {\begin{longtable}{|l|l|l|l|}}%  
4986     {\end{longtable}}%  
    Add horizontal lines to the head and foot of the table:  
4987   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
4988 }
```

`long4colheaderborder` The `long4colheaderborder` style is like the above but with a border.

```
4989 \newglossarystyle{long4colheaderborder}{%  
    Base it on the glostylelong4col style:  
4990   \glossarystyle{long4col}%  
    Use a longtable with 4 columns surrounded by vertical lines:  
4991   \renewenvironment{theglossary}%  
4992     {\begin{longtable}{|l|l|l|l|}}%  
4993     {\end{longtable}}%  
    Add table header and horizontal line at the table's foot:  
4994   \renewcommand*{\glossaryheader}{%  
4995     \hline\bfseries\entryname&\bfseries\descriptionname&  
4996     \bfseries \symbolname&  
4997     \bfseries\pagelistname\\\hline\endhead\hline\endfoot}%  
4998 }
```

`altlong4col` The `altlong4col` style is like the `long4col` style but can have multiline descriptions and page lists.

```
4999 \newglossarystyle{altlong4col}{%  
    Base it on the glostylelong4col style:  
5000   \glossarystyle{long4col}%
```


Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5001 \renewenvironment{theglossary}%
5002   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
5003   {\end{longtable}}%
5004 }
```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
5005 \newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
5006 \glossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5007 \renewenvironment{theglossary}%
5008   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
5009   {\end{longtable}}%
5010 }
```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
5011 \newglossarystyle{altlong4colborder}{%
```

Base it on the glostylelong4colborder style:

```
5012 \glossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5013 \renewenvironment{theglossary}%
5014   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
5015   {\end{longtable}}%
5016 }
```

ong4colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
5017 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the glostylelong4colheaderborder style:

```
5018 \glossarystyle{long4colheaderborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5019 \renewenvironment{theglossary}%
5020   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
5021   {\end{longtable}}%
5022 }
```

3.5 Glossary Styles using longtable (the glossary-longragged package)

The glossary styles defined in the package used the longtable environment in the glossary and use ragged right formatting for the multiline columns.

5023 \ProvidesPackage{glossary-longragged}[2012/09/21 v3.03 (NLCT)]

Requires the package:

5024 \RequirePackage{array}

Requires the package:

5025 \RequirePackage{longtable}

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
5026 \@ifundefined{glsdescwidth}{%
5027   \newlength{glsdescwidth
5028   \setlength{glsdescwidth}{0.6\hsize}
5029 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
5030 \@ifundefined{glspagelistwidth}{%
5031   \newlength{glspagelistwidth
5032   \setlength{glspagelistwidth}{0.1\hsize}
5033 }{}
```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

5034 \newglossarystyle{longragged}{%

Use longtable with two columns:

```
5035   \renewenvironment{theglossary}{%
5036     {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%
5037     {\end{longtable}}%
```

Do nothing at the start of the environment:

5038 \renewcommand*{\glossaryheader}{}%

No heading between groups:

5039 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries displayed in a row:

```
5040 \renewcommand*{\glossaryentryfield}[5]{%
5041   \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
5042   \tabularnewline}%

```

Sub entries displayed on the following row without the name:

```
5043 \renewcommand*{\glossarysubentryfield}[6]{%
5044   &
5045   \glssubentryitem{##2}%

```

```

5046     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
5047     \tabularnewline}%

```

Blank row between groups:

```

5048     \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%
5049 }

```

longraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

```

5050 \newglossarystyle{longraggedborder}{%

```

Base it on the glosstylelongragged style:

```

5051     \glossarystyle{longragged}%

```

Use longtable with two columns with vertical lines between each column:

```

5052     \renewenvironment{theglossary}{%
5053         \begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|}%
5054         {\end{longtable}}%

```

Place horizontal lines at the head and foot of the table:

```

5055     \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
5056 }

```

longraggedheader The longraggedheader style is like the longragged style but with a header:

```

5057 \newglossarystyle{longraggedheader}{%

```

Base it on the glosstylelongragged style:

```

5058     \glossarystyle{longragged}%

```

Set the table's header:

```

5059     \renewcommand*{\glossaryheader}{%
5060         \bfseries \entryname & \bfseries \descriptionname
5061         \tabularnewline\endhead}%
5062 }

```

longraggedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```

5063 \newglossarystyle{longraggedheaderborder}{%

```

Base it on the glosstylelongraggedborder style:

```

5064     \glossarystyle{longraggedborder}%

```

Set the table's header and add horizontal line to table's foot:

```

5065     \renewcommand*{\glossaryheader}{%
5066         \hline\bfseries \entryname & \bfseries \descriptionname
5067         \tabularnewline\hline
5068         \endhead
5069         \hline\endfoot}%
5070 }

```

longragged3col The longragged3col style is like longragged but with 3 columns

```

5071 \newglossarystyle{longragged3col}{%

```

Use a longtable with 3 columns:

```
5072 \renewenvironment{theglossary}%  
5073   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%  
5074     >{\raggedright}p{\glspagelistwidth}}}%  
5075   {\end{longtable}}%
```

No table header:

```
5076 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
5077 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
5078 \renewcommand*{\glossaryentryfield}[5]{%  
5079   \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
5080 \renewcommand*{\glossarysubentryfield}[6]{%  
5081   &  
5082   \glssubentryitem{##2}%  
5083   \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
5084 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%  
5085 }
```

`longragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
5086 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
5087 \glossarystyle{longragged3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
5088 \renewenvironment{theglossary}%  
5089   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|%  
5090     >{\raggedright}p{\glspagelistwidth}|}%  
5091   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
5092 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
5093 }
```

`longragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
5094 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
5095 \glossarystyle{longragged3col}%
```

Set the table's header:

```
5096 \renewcommand*{\glossaryheader}{%
5097 \bfseries\entryname&\bfseries\descriptionname&
5098 \bfseries\pagelistname\tabularnewline\endhead}%
5099 }
```

`ged3colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
5100 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
5101 \glossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
5102 \renewcommand*{\glossaryheader}{%
5103 \hline
5104 \bfseries\entryname&\bfseries\descriptionname&
5105 \bfseries\pagelistname\tabularnewline\hline\endhead
5106 \hline\endfoot}%
5107 }
```

`altlongragged4col` The `altlongragged4col` style is like the `altlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
5108 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5109 \renewenvironment{theglossary}%
5110 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
5111 >{\raggedright}p{\glspagelistwidth}}}%
5112 {\end{longtable}}%
```

No table header:

```
5113 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5114 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```
5115 \renewcommand*{\glossaryentryfield}[5]{%
5116 \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
5117 \renewcommand*{\glossarysubentryfield}[6]{%
5118 &
5119 \glssubentryitem{##2}%
5120 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
```

Blank row between groups:

```
5121 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & \tabularnewline\fi}%  
5122 }
```

ongragged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
5123 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
5124 \glossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5125 \renewenvironment{theglossary}%  
5126 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%  
5127 >{\raggedright}p{\glspagelistwidth}}}%  
5128 {\end{longtable}}%
```

Table has a header:

```
5129 \renewcommand*{\glossaryheader}{%  
5130 \bfseries\entryname&\bfseries\descriptionname&  
5131 \bfseries \symbolname&  
5132 \bfseries\pagelistname\tabularnewline\endhead}%  
5133 }
```

ongragged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
5134 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the glostylealtlongragged4col style:

```
5135 \glossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5136 \renewenvironment{theglossary}%  
5137 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%  
5138 >{\raggedright}p{\glspagelistwidth}|}%  
5139 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
5140 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
5141 }
```

ged4colheaderborder The altlongragged4colheaderborder style is like the above but with a header as well as a border.

```
5142 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the glostylealtlongragged4col style:

```
5143 \glossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5144 \renewenvironment{theglossary}%
5145   {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%
5146     >{\raggedright}p{\glspagelistwidth}|}%
5147   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
5148 \renewcommand*{\glossaryheader}{%
5149   \hline\bfseries\entryname&\bfseries\descriptionname&
5150   \bfseries \symbolname&
5151   \bfseries\pagelistname\tabularnewline\hline\endhead
5152   \hline\endfoot}%
5153 }
```

3.6 Glossary Styles using multicol (glossary-mcols.sty)

The style file defines glossary styles that use the multicol package. These use the tree-like glossary styles in a multicol environment.

```
5154 \ProvidesPackage{glossary-mcols}[2013/04/21 v3.05 (NLCT)]
```

Required packages:

```
5155 \RequirePackage{multicol}
5156 \RequirePackage{glossary-tree}
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
5157 \newcommand*{\glsmcols}{2}
```

`mcolindex` Multi-column index style. Same as the index, but puts the glossary in multiple columns. (Ideally the glossary title should go in the optional argument of multicols, but the title isn't part of the glossary style.)

```
5158 \newglossarystyle{mcolindex}{%
5159   \glossarystyle{index}%
5160   \renewenvironment{theglossary}%
5161     {%
5162       \begin{multicols}{\glsmcols}
5163       \setlength{\parindent}{0pt}%
5164       \setlength{\parskip}{0pt plus 0.3pt}%
5165       \let\item\@idxitem}%
5166     {\end{multicols}}%
5167 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
5168 \newglossarystyle{mcolindexgroup}{%
5169   \glossarystyle{mcolindex}%
5170   \renewcommand*{\glsgroupheading}[1]{%
5171     \item\textbf{\glsggetgrouptitle{##1}}\indexspace}%
5172 }
```

`mcolindexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
5173 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
5174 \glossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
5175 \renewcommand*{\glsaryheader}{%
```

```
5176 \item\textbf{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
5177 \renewcommand*{\glsgroupheading}[1]{%
```

```
5178 \item\textbf{\glsnavhypertarget{##1}}{\glsgetgrouptitle{##1}}}%
```

```
5179 \indexspace}%
```

```
5180 }
```

`mcoltree` Multi-column index style. Same as the `tree`, but puts the glossary in multiple columns.

```
5181 \newglossarystyle{mcoltree}{%
```

```
5182 \glossarystyle{tree}%
```

```
5183 \renewenvironment{theglossary}%
```

```
5184 {%
```

```
5185 \begin{multicols}{\glsmcols}
```

```
5186 \setlength{\parindent}{0pt}%
```

```
5187 \setlength{\parskip}{0pt plus 0.3pt}%
```

```
5188 }%
```

```
5189 {\end{multicols}}}%
```

```
5190 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```
5191 \newglossarystyle{mcoltreegroup}{%
```

Base it on the `glostylemcoltree` style:

```
5192 \glossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
5193 \renewcommand{\glsgroupheading}[1]{\par
```

```
5194 \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
```

```
5195 }
```

`mcoltreehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
5196 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
5197 \glossarystyle{mcoltree}%
```


Put navigation links to the groups at the start of the theglossary environment:

```
5198 \renewcommand*{\glossaryheader}{%
5199 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
5200 \renewcommand*{\glsgroupheading}[1]{%
5201 \par\noindent
5202 \textbf{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
5203 \indexspace}%
5204 }
```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
5205 \newglossarystyle{mcoltreenoname}{%
5206 \glossarystyle{treenoname}%
5207 \renewenvironment{theglossary}%
5208 {%
5209 \begin{multicols}{\glsmcols}
5210 \setlength{\parindent}{0pt}%
5211 \setlength{\parskip}{0pt plus 0.3pt}%
5212 }%
5213 {\end{multicols}}}%
5214 }
```

mcoltreenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```
5215 \newglossarystyle{mcoltreenonamegroup}{%
5216 \glossarystyle{mcoltreenoname}%
5217 \renewcommand{\glsgroupheading}[1]{\par
5218 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
5219 }
```

treenonamehypergroup The mcoltreenonamehypergroup style is like the mcoltreenonamegroup style, but has a set of links to the groups at the start of the glossary.

```
5220 \newglossarystyle{mcoltreenonamehypergroup}{%
5221 \glossarystyle{mcoltreenoname}%
5222 \renewcommand*{\glossaryheader}{%
5223 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
5224 \renewcommand*{\glsgroupheading}[1]{%
5225 \par\noindent
5226 \textbf{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
```

```

5227 \indexspace}%
5228 }

```

mcolalttree Multi-column index style. Same as the **alttree**, but puts the glossary in multiple columns.

```

5229 \newglossarystyle{mcolalttree}{%
5230 \glossarystyle{alttree}%
5231 \renewenvironment{theglossary}%
5232 {%
5233 \begin{multicols}{\glsmcols}
5234 \def\@gls@prevlevel{-1}%
5235 \mbox{}\par
5236 }%
5237 {\par\end{multicols}}%
5238 }

```

mcolalttreegroup Like the **mcolalttree** style but the glossary groups have headings.

```

5239 \newglossarystyle{mcolalttreegroup}{%
    Base it on the glostylemcolalttree style:
5240 \glossarystyle{mcolalttree}%
    Give each group a heading.
5241 \renewcommand{\glsgroupheading}[1]{\par
5242 \def\@gls@prevlevel{-1}%
5243 \hangindent0pt\relax
5244 \parindent0pt\relax
5245 \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
5246 }

```

colalttreehypergroup The **mcolalttreehypergroup** style is like the **mcolalttreegroup** style, but has a set of links to the groups at the start of the glossary.

```

5247 \newglossarystyle{mcolalttreehypergroup}{%
    Base it on the glostylemcolalttree style:
5248 \glossarystyle{mcolalttree}%
    Put the navigation links in the header
5249 \renewcommand*\glossaryheader{%
5250 \par
5251 \def\@gls@prevlevel{-1}%
5252 \hangindent0pt\relax
5253 \parindent0pt\relax
5254 \textbf{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
5255 \renewcommand*\glsgroupheading[1]{%
5256 \par
5257 \def\@gls@prevlevel{-1}%
5258 \hangindent0pt\relax

```

```

5259 \parindent0pt\relax
5260 \textbf{\glsnahypertarget{##1}{\glsggetgrouptitle{##1}}}\par
5261 \indexspace}}

```

3.7 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
5262 \ProvidesPackage{glossary-super}[2012/09/21 v3.03 (NLCT)]
```

Requires the package:

```
5263 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if has been loaded.

```

5264 \@ifundefined{glsdescwidth}{%
5265 \newlength{glsdescwidth
5266 \setlength{glsdescwidth}{0.6\hsize}
5267 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

5268 \@ifundefined{glspagelistwidth}{%
5269 \newlength{glspagelistwidth
5270 \setlength{glspagelistwidth}{0.1\hsize}
5271 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
5272 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

5273 \renewenvironment{theglossary}%
5274 {\tablehead{}\tabletail}%
5275 \begin{supertabular}{lp{glsdescwidth}}%
5276 {\end{supertabular}}%

```

Do nothing at the start of the table:

```
5277 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5278 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

5279 \renewcommand*{\glossaryentryfield}[5]{%
5280 \glstryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%

```

Sub entries put in a row (no name, description and page list in second column):

```
5281 \renewcommand*{\glossarysubentryfield}[6]{%
5282     &
5283     \glssubentryitem{##2}%
5284     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
```

Blank row between groups:

```
5285 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \\fi}%
5286 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
5287 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
5288 \glossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
5289 \renewenvironment{theglossary}%
5290     {\tablehead{\hline}\tabletail{\hline}%
5291     \begin{supertabular}{|l|p{\glstdescwidth}|}%
5292     {\end{supertabular}}}%
5293 }
```

superheader The superheader style is like the super style, but with a header:

```
5294 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
5295 \glossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
5296 \renewenvironment{theglossary}%
5297     {\tablehead{\bfseries \entryname & \bfseries \descriptionname\\}%
5298     \tabletail{}}%
5299     \begin{supertabular}{|lp{\glstdescwidth}|}%
5300     {\end{supertabular}}}%
5301 }
```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
5302 \newglossarystyle{superheaderborder}{%
```

Base it on the `glostylesuper` style:

```
5303 \glossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
5304 \renewenvironment{theglossary}%
5305     {\tablehead{\hline\bfseries \entryname &
5306     \bfseries \descriptionname\\ \hline}%
5307     \tabletail{\hline}}
```

```

5308     \begin{supertabular}{|l|p{\glstdescwidth}|}%
5309     {\end{supertabular}}%
5310 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

5311 \newglossarystyle{super3col}{%
    Put the glossary in a supertabular environment with three columns and no head
    or tail:
5312     \renewenvironment{theglossary}%
5313     {\tablehead{}\tabletail}%
5314     \begin{supertabular}{lp{\glstdescwidth}p{\glspagelistwidth}}%
5315     {\end{supertabular}}%
    Do nothing at the start of the table:
5316     \renewcommand*{\glossaryheader}{}%
    No group headings:
5317     \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries on a row (name in first column, description in second
    column, page list in last column):
5318     \renewcommand*{\glossaryentryfield}[5]{%
5319         \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
    Sub entries on a row (no name, description in second column, page list in last
    column):
5320     \renewcommand*{\glossarysubentryfield}[6]{%
5321         &
5322         \glssubentryitem{##2}%
5323         \glstarget{##2}{\strut}##4 & ##6\\}%
    Blank row between groups:
5324     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \\\fi}%
5325 }

```

super3colborder The super3colborder style is like the super3col style, but with a border:

```

5326 \newglossarystyle{super3colborder}{%
    Base it on the glostylesuper3col style:
5327     \glossarystyle{super3col}%
    Put the glossary in a supertabular environment with three columns and a hori-
    zontal line in the head and tail:
5328     \renewenvironment{theglossary}%
5329     {\tablehead{\hline}\tabletail{\hline}%
5330     \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
5331     {\end{supertabular}}%
5332 }

```

super3colheader The super3colheader style is like the super3col style but with a header row:

```

5333 \newglossarystyle{super3colheader}{%

```

Base it on the `glostylessuper3col` style:

```
5334 \glossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
5335 \renewenvironment{theglossary}%
5336   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5337     \bfseries\pagelistname\\}\tabletail{}}%
5338   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
5339   {\end{supertabular}}%
5340 }
```

`super3colheaderborder` The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```
5341 \newglossarystyle{super3colheaderborder}{%
```

Base it on the `glostylessuper3colborder` style:

```
5342 \glossarystyle{super3colborder}%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
5343 \renewenvironment{theglossary}%
5344   {\tablehead{\hline
5345     \bfseries\entryname&\bfseries\descriptionname&
5346     \bfseries\pagelistname\\ \hline}%
5347   \tabletail{\hline}%
5348   \begin{supertabular}{|lp{\glsgdescwidth}|p{\glspagelistwidth}|}%
5349   {\end{supertabular}}%
5350 }
```

`super4col` 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.

```
5351 \newglossarystyle{super4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
5352 \renewenvironment{theglossary}%
5353   {\tablehead{}\tabletail{}}%
5354   \begin{supertabular}{llll}}}%
5355   \end{supertabular}}%
```

Do nothing at the start of the table:

```
5356 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5357 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row with the name in the first column, description in second column, symbol in third column and page list in last column:

```
5358 \renewcommand*{\glossaryentryfield}[5]{%
5359   \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
5360 \renewcommand*{\glossarysubentryfield}[6]{%
5361     &
5362     \glssubentryitem{##2}%
5363     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
```

Blank row between groups:

```
5364 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & \\fi}%
5365 }
```

super4colheader The **super4colheader** style is like the **super4col** but with a header row.

```
5366 \newglossarystyle{super4colheader}{%
```

Base it on the **glostylesuper4col** style:

```
5367 \glossarystyle{super4col}%
```

Put the glossary in a **supertabular** environment with four columns, a header and no tail:

```
5368 \renewenvironment{theglossary}%
5369     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5370                 \bfseries\symbolname &
5371                 \bfseries\pagelistname\\}}%
5372     \tabletail{}}%
5373     \begin{supertabular}{|l|l|l|l|}%
5374     {\end{supertabular}}%
5375 }
```

super4colborder The **super4colborder** style is like the **super4col** but with a border.

```
5376 \newglossarystyle{super4colborder}{%
```

Base it on the **glostylesuper4col** style:

```
5377 \glossarystyle{super4col}%
```

Put the glossary in a **supertabular** environment with four columns and a horizontal line in the head and tail:

```
5378 \renewenvironment{theglossary}%
5379     {\tablehead{\hline}\tabletail{\hline}%
5380     \begin{supertabular}{|l|l|l|l|}%
5381     {\end{supertabular}}%
5382 }
```

super4colheaderborder The **super4colheaderborder** style is like the **super4col** but with a header and border.

```
5383 \newglossarystyle{super4colheaderborder}{%
```

Base it on the **glostylesuper4col** style:

```
5384 \glossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```

5385 \renewenvironment{theglossary}%
5386   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
5387     \bfseries\symbolname &
5388     \bfseries\pagelistname\\ \hline}\tabletail{\hline}%
5389   \begin{supertabular}{|l|l|l|l|}%
5390   {\end{supertabular}}%
5391 }
```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```
5392 \newglossarystyle{altsuper4col}{%
```

Base it on the glostylesuper4col style:

```
5393 \glossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

5394 \renewenvironment{theglossary}%
5395   {\tablehead{}}\tabletail{}%
5396   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
5397   {\end{supertabular}}%
5398 }
```

altsuper4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```
5399 \newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylesuper4colheader style:

```
5400 \glossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

5401 \renewenvironment{theglossary}%
5402   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5403     \bfseries\symbolname &
5404     \bfseries\pagelistname\\}\tabletail{}}%
5405   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
5406   {\end{supertabular}}%
5407 }
```

altsuper4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```
5408 \newglossarystyle{altsuper4colborder}{%
```

Base it on the glostylesuper4colborder style:

```
5409 \glossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
5410 \renewenvironment{theglossary}%
```



```

5411    {\tablehead{\hline}\tabletail{\hline}%
5412     \begin{supertabular}%
5413       {\lllp{\glsdescwidth}\lllp{\glspagelistwidth}}}%
5414     {\end{supertabular}}}%
5415 }

```

`per4colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

5416 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylesuper4colheaderborder` style:

```

5417 \glossarystyle{super4colheaderborder}%

```

Put the glossary in a `supertabular` environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```

5418 \renewenvironment{theglossary}%
5419   {\tablehead{\hline
5420     \bfseries\entryname &
5421     \bfseries\descriptionname &
5422     \bfseries\symbolname &
5423     \bfseries\pagelistname\\\hline}%
5424   \tabletail{\hline}%
5425   \begin{supertabular}%
5426     {\lllp{\glsdescwidth}\lllp{\glspagelistwidth}}}%
5427   {\end{supertabular}}}%
5428 }

```

3.8 Glossary Styles using `supertabular` environment (`glossary-superragged` package)

The glossary styles defined in the package use the `supertabular` environment. These styles are like those provided by the package, except that the multiline columns have ragged right justification.

```

5429 \ProvidesPackage{glossary-superragged}[2012/09/21 v3.03 (NLCT)]

```

Requires the package:

```

5430 \RequirePackage{array}

```

Requires the package:

```

5431 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

5432 \@ifundefined{glsdescwidth}{%
5433   \newlength\glsdescwidth
5434   \setlength{\glsdescwidth}{0.6\hsize}
5435 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
5436 \@ifundefined{glspagelistwidth}{%  
5437   \newlength{glspagelistwidth}  
5438   \setlength{glspagelistwidth}{0.1\hsize}  
5439 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
5440 \newglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns and no head or tail:

```
5441   \renewenvironment{theglossary}%  
5442     {\tablehead{}\tabletail{}}%  
5443     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}}}%  
5444     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
5445   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5446   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
5447   \renewcommand*{\glossaryentryfield}[5]{%  
5448     \glstryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%  
5449     \tabularnewline}%
```

Sub entries put in a row (no name, description and page list in second column):

```
5450   \renewcommand*{\glossarysubentryfield}[6]{%  
5451     &  
5452     \glssubentryitem{##2}%  
5453     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%  
5454     \tabularnewline}%
```

Blank row between groups:

```
5455   \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%  
5456 }
```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
5457 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylessuperragged` style:

```
5458   \glossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
5459   \renewenvironment{theglossary}%  
5460     {\tablehead{\hline}\tabletail{\hline}}%
```

```

5461     \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
5462     {\end{supertabular}}}%
5463 }

```

superraggedheader The superraggedheader style is like the super style, but with a header:

```

5464 \newglossarystyle{superraggedheader}{%
    Base it on the glostylesuperragged style:
5465     \glossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and
    no tail:
5466 \renewenvironment{theglossary}%
5467     {\tablehead{\bfseries \entryname & \bfseries \descriptionname
5468         \tabularnewline}%
5469     \tabletail{}}%
5470     \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
5471     {\end{supertabular}}}%
5472 }

```

superraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```

5473 \newglossarystyle{superraggedheaderborder}{%
    Base it on the glostylesuper style:
5474     \glossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and
    horizontal lines above and below the table:
5475     \renewenvironment{theglossary}%
5476     {\tablehead{\hline\bfseries \entryname &
5477         \bfseries \descriptionname\tabularnewline\hline}%
5478     \tabletail{\hline}
5479     \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
5480     {\end{supertabular}}}%
5481 }

```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```

5482 \newglossarystyle{superragged3col}{%
    Put the glossary in a supertabular environment with three columns and no head
    or tail:
5483     \renewenvironment{theglossary}%
5484     {\tablehead{}\tabletail{}}%
5485     \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}%
5486         >{\raggedright}p{\glspagelistwidth}|}}%
5487     {\end{supertabular}}}%
    Do nothing at the start of the table:
5488     \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
5489 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
5490 \renewcommand*{\glossaryentryfield}[5]{%
5491 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
5492 \renewcommand*{\glossarysubentryfield}[6]{%
5493 &
5494 \glssubentryitem{##2}%
5495 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
5496 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \&\tabularnewline\fi}%
5497 }
```

superragged3colborder The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
5498 \newglossarystyle{superragged3colborder}{%
```

Base it on the `glostylesuperragged3col` style:

```
5499 \glossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
5500 \renewenvironment{theglossary}%
5501 {\tablehead{\hline}\tabletail{\hline}%
5502 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
5503 >{\raggedright}p{\glspagelistwidth}|}%
5504 {\end{supertabular}}%
5505 }
```

superragged3colheader The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
5506 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
5507 \glossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
5508 \renewenvironment{theglossary}%
5509 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5510 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
5511 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}%
5512 >{\raggedright}p{\glspagelistwidth}}}%
5513 {\end{supertabular}}%
5514 }
```

ght3colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
5515 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
5516 \glossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
5517 \renewenvironment{theglossary}{%
5518     {\tablehead{\hline
5519         \bfseries\entryname&\bfseries\descriptionname&
5520         \bfseries\pagelistname\tabularnewline\hline}%
5521     \tabletail{\hline}%
5522     \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
5523         >{\raggedright}p{\glspagelistwidth}|}%
5524     {\end{supertabular}}}%
5525 }
```

altsuperragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
5526 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
5527 \renewenvironment{theglossary}{%
5528     {\tablehead{}\tabletail{}}%
5529     \begin{supertabular}{|l>{\raggedright}p{\glsgdescwidth}l%
5530         >{\raggedright}p{\glspagelistwidth}|}%
5531     {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
5532 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5533 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row with the name in the first column, description in second column, symbol in third column and page list in last column:

```
5534 \renewcommand*{\glossaryentryfield}[5]{%
5535     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
5536 \renewcommand*{\glossarysubentryfield}[6]{%
5537     &
5538     \glssubentryitem{##2}%
5539     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%

```

Blank row between groups:

```
5540 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & & \tabularnewline\fi}%
5541 }
```

`altsuperragged4colheader` The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```
5542 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
5543 \glossarystyle{altsuperragged4col}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
5544 \renewenvironment{theglossary}%
5545   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5546     \bfseries\symbolname &
5547     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
5548   \begin{supertabular}{1>{\raggedright}p{\glsgdescwidth}1%
5549     >{\raggedright}p{\glspagelistwidth}}}%
5550   {\end{supertabular}}%
5551 }
```

`altsuperragged4colborder` The `altsuperragged4colborder` style is like the `altsuperragged4col` style but with a border.

```
5552 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
5553 \glossarystyle{altsuper4col}%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
5554 \renewenvironment{theglossary}%
5555   {\tablehead{\hline}\tabletail{\hline}%
5556   \begin{supertabular}%
5557     {|1|>{\raggedright}p{\glsgdescwidth}|1|}%
5558     >{\raggedright}p{\glspagelistwidth}|}%
5559   {\end{supertabular}}%
5560 }
```

`altsuperragged4colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
5561 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
5562 \glossarystyle{altsuperragged4col}%
```

Put the glossary in a `supertabular` environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
5563 \renewenvironment{theglossary}%
5564   {\tablehead{\hline
5565     \bfseries\entryname &
5566     \bfseries\descriptionname &
5567     \bfseries\symbolname &
5568     \bfseries\pagelistname\tabularnewline\hline}%
5569   {\end{supertabular}}%
```

```

5569 \tabletail{\hline}%
5570 \begin{supertabular}%
5571     {\l|>\raggedright}p{\glsdescwidth}|l|%
5572     >\raggedright}p{\glspagelistwidth}|}%
5573 {\end{supertabular}}%
5574 }

```

3.9 Tree Styles (glossary-tree.sty)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```

5575 \ProvidesPackage{glossary-tree}[2012/09/21 v3.03 (NLCT)]

```

index The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```

5576 \newglossarystyle{index}{%

```

Set the paragraph indentation and skip and define `\item` to be the same as that used by the index:

```

5577 \renewenvironment{theglossary}%
5578     {\setlength{\parindent}{0pt}%
5579     \setlength{\parskip}{0pt plus 0.3pt}%
5580     \let\item\@idxitem}%
5581     {}%

```

Do nothing at the start of the environment:

```

5582 \renewcommand*{\glossaryheader}{}%

```

No group headers:

```

5583 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entry starts a new item with the name in bold followed by the symbol in brackets (if it exists), the description and the page list.

```

5584 \renewcommand*{\glossaryentryfield}[5]{%
5585 \item\glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
5586 \ifx\relax##4\relax
5587 \else
5588 \space{##4}%
5589 \fi
5590 \space ##3\glspostdescription \space ##5}%

```

Sub entries: level 1 entries use `\subitem`, levels greater than 1 use `\subsubitem`. The level (##1) shouldn't be 0, as that's catered by `\glossaryentryfield`, but for completeness, if the level is 0, `\item` is used. The name is put in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

5591 \renewcommand*{\glossarysubentryfield}[6]{%
5592 \ifcase##1\relax
5593 % level 0

```

```

5594     \item
5595   \or
5596     % level 1
5597     \subitem
5598     \glssubentryitem{##2}%
5599   \else
5600     % all other levels
5601     \subsubitem
5602   \fi
5603   \textbf{\glstarget{##2}{##3}}%
5604   \ifx\relax##5\relax
5605   \else
5606     \space{##5}%
5607   \fi
5608   \space##4\glspostdescription\space ##6}%

```

Vertical gap between groups is the same as that used by indices:

```

5609   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}

```

indexgroup The indexgroup style is like the index style but has headings.

```

5610 \newglossarystyle{indexgroup}{%

```

Base it on the glostyleindex style:

```

5611   \glossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

5612   \renewcommand*{\glsgroupheading}[1]{%
5613     \item\textbf{\glsgrouptitle{##1}}\indexspace}%
5614 }

```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```

5615 \newglossarystyle{indexhypergroup}{%

```

Base it on the glostyleindex style:

```

5616   \glossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

5617   \renewcommand*{\glossaryheader}{%
5618     \item\textbf{\glsnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

5619   \renewcommand*{\glsgroupheading}[1]{%
5620     \item\textbf{\glsnavigation\hypertarget{##1}{\glsgrouptitle{##1}}}%
5621     \indexspace}%
5622 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

5623 \newglossarystyle{tree}{%

```


Set the paragraph indentation and skip:

```
5624 \renewenvironment{theglossary}%
5625     {\setlength{\parindent}{0pt}%
5626      \setlength{\parskip}{0pt plus 0.3pt}}%
5627     {}%
```

Do nothing at the start of the theglossary environment:

```
5628 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5629 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
5630 \renewcommand{\glossaryentryfield}[5]{%
5631     \hangindent0pt\relax
5632     \parindent0pt\relax
5633     \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
5634     \ifx\relax##4\relax
5635     \else
5636         \space{##4}%
5637     \fi
5638     \space ##3\glspostdescription \space ##5\par}%
```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
5639 \renewcommand{\glossarysubentryfield}[6]{%
5640     \hangindent##1\glstreeindent\relax
5641     \parindent##1\glstreeindent\relax
5642     \ifnum##1=1\relax
5643         \glssubentryitem{##2}%
5644     \fi
5645     \textbf{\glstarget{##2}{##3}}%
5646     \ifx\relax##5\relax
5647     \else
5648         \space{##5}%
5649     \fi
5650     \space##4\glspostdescription\space ##6\par}%
```

Vertical gap between groups is the same as that used by indices:

```
5651 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```
5652 \newglossarystyle{treegroup}{%
```

Base it on the glostyletree style:

```
5653 \glossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
5654 \renewcommand{\glsgroupheading}[1]{\par
```

```

5655 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
5656 }

```

treehypergroup The treehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

5657 \newglossarystyle{treehypergroup}{%

```

Base it on the glostyletree style:

```

5658 \glossarystyle{tree}%

```

Put navigation links to the groups at the start of the theglossary environment:

```

5659 \renewcommand*{\glossaryheader}{%
5660 \par\noindent\textbf{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

5661 \renewcommand*{\glsgroupheading}[1]{%
5662 \par\noindent
5663 \textbf{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
5664 \indexspace}%
5665 }

```

\glstreeindent Length governing left indent for each level of the tree style.

```

5666 \newlength\glstreeindent
5667 \setlength{\glstreeindent}{10pt}

```

treenoname The treenoname glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

5668 \newglossarystyle{treenoname}{%

```

Set the paragraph indentation and skip:

```

5669 \renewenvironment{theglossary}%
5670 {\setlength{\parindent}{0pt}%
5671 \setlength{\parskip}{0pt plus 0.3pt}}%
5672 {}%

```

No header:

```

5673 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

5674 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: the name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

5675 \renewcommand{\glossaryentryfield}[5]{%
5676 \hangindent0pt\relax
5677 \parindent0pt\relax
5678 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
5679 \ifx\relax##4\relax
5680 \else
5681 \space{##4}%
5682 \fi
5683 \space ##3\glspostdescription \space ##5\par}%

```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name and symbol are omitted. The description followed by the page list are displayed.

```
5684 \renewcommand{\glossarysubentryfield}[6]{%
5685   \hangindent##1\glstreeindent\relax
5686   \parindent##1\glstreeindent\relax
5687   \ifnum##1=1\relax
5688     \glssubentryitem{##2}%
5689   \fi
5690   \glstarget{##2}{\strut}%
5691   ##4\glspostdescription\space ##6\par}%
```

Vertical gap between groups is the same as that used by indices:

```
5692 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
5693 }
```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```
5694 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
5695 \glossarystyle{treenoname}%
```

Give each group a heading:

```
5696 \renewcommand{\glsgroupheading}[1]{\par
5697   \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
5698 }
```

treenonamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
5699 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
5700 \glossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
5701 \renewcommand*{\glossaryheader}{%
5702   \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
5703 \renewcommand*{\glsgroupheading}[1]{%
5704   \par\noindent
5705   \textbf{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
5706   \indexspace}%
5707 }
```

\glissetwidest `\glissetwidest[$\langle level \rangle$]{ $\langle text \rangle$ }` sets the widest text for the given level. It is used by the `alttree` glossary styles to determine the indentation of each level.

```
5708 \newcommand*{\glissetwidest}[2][0]{%
5709   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
5710     #2}%
5711 }
```

```

\@glswidestname Initialise \@glswidestname.
5712 \newcommand*{\@glswidestname}{%

almtree The almtree glossary style is similar in style to the tree style, but the inden-
        tation is obtained from the width of \@glswidestname which is set using
        \glsetwidest.
5713 \newglossarystyle{almtree}{%
    Redefine theglossary environment.
5714 \renewenvironment{theglossary}%
5715     {\def\@gls@prevlevel{-1}%
5716      \mbox{}\par}%
5717     {\par}%

    Set the header and group headers to nothing.
5718 \renewcommand*{\glossaryheader}{}%
5719 \renewcommand*{\glsgroupheading}[1]{}%

    Redefine the way that the level 0 entries are displayed.
5720 \renewcommand{\glossaryentryfield}[5]{%

        If the level hasn't changed, keep the same settings, otherwise change \glstreeindent
        accordingly.
5721 \ifnum\@gls@prevlevel=0\relax
5722 \else

        Find out how big the indentation should be by measuring the widest entry.
5723 \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%

        Set the hangindent and paragraph indent.
5724 \hangindent\glstreeindent
5725 \parindent\glstreeindent
5726 \fi

        Put the name to the left of the paragraph block.
5727 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
5728 \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}}%

        If the symbol is missing, ignore it, otherwise put it in brackets.
5729 \ifx\relax##4\relax
5730 \else
5731 (##4)\space
5732 \fi

        Do the description followed by the description terminator and location list.
5733 ##3\glspostdescription \space ##5\par

        Set the previous level to 0.
5734 \def\@gls@prevlevel{0}%
5735 }%

        Redefine the way sub-entries are displayed.
5736 \renewcommand{\glossarysubentryfield}[6]{%

```

Increment and display the sub-entry counter if this is a level 1 entry and the sub-entry counter is in use.

```
5737 \ifnum##1=1\relax
5738 \glssubentryitem{##2}%
5739 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
5740 \ifnum\@gls@prevlevel=##1\relax
5741 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.

Store in `\gls@tmplen`

```
5742 \@ifundefined{@glswidestname\romannumeral##1}{%
5743 \settowidth{\gls@tmplen}{\textbf{@glswidestname\space}}}%
5744 \settowidth{\gls@tmplen}{\textbf{%
5745 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
5746 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
5747 \setlength\glstreeindent\gls@tmplen
5748 \addtolength\glstreeindent\parindent
5749 \parindent\glstreeindent
5750 \else
```

Depth has decreased, so subtract width of the widest entry from the previous level to `\glstreeindent`. First determine the width of the widest entry for the previous level and store in `\glstreeindent`.

```
5751 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
5752 \settowidth{\glstreeindent}{\textbf{%
5753 @glswidestname\space}}}%
5754 \settowidth{\glstreeindent}{\textbf{%
5755 \csname @glswidestname\romannumeral\@gls@prevlevel
5756 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
5757 \addtolength\parindent{-\glstreeindent}%
5758 \setlength\glstreeindent\parindent
5759 \fi
5760 \fi
```

Set the hanging indentation.

```
5761 \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
5762 \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
5763 \textbf{\glstarget{##2}{##3}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
5764 \ifx##5\relax\relax
5765 \else
5766   (##5)\space
5767 \fi
```

Do the description followed by the description terminator and location list.

```
5768 ##4\glspostdescription\space ##6\par
```

Set the previous level macro to the current level.

```
5769 \def\@gls@prevlevel{##1}%
5770 }%
```

Vertical gap between groups is the same as that used by indices:

```
5771 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
5772 }
```

alttreegroup Like the alttree style but the glossary groups have headings.

```
5773 \newglossarystyle{alttreegroup}{%
```

Base it on the glostylealttree style:

```
5774 \glossarystyle{alttree}%
```

Give each group a heading.

```
5775 \renewcommand{\glsgroupheading}[1]{\par
5776 \def\@gls@prevlevel{-1}%
5777 \hangindent0pt\relax
5778 \parindent0pt\relax
5779 \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
5780 }
```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```
5781 \newglossarystyle{alttreehypergroup}{%
```

Base it on the glostylealttree style:

```
5782 \glossarystyle{alttree}%
```

Put the navigation links in the header

```
5783 \renewcommand*{\glossaryheader}{%
5784 \par
5785 \def\@gls@prevlevel{-1}%
5786 \hangindent0pt\relax
5787 \parindent0pt\relax
5788 \textbf{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
5789 \renewcommand*{\glsgroupheading}[1]{%
5790 \par
5791 \def\@gls@prevlevel{-1}%
5792 \hangindent0pt\relax
5793 \parindent0pt\relax
```

```

5794 \textbf{\glsnabhypertarget{##1}{\glsgrouptitle{##1}}}\par
5795 \indexspace}}

```

4 glossaries-compatible-207

Provides compatibility with version 2.07 and below. This uses original glossaries xindy and makeindex formatting, so can be used with old documents that had customized style files, but hyperlinks may not work properly.

```

5796 \NeedsTeXFormat{LaTeX2e}
5797 \ProvidesPackage{glossaries-compatible-207}[2011/04/02 v1.0 (NLCT)]

```

`\GlsAddXdyAttribute` Adds an attribute in old format.

```

5798 \ifglsxindy
5799 \renewcommand*\GlsAddXdyAttribute[1]{%
5800 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
5801 \expandafter\toks@\expandafter{\@xdylocref}%
5802 \edef\@xdylocref{\the\toks@ ^^J%
5803 (markup-locref
5804 :open \string"\string~n\string\setentrycounter
5805 {\noexpand\glscounter}%
5806 \expandafter\string\csname#1\endcsname
5807 \expandafter\@gobble\string\{\string" ^^J
5808 :close \string"\expandafter\@gobble\string\}\string" ^^J
5809 :attr \string"#1\string"))}

```

Only has an effect before `\writeist`:

```

5810 \fi

```

`\GlsAddXdyCounters`

```

5811 \renewcommand*\GlsAddXdyCounters[1]{%
5812 \GlossariesWarning{\string\GlsAddXdyCounters\space not available
5813 in compatibility mode.}%
5814 }

```

Add predefined attributes

```

5815 \GlsAddXdyAttribute{glnumberformat}
5816 \GlsAddXdyAttribute{textrm}
5817 \GlsAddXdyAttribute{textsf}
5818 \GlsAddXdyAttribute{texttt}
5819 \GlsAddXdyAttribute{textbf}
5820 \GlsAddXdyAttribute{textmd}
5821 \GlsAddXdyAttribute{textit}
5822 \GlsAddXdyAttribute{textup}
5823 \GlsAddXdyAttribute{textsl}
5824 \GlsAddXdyAttribute{textsc}
5825 \GlsAddXdyAttribute{emph}
5826 \GlsAddXdyAttribute{glshypernumber}
5827 \GlsAddXdyAttribute{hyperrrm}

```

```

5828 \GlsAddXdyAttribute{hypersf}
5829 \GlsAddXdyAttribute{hypertt}
5830 \GlsAddXdyAttribute{hyperbf}
5831 \GlsAddXdyAttribute{hypermd}
5832 \GlsAddXdyAttribute{hyperit}
5833 \GlsAddXdyAttribute{hyperup}
5834 \GlsAddXdyAttribute{hypersl}
5835 \GlsAddXdyAttribute{hypersc}
5836 \GlsAddXdyAttribute{hyperemph}

```

\GlsAddXdyLocation Restore v2.07 definition:

```

5837 \ifglxsindy
5838   \renewcommand*{\GlsAddXdyLocation}[2]{%
5839     \edef\@xdyuserlocationdefs{%
5840       \@xdyuserlocationdefs ^^J%
5841       (define-location-class \string"#1\string"^^J\space\space
5842       \space(#2))
5843     }%
5844     \edef\@xdyuserlocationnames{%
5845       \@xdyuserlocationnames^^J\space\space\space
5846       \string"#1\string"}%
5847   }
5848 \fi

```

\@do@wrglossary

```

5849 \renewcommand{\@do@wrglossary}[1]{%

```

Determine whether to use xindy or makeindex syntax

```

5850 \ifglxsindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5851 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5852 \def\@glo@range{}%
5853 \expandafter\if\@glo@prefix(\relax
5854   \def\@glo@range{:open-range}%
5855 \else
5856   \expandafter\if\@glo@prefix)\relax
5857   \def\@glo@range{:close-range}%
5858 \fi
5859 \fi

```

Get the location and escape any special characters

```

5860 \protected@edef\@glslocref{\theglentrycounter}%
5861 \@gls@checkmkidxchars\@glslocref

```

Write to the glossary file using xindy syntax.

```

5862 \glossary[\csname glo@#1@type\endcsname]{%
5863 (indexentry :tkey (\csname glo@#1@index\endcsname)
5864   :locref \string"\@glslocref\string" %

```



```

5865      :attr \string"\@glo@suffix\string" \@glo@range
5866    )
5867  }%
5868 \else

```

Convert the format information into the format required for makeindex

```

5869 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

5870 \glossary[\csname glo@#1@type\endcsname]{%
5871 \string\glossaryentry{\csname glo@#1@index\endcsname
5872 \@gls@encapchar\@glo@numfmt}\theglsentrycounter}}%
5873 \fi
5874 }

```

\@set@glo@numformat Only had 3 arguments in v2.07

```

5875 \def\@set@glo@numformat#1#2#3{%
5876 \expandafter\@glo@check@mkidxrangechar#3\@nil
5877 \protected@edef#1{%
5878 \@glo@prefix setentrycounter[] {#2}%
5879 \expandafter\string\csname\@glo@suffix\endcsname
5880 }%
5881 \@gls@checkmkidxchars#1%
5882 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

5883 \ifglxindy
5884 \def\writeist{%
5885 \openout\glswrite=\istfilename
5886 \write\glswrite{;; xindy style file created by the glossaries
5887 package in compatible-2.07 mode}%
5888 \write\glswrite{;; for document '\jobname' on
5889 \the\year-\the\month-\the\day}%
5890 \write\glswrite{^^J; required styles^^J}
5891 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
5892 \ifx\@xdystyle\@empty
5893 \else
5894 \protected@write\glswrite{{(require
5895 \string"\@xdystyle.xdy\string")}}%
5896 \fi
5897 }%
5898 \write\glswrite{^^J%
5899 ; list of allowed attributes (number formats)^^J}%
5900 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
5901 \write\glswrite{^^J; user defined alphabets^^J}%
5902 \write\glswrite{\@xdyuseralphabets}%
5903 \write\glswrite{^^J; location class definitions^^J}%
5904 \protected@edef\@gls@roman{\@roman{0}\string"
5905 \string"roman-numbers-lowercase\string" :sep \string"}}%

```

```

5906 \@onelevel@sanitize\@gls@roman
5907 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
5908 :sep \string"}%
5909 \@onelevel@sanitize\@tmp
5910 \ifx\@tmp\@gls@roman
5911 \write\glswrite{(define-location-class
5912 \string"roman-page-numbers\string"^^J\space\space\space
5913 (\string"roman-numbers-lowercase\string")
5914 :min-range-length \@glsminrange)}}%
5915 \else
5916 \write\glswrite{(define-location-class
5917 \string"roman-page-numbers\string"^^J\space\space\space
5918 (:sep "\@gls@roman")
5919 :min-range-length \@glsminrange)}}%
5920 \fi
5921 \write\glswrite{(define-location-class
5922 \string"Roman-page-numbers\string"^^J\space\space\space
5923 (\string"roman-numbers-uppercase\string")
5924 :min-range-length \@glsminrange)}}%
5925 \write\glswrite{(define-location-class
5926 \string"arabic-page-numbers\string"^^J\space\space\space
5927 (\string"arabic-numbers\string")
5928 :min-range-length \@glsminrange)}}%
5929 \write\glswrite{(define-location-class
5930 \string"alpha-page-numbers\string"^^J\space\space\space
5931 (\string"alpha\string")
5932 :min-range-length \@glsminrange)}}%
5933 \write\glswrite{(define-location-class
5934 \string"Alpha-page-numbers\string"^^J\space\space\space
5935 (\string"ALPHA\string")
5936 :min-range-length \@glsminrange)}}%
5937 \write\glswrite{(define-location-class
5938 \string"Appendix-page-numbers\string"^^J\space\space\space
5939 (\string"ALPHA\string"
5940 :sep \string"\@glsAlphacompositor\string"
5941 \string"arabic-numbers\string")
5942 :min-range-length \@glsminrange)}}%
5943 \write\glswrite{(define-location-class
5944 \string"arabic-section-numbers\string"^^J\space\space\space
5945 (\string"arabic-numbers\string"
5946 :sep \string"\glscompositor\string"
5947 \string"arabic-numbers\string")
5948 :min-range-length \@glsminrange)}}%
5949 \write\glswrite{^^J; user defined location classes}%
5950 \write\glswrite{\@xdyuserlocationdefs}%
5951 \write\glswrite{^^J; define cross-reference class^^J}%
5952 \write\glswrite{(define-crossref-class \string"see\string"
5953 :unverified )}%
5954 \write\glswrite{(markup-crossref-list

```

```

5955      :class \string"see\string"^^J\space\space\space
5956      :open \string"\string\glseeformat\string"
5957      :close \string"{}\string")}}%
5958 \write\glswrite{^^J; define the order of the location classes}%
5959 \write\glswrite{(define-location-class-order
5960   (\@xdylocationclassorder))}%
5961 \write\glswrite{^^J; define the glossary markup^^J}%
5962 \write\glswrite{(markup-index^^J\space\space\space
5963   :open \string"\string
5964     \glossarysection[\string\glossarytoctitle]{\string
5965     \glossarytitle}\string\glossarypreamble\string~n\string\begin
5966     {theglossary}\string\glossaryheader\string~n\string" ^^J\space
5967     \space\space:close \string"\expandafter\@gobble
5968     \string%\string~n\string
5969     \end{theglossary}\string\glossarypostamble
5970     \string~n\string" ^^J\space\space\space
5971     :tree)}}%
5972 \write\glswrite{(markup-letter-group-list
5973   :sep \string"\string\glsgroupskip\string~n\string")}}%
5974 \write\glswrite{(markup-indexentry
5975   :open \string"\string\relax \string\glresetentrylist
5976     \string~n\string")}}%
5977 \write\glswrite{(markup-locclass-list :open
5978   \string"\glsoopenbrace\string\glossaryentrynumbers
5979   \glsoopenbrace\string\relax\space \string"^^J\space\space\space
5980   :sep \string", \string"
5981   :close \string"\glsclosebrace\glsclosebrace\string")}}%
5982 \write\glswrite{(markup-locref-list
5983   :sep \string"\string\delimN\space\string")}}%
5984 \write\glswrite{(markup-range
5985   :sep \string"\string\delimR\space\string")}}%
5986 \@onelevel@sanitize\gls@suffixF
5987 \@onelevel@sanitize\gls@suffixFF
5988 \ifx\gls@suffixF\@empty
5989 \else
5990   \write\glswrite{(markup-range
5991     :close "\gls@suffixF" :length 1 :ignore-end)}}%
5992 \fi
5993 \ifx\gls@suffixFF\@empty
5994 \else
5995   \write\glswrite{(markup-range
5996     :close "\gls@suffixFF" :length 2 :ignore-end)}}%
5997 \fi
5998 \write\glswrite{^^J; define format to use for locations^^J}%
5999 \write\glswrite{\@xdylocref}%
6000 \write\glswrite{^^J; define letter group list format^^J}%
6001 \write\glswrite{(markup-letter-group-list
6002   :sep \string"\string\glsgroupskip\string~n\string")}}%
6003 \write\glswrite{^^J; letter group headings^^J}%

```

```

6004 \write\glswrite{(markup-letter-group
6005   :open-head \string"\string\glsgroupheading
6006   \glsoopenbrace\string"^^J\space\space\space
6007   :close-head \string"\glsclosebrace\string"))}%
6008 \write\glswrite{^^J; additional letter groups^^J}%
6009 \write\glswrite{\@xdylettergroups}%
6010 \write\glswrite{^^J; additional sort rules^^J}
6011 \write\glswrite{\@xdysortrules}%
6012 \noist}
6013 \else
6014 \edef\@gls@actualchar{\string?}
6015 \edef\@gls@encapchar{\string|}
6016 \edef\@gls@levelchar{\string!}
6017 \edef\@gls@quotechar{\string"}
6018 \def\writeist{\relax
6019   \openout\glswrite=\istfilename
6020   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
6021     created by the glossaries package}
6022   \write\glswrite{\expandafter\@gobble\string\% for document
6023     '\jobname' on \the\year-\the\month-\the\day}
6024   \write\glswrite{actual '\@gls@actualchar'}
6025   \write\glswrite{encap '\@gls@encapchar'}
6026   \write\glswrite{level '\@gls@levelchar'}
6027   \write\glswrite{quote '\@gls@quotechar'}
6028   \write\glswrite{keyword \string"\string\glossaryentry\string"}
6029   \write\glswrite{preamble \string"\string\glossarysection[\string
6030     \glossarytoctitle]{\string\glossarytitle}\string
6031     \glossarypreamble\string\n\string\begin{theglossary}\string
6032     \glossaryheader\string\n\string"}
6033   \write\glswrite{postamble \string"\string\%\string\n\string
6034     \end{theglossary}\string\glossarypostamble\string\n
6035     \string"}
6036   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
6037     \string"}
6038   \write\glswrite{item_0 \string"\string\%\string\n\string"}
6039   \write\glswrite{item_1 \string"\string\%\string\n\string"}
6040   \write\glswrite{item_2 \string"\string\%\string\n\string"}
6041   \write\glswrite{item_01 \string"\string\%\string\n\string"}
6042   \write\glswrite{item_x1
6043     \string"\string\relax \string\glsresetentrylist\string\n
6044     \string"}
6045   \write\glswrite{item_12 \string"\string\%\string\n\string"}
6046   \write\glswrite{item_x2
6047     \string"\string\relax \string\glsresetentrylist\string\n
6048     \string"}
6049   \write\glswrite{delim_0 \string"\string\{\string
6050     \glossaryentrynumbers\string\{\string\relax \string"}
6051   \write\glswrite{delim_1 \string"\string\{\string
6052     \glossaryentrynumbers\string\{\string\relax \string"}

```

```

6053 \write\glswrite{delim_2 \string"\string\{\string
6054 \glossaryentrynumbers\string\{\string\relax \string"}
6055 \write\glswrite{delim_t \string"\string\}\string\}\string"}
6056 \write\glswrite{delim_n \string"\string\delimN \string"}
6057 \write\glswrite{delim_r \string"\string\delimR \string"}
6058 \write\glswrite{headings_flag 1}
6059 \write\glswrite{heading_prefix
6060 \string"\string\glsgroupheading\string\{\string"}
6061 \write\glswrite{heading_suffix
6062 \string"\string\}\string\relax
6063 \string\glsresetentrylist \string"}
6064 \write\glswrite{symhead_positive \string"glssymbols\string"}
6065 \write\glswrite{numhead_positive \string"glsglnumbers\string"}
6066 \write\glswrite{page_compositor \string"glsglcompositor\string"}
6067 \@gls@escbsdq\gls@suffixF
6068 \@gls@escbsdq\gls@suffixFF
6069 \ifx\gls@suffixF\@empty
6070 \else
6071 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
6072 \fi
6073 \ifx\gls@suffixFF\@empty
6074 \else
6075 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
6076 \fi
6077 \noist
6078 }
6079 \fi

```

\noist

```

6080 \renewcommand*{\noist}{\let\writeist\relax}

```

5 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibility support in glossary entries. See the documentation for further details about accessibility support.

```

6081 \NeedsTeXFormat{LaTeX2e}

```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```

6082 \ProvidesPackage{glossaries-accsupp}[2011/04/02 v3.0 (NLCT)
6083 Experimental glossaries accessibility]

```

Pass all options to glossaries:

```

6084 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}

```

Process options:

```

6085 \ProcessOptions

```

Required packages:

```
6086 \RequirePackage{glossaries}
6087 \RequirePackage{accsupp}
```

5.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}
```

access The replacement text corresponding to the name key:

```
6088 \define@key{glossentry}{access}{%
6089   \def\@glo@access{#1}%
6090 }
```

textaccess The replacement text corresponding to the text key:

```
6091 \define@key{glossentry}{textaccess}{%
6092   \def\@glo@textaccess{#1}%
6093 }
```

firstaccess The replacement text corresponding to the first key:

```
6094 \define@key{glossentry}{firstaccess}{%
6095   \def\@glo@firstaccess{#1}%
6096 }
```

pluralaccess The replacement text corresponding to the plural key:

```
6097 \define@key{glossentry}{pluralaccess}{%
6098   \def\@glo@pluralaccess{#1}%
6099 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
6100 \define@key{glossentry}{firstpluralaccess}{%
6101   \def\@glo@firstpluralaccess{#1}%
6102 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
6103 \define@key{glossentry}{symbolaccess}{%
6104   \def\@glo@symbolaccess{#1}%
6105 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
6106 \define@key{glossentry}{symbolpluralaccess}{%
6107   \def\@glo@symbolpluralaccess{#1}%
6108 }
```

descriptionaccess The replacement text corresponding to the description key:

```

6109 \define@key{glossentry}{descriptionaccess}{%
6110   \def\@glo@descaccess{#1}%
6111 }

```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

6112 \define@key{glossentry}{descriptionpluralaccess}{%
6113   \def\@glo@descpluralaccess{#1}%
6114 }

```

shortaccess The replacement text corresponding to the short key:

```

6115 \define@key{glossentry}{shortaccess}{%
6116   \def\@glo@shortaccess{#1}%
6117 }

```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

6118 \define@key{glossentry}{shortpluralaccess}{%
6119   \def\@glo@shortpluralaccess{#1}%
6120 }

```

longaccess The replacement text corresponding to the long key:

```

6121 \define@key{glossentry}{longaccess}{%
6122   \def\@glo@longaccess{#1}%
6123 }

```

longpluralaccess The replacement text corresponding to the longplural key:

```

6124 \define@key{glossentry}{longpluralaccess}{%
6125   \def\@glo@longpluralaccess{#1}%
6126 }

```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacsupp{inches}{in}}.

\@gls@noaccess Indicates that no replacement text has been provided.

```

6127 \def\@gls@noaccess{\relax}

```

Add to the start hook (the access key is initialised to the value of the symbol key at the start for backwards compatibility):

```

6128 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
6129 \renewcommand*{\@newglossaryentryprehook}{%
6130   \@gls@oldnewglossaryentryprehook
6131   \def\@glo@access{\@glo@symbol}%

```

Initialise the other keys:

```

6132   \def\@glo@textaccess{\@glo@access}%
6133   \def\@glo@firstaccess{\@glo@access}%
6134   \def\@glo@pluralaccess{\@glo@textaccess}%
6135   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%

```

```

6136 \def\@glo@symbolaccess{\relax}%
6137 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
6138 \def\@glo@descaccess{\relax}%
6139 \def\@glo@descpluralaccess{\@glo@descaccess}%
6140 \def\@glo@shortaccess{\relax}%
6141 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
6142 \def\@glo@longaccess{\relax}%
6143 \def\@glo@longpluralaccess{\@glo@longaccess}%
6144 }

```

Add to the end hook:

```

6145 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
6146 \renewcommand*{\@newglossaryentryposthook}{%
6147 \@gls@oldnewglossaryentryposthook

```

Store the access information:

```

6148 \expandafter
6149 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
6150 \@glo@access}%
6151 \expandafter
6152 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
6153 \@glo@textaccess}%
6154 \expandafter
6155 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
6156 \@glo@firstaccess}%
6157 \expandafter
6158 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
6159 \@glo@pluralaccess}%
6160 \expandafter
6161 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
6162 \@glo@firstpluralaccess}%
6163 \expandafter
6164 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
6165 \@glo@symbolaccess}%
6166 \expandafter
6167 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
6168 \@glo@symbolpluralaccess}%
6169 \expandafter
6170 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
6171 \@glo@descaccess}%
6172 \expandafter
6173 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
6174 \@glo@descpluralaccess}%
6175 \expandafter
6176 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
6177 \@glo@shortaccess}%
6178 \expandafter
6179 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
6180 \@glo@shortpluralaccess}%
6181 \expandafter

```



```

6182 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
6183 \@glo@longaccess}%
6184 \expandafter
6185 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
6186 \@glo@longpluralaccess}%
6187 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

6188 \newcommand*\glsentryaccess}[1]{%
6189 \csname glo@#1@access\endcsname
6190 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

6191 \newcommand*\glsentrytextaccess}[1]{%
6192 \csname glo@#1@textaccess\endcsname
6193 }

```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```

6194 \newcommand*\glsentryfirstaccess}[1]{%
6195 \csname glo@#1@firstaccess\endcsname
6196 }

```

`\glsentrypluralaccess` Get the value of the pluralaccess key for the entry with the given label:

```

6197 \newcommand*\glsentrypluralaccess}[1]{%
6198 \csname glo@#1@pluralaccess\endcsname
6199 }

```

`\glsentryfirstpluralaccess` Get the value of the firstpluralaccess key for the entry with the given label:

```

6200 \newcommand*\glsentryfirstpluralaccess}[1]{%
6201 \csname glo@#1@firstpluralaccess\endcsname
6202 }

```

`\glsentrysymbolaccess` Get the value of the symbolaccess key for the entry with the given label:

```

6203 \newcommand*\glsentrysymbolaccess}[1]{%
6204 \csname glo@#1@symbolaccess\endcsname
6205 }

```

`\glsentrysymbolpluralaccess` Get the value of the symbolpluralaccess key for the entry with the given label:

```

6206 \newcommand*\glsentrysymbolpluralaccess}[1]{%
6207 \csname glo@#1@symbolpluralaccess\endcsname
6208 }

```

`\glsentrydescaccess` Get the value of the descriptionaccess key for the entry with the given label:

```

6209 \newcommand*\glsentrydescaccess}[1]{%
6210 \csname glo@#1@descaccess\endcsname
6211 }

```

trydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
6212 \newcommand*{\glentrydescpluralaccess}[1]{%
6213   \csname glo@#1@descaccess\endcsname
6214 }
```

glentryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
6215 \newcommand*{\glentryshortaccess}[1]{%
6216   \csname glo@#1@shortaccess\endcsname
6217 }
```

ryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
6218 \newcommand*{\glentryshortpluralaccess}[1]{%
6219   \csname glo@#1@shortpluralaccess\endcsname
6220 }
```

\glentrylongaccess Get the value of the longaccess key for the entry with the given label:

```
6221 \newcommand*{\glentrylongaccess}[1]{%
6222   \csname glo@#1@longaccess\endcsname
6223 }
```

trylongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
6224 \newcommand*{\glentrylongpluralaccess}[1]{%
6225   \csname glo@#1@longpluralaccess\endcsname
6226 }
```

\glsaccsupp \glsaccsupp{<replacement text>}{<text>}

This can be redefined to use E or Alt instead of ActualText. (I don't have the software to test the E or Alt options.)

```
6227 \newcommand*{\glsaccsupp}[2]{%
6228   \BeginAccSupp{ActualText=#1}#2\EndAccSupp}%
6229 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
6230 \newcommand*{\xglsaccsupp}[2]{%
6231   \protected@edef\@gls@replacementtext{#1}%
6232   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
6233 }
```

lnameaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
6234 \DeclareRobustCommand*{\glnameaccessdisplay}[2]{%
6235   \protected@edef\@glo@access{\glentryaccess{#2}}%
6236   \ifx\@glo@access\@gls@noaccess
6237     #1%
6238   \else
6239     \xglsaccsupp{\@glo@access}{#1}%
6240   \fi}
```

```

6240 \fi
6241 }

```

`lsttextaccessdisplay` As above but for the `textaccess` replacement text.

```

6242 \DeclareRobustCommand*\glstextaccessdisplay}[2]{%
6243   \protected@edef\@glo@access{\glstentrytextaccess{#2}}%
6244   \ifx\@glo@access\@gls@noaccess
6245     #1%
6246   \else
6247     \xglsaccsupp{\@glo@access}{#1}%
6248   \fi
6249 }

```

`pluralaccessdisplay` As above but for the `pluralaccess` replacement text.

```

6250 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
6251   \protected@edef\@glo@access{\glsentrypluralaccess{#2}}%
6252   \ifx\@glo@access\@gls@noaccess
6253     #1%
6254   \else
6255     \xglsaccsupp{\@glo@access}{#1}%
6256   \fi
6257 }

```

`sfirstaccessdisplay` As above but for the `firstaccess` replacement text.

```

6258 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
6259   \protected@edef\@glo@access{\glsentryfirstaccess{#2}}%
6260   \ifx\@glo@access\@gls@noaccess
6261     #1%
6262   \else
6263     \xglsaccsupp{\@glo@access}{#1}%
6264   \fi
6265 }

```

`pluralaccessdisplay` As above but for the `firstpluralaccess` replacement text.

```

6266 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
6267   \protected@edef\@glo@access{\glsentryfirstpluralaccess{#2}}%
6268   \ifx\@glo@access\@gls@noaccess
6269     #1%
6270   \else
6271     \xglsaccsupp{\@glo@access}{#1}%
6272   \fi
6273 }

```

`symbolaccessdisplay` As above but for the `symbolaccess` replacement text.

```

6274 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
6275   \protected@edef\@glo@access{\glsentrysymbolaccess{#2}}%
6276   \ifx\@glo@access\@gls@noaccess
6277     #1%
6278   \else

```

```

6279 \xglsaccsupp{\@glo@access}{#1}%
6280 \fi
6281 }

```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

6282 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
6283 \protected@edef\@glo@access{\glstrysymbolpluralaccess{#2}}%
6284 \ifx\@glo@access\@gls@noaccess
6285 #1%
6286 \else
6287 \xglsaccsupp{\@glo@access}{#1}%
6288 \fi
6289 }

```

descriptionaccessdisplay As above but for the descriptionaccess replacement text.

```

6290 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
6291 \protected@edef\@glo@access{\glstrydescaccess{#2}}%
6292 \ifx\@glo@access\@gls@noaccess
6293 #1%
6294 \else
6295 \xglsaccsupp{\@glo@access}{#1}%
6296 \fi
6297 }

```

descriptionpluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

6298 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
6299 \protected@edef\@glo@access{\glstrydescpluralaccess{#2}}%
6300 \ifx\@glo@access\@gls@noaccess
6301 #1%
6302 \else
6303 \xglsaccsupp{\@glo@access}{#1}%
6304 \fi
6305 }

```

shortaccessdisplay As above but for the shortaccess replacement text.

```

6306 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
6307 \protected@edef\@glo@access{\glstryshortaccess{#2}}%
6308 \ifx\@glo@access\@gls@noaccess
6309 #1%
6310 \else
6311 \xglsaccsupp{\@glo@access}{#1}%
6312 \fi
6313 }

```

shortpluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

6314 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
6315 \protected@edef\@glo@access{\glstryshortpluralaccess{#2}}%
6316 \ifx\@glo@access\@gls@noaccess
6317 #1%

```

```

6318 \else
6319 \xglsaccsupp{\@glo@access}{#1}%
6320 \fi
6321 }

```

`\glslongaccessdisplay` As above but for the longaccess replacement text.

```

6322 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
6323 \protected@edef\@glo@access{\glsentrylongaccess{#2}}%
6324 \ifx\@glo@access\@gls@noaccess
6325 #1%
6326 \else
6327 \xglsaccsupp{\@glo@access}{#1}%
6328 \fi
6329 }

```

`\glspluralaccessdisplay` As above but for the longpluralaccess replacement text.

```

6330 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
6331 \protected@edef\@glo@access{\glsentrylongpluralaccess{#2}}%
6332 \ifx\@glo@access\@gls@noaccess
6333 #1%
6334 \else
6335 \xglsaccsupp{\@glo@access}{#1}%
6336 \fi
6337 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

6338 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
6339 \@ifundefined{gls#1accessdisplay}%
6340 {%
6341 \PackageError{glossaries-accsupp}{No accessibility support
6342 for key ‘#1’}{}%
6343 }%
6344 {%
6345 \csname gls#1accessdisplay\endcsname{#2}{#3}%
6346 }%
6347 }

```

`\@gls@` Redefine `\@gls@` to change the way the link text is defined

```

6348 \def\@gls@#1#2[#3]{%
6349 \glsdoifexists{#2}%
6350 {%
6351 \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6352 \def\@gls@link@opts{#1}%
6353 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text). This is no longer expanded.

```

6354 \ifglsused{#2}%
6355 {%
6356 \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6357 {\glstextaccessdisplay{\glentrytext{#2}}{#2}}%
6358 {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}%
6359 {\glssymbolaccessdisplay{\glentrysymbol{#2}}{#2}}%
6360 {#3}}%
6361 }%
6362 {%
6363 \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6364 {\glsfirstaccessdisplay{\glentryfirst{#2}}{#2}}%
6365 {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}%
6366 {\glssymbolaccessdisplay{\glentrysymbol{#2}}{#2}}%
6367 {#3}}%
6368 }%

```

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

```

6369 \ifglsused{#2}%
6370 {%
6371 \@gls@link[#1]{#2}{\@glo@text}%
6372 }%
6373 {%
6374 \gls@checkisacronymlist\@glo@type
6375 \ifthenelse{(\boolean{@glsisacronymlist})\AND
6376 \boolean{glsacrfootnote}}{\OR\NOT\boolean{glshyperfirst}}}%
6377 {%
6378 \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6379 }%
6380 {%
6381 \@gls@link[#1]{#2}{\@glo@text}%
6382 }%
6383 }%

```

Indicate that this entry has now been used

```

6384 \glsunset{#2}%
6385 }%
6386 }

```

\@Gls@

```

6387 \def\@Gls@#1#2[#3]{%
6388 \glsdoifexists{#2}%
6389 {%
6390 \edef\@glo@type{\glentrytype{#2}}%

```

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

```

6391 \def\@gls@link@opts{#1}%
6392 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text). The first character of the entry text is converted to uppercase before passing to \gls@<type>@display or \gls@<type>@displayfirst

```

6393   \ifglsused{#2}%
6394   {%
6395     \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6396       {\glsfirstaccessdisplay{\Glsentrytext{#2}}{#2}}%
6397       {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6398       {\glsymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6399       {#3}}%
6400   }%
6401   {%
6402     \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6403       {\glsfirstaccessdisplay{\Glsentryfirst{#2}}{#2}}%
6404       {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6405       {\glsymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6406       {#3}}%
6407   }%

```

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

```

6408   \ifglsused{#2}%
6409   {%
6410     \@gls@link[#1]{#2}{\@glo@text}%
6411   }%
6412   {%
6413     \gls@checkisacronymlist\@glo@type
6414     \ifthenelse{(\boolean{glsisacronymlist})\AND
6415       \boolean{glsacrfootnote}) \OR\NOT\boolean{glshyperfirst}}%
6416     {%
6417       \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6418     }%
6419     {%
6420       \@gls@link[#1]{#2}{\@glo@text}%
6421     }%
6422   }%

```

Indicate that this entry has now been used

```

6423   \glsunset{#2}%
6424   }%
6425 }

```

\@GLS@

```

6426 \def\@GLS@#1#2[#3]{%
6427   \glsdoifexists{#2}{%
6428     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6429   \def\@gls@link@opts{#1}%
6430   \def\@gls@link@label{#2}%

```

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

```

6431 \ifglsused{#2}%
6432 {%
6433 \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6434 {\gls@textaccessdisplay{\glsentrytext{#2}}{#2}}%
6435 {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6436 {\glsymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6437 {#3}}%
6438 }%
6439 {%
6440 \edef\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6441 {\glsfirstaccessdisplay{\glsentryfirst{#2}}{#2}}%
6442 {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6443 {\glsymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6444 {#3}}%
6445 }%

```

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

```

6446 \ifglsused{#2}%
6447 {%
6448 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6449 }%
6450 {%
6451 \gls@checkisacronymlist\@glo@type
6452 \ifthenelse{\boolean{glsisacronymlist}\AND
6453 \boolean{glsacrfootnote}}{\OR\NOT\boolean{gls@hyperfirst}}{%
6454 \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
6455 }%
6456 {%
6457 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6458 }%
6459 }%

```

Indicate that this entry has now been used

```

6460 \glsunset{#2}%
6461 }%
6462 }

```

\@gls@pl@

```

6463 \def\@glspl@#1#2[#3]{%
6464 \glsdoifexists{#2}%
6465 {%
6466 \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6467 \def\@gls@link@opts{#1}%
6468 \def\@gls@link@label{#2}%

```

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


```

6469 \ifglsused{#2}%
6470 {%
6471   \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6472     {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%
6473     {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6474     {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6475     {#3}}%
6476   }%
6477   {%
6478     \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6479       {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
6480       {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6481       {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6482       {#3}}%
6483     }%

```

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

```

6484 \ifglsused{#2}%
6485 {%
6486   \@gls@link[#1]{#2}{\@glo@text}%
6487 }%
6488 {%
6489   \gls@checkisacronymlist\@glo@type
6490   \ifthenelse{\boolean{glsisacronymlist}\AND
6491     \boolean{glsacrfootnote}}{\OR\NOT\boolean{glshyperfirst}}%
6492   {%
6493     \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6494   }%
6495   {%
6496     \@gls@link[#1]{#2}{\@glo@text}%
6497   }%
6498 }%

```

Indicate that this entry has now been used

```

6499 \glsunset{#2}%
6500 }%
6501 }

```

\@Glspl@

```

6502 \def\@Glspl@#1#2[#3]{%
6503   \glsdoifexists{#2}%
6504   {%
6505     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6506 \def\@gls@link@opts{#1}%
6507 \def\@gls@link@label{#2}%

```

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

```

6508 \ifglsused{#2}%

```

```

6509  {%
6510    \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6511      {\glspluralaccessdisplay{\Glsentryplural{#2}}{#2}}%
6512      {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6513      {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6514      {#3}}%
6515  }%
6516  {%
6517    \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6518      {\glsfirstpluralaccessdisplay{\Glsentryfirstplural{#2}}{#2}}%
6519      {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6520      {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6521      {#3}}%
6522  }%

```

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

```

6523  \ifglsused{#2}%
6524  {%
6525    \@gls@link[#1]{#2}{\@glo@text}%
6526  }%
6527  {%
6528    \ifthenelse{\equal{\@glo@type}{\acronymtype}\and
6529      \boolean{glsacrfootnote}}%
6530    {%
6531      \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6532    }%
6533    {%
6534      \@gls@link[#1]{#2}{\@glo@text}%
6535    }%
6536  }%

```

Indicate that this entry has now been used

```

6537  \glsunset{#2}%
6538  }%
6539 }

```

\@GLSpl@

```

6540 \def\@GLSpl@#1#2[#3]{%
6541   \glsdoifexists{#2}%
6542   {%
6543     \edef\@glo@type{\glsentrytype{#2}}%
6544     Save options in \@gls@link@opts and label in \@gls@link@label
6545     \def\@gls@link@opts{#1}%
6546     \def\@gls@link@label{#2}%
6547     Determine what the link text should be (this is stored in \@glo@text)
6548     \ifglsused{#2}%
6549     {%
6550       \def\@glo@text{\csname gls@\@glo@type @display\endcsname

```

```

6549      {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%
6550      {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6551      {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6552      {#3}}%
6553  }%
6554  {%
6555      \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6556      {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
6557      {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6558      {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6559      {#3}}%
6560  }%

```

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

```

6561      \ifglsused{#2}%
6562      {%
6563          \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6564      }%
6565      {%
6566          \gls@checkisacronymlist\@glo@type
6567          \ifthenelse{\(\boolean{@glsisacronymlist}\AND
6568              \boolean{glsacrfootnote})\OR\NOT\boolean{glshyperfirst}}%
6569              {%
6570                  \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
6571              }%
6572              {%
6573                  \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6574              }%
6575      }%

```

Indicate that this entry has now been used

```

6576      \glsunset{#2}%
6577  }%
6578 }

```

\@acrshort

```

6579 \def\@acrshort#1#2[#3]{%
6580     \glsdoifexists{#2}%
6581     {%
6582         \edef\@glo@type{\glsentrytype{#2}}%
6583         \def\@glo@text{%
6584             \glsshortaccessdisplay{\glsentryshort{#2}}{#2}%
6585         }%
6586         \def\@glo@text{%
6587             \glsshortaccessdisplay{\glsentryshort{#2}}{#2}%
6588         }%

```

Call \@gls@link

```

6586      \@gls@link[#1]{#2}{\acronymfont{\@glo@text}#3}%
6587  }%
6588 }

```

\@Acrshort

```
6589 \def\@Acrshort#1#2[#3]{%
6590   \glsdoifexists{#2}%
6591   {%
6592     \edef\@glo@type{\glsentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6593     \def\@glo@text{%
6594       \glsshortaccessdisplay{\Glsentryshort{#2}}{#2}%
6595     }%
        Call \@gls@link
6596     \@gls@link[#1]{#2}{\acronymfont{\@glo@text}#3}%
6597   }%
6598 }
```

\@ACRshort

```
6599 \def\@ACRshort#1#2[#3]{%
6600   \glsdoifexists{#2}%
6601   {%
6602     \edef\@glo@type{\glsentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6603     \def\@glo@text{%
6604       \glsshortaccessdisplay{\MakeUppercase{\glsentryshort{#2}}}{#2}%
6605     }%
        Call \@gls@link
6606     \@gls@link[#1]{#2}{\acronymfont{\@glo@text}#3}%
6607   }%
6608 }
```

\@acrlong

```
6609 \def\@acrlong#1#2[#3]{%
6610   \glsdoifexists{#2}%
6611   {%
6612     \edef\@glo@type{\glsentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6613     \def\@glo@text{%
6614       \glslongaccessdisplay{\glsentrylong{#2}}{#2}%
6615     }%
        Call \@gls@link
6616     \@gls@link[#1]{#2}{\@glo@text}#3}%
6617   }%
6618 }
```

\@Acrlong

```
6619 \def\@Acrlong#1#2[#3]{%
6620   \glsdoifexists{#2}%
```

```

6621  {%
6622    \edef\@glo@type{\glentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6623    \def\@glo@text{%
6624        \glslongaccessdisplay{\Glsentrylong{#2}}{#2}%
6625    }%
        Call \@gls@link
6626    \@gls@link[#1]{#2}{\@glo@text#3}%
6627  }%
6628 }

```

\@ACRlong

```

6629 \def\@ACRlong#1#2[#3]{%
6630   \glsdoifexists{#2}%
6631   {%
6632     \edef\@glo@type{\glentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6633     \def\@glo@text{%
6634         \glslongaccessdisplay{\MakeUppercase{\glentrylong{#2}}}{#2}%
6635     }%
        Call \@gls@link
6636     \@gls@link[#1]{#2}{\@glo@text#3}%
6637   }%
6638 }

```

5.3 Displaying the Glossary

Entries within the glossary or list of acronyms are now formatted via `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

@glossaryentryfield

```

6639 \ifglxindy
6640   \renewcommand*{\@glossaryentryfield}{%
6641     \string\accsuppglossaryentryfield}
6642 \else
6643   \renewcommand*{\@glossaryentryfield}{%
6644     \string\accsuppglossaryentryfield}
6645 \fi

```

ossarysubentryfield

```

6646 \ifglxindy
6647   \renewcommand*{\@glossarysubentryfield}{%
6648     \string\accsuppglossarysubentryfield}
6649 \else
6650   \renewcommand*{\@glossarysubentryfield}{%
6651     \string\accsuppglossarysubentryfield}
6652 \fi

```

pglossaryentryfield

```

6653 \newcommand*{\accsuppglossaryentryfield}[5]{%
6654   \glossaryentryfield{#1}%
6655   {\glsnameaccessdisplay{#2}{#1}}%
6656   {\glsdescriptionaccessdisplay{#3}{#1}}%
6657   {\glsymbolaccessdisplay{#4}{#1}}{#5}%
6658 }

```

glossarysubentryfield

```

6659 \newcommand*{\accsuppglossarysubentryfield}[6]{%
6660   \glossaryentryfield{#1}{#2}%
6661   {\glsnameaccessdisplay{#3}{#2}}%
6662   {\glsdescriptionaccessdisplay{#4}{#2}}%
6663   {\glsymbolaccessdisplay{#5}{#2}}{#6}%
6664 }

```

5.4 Acronyms

Use `\newacronymhook` to modify the key list to set the access text to the long version by default.

```

6665 \renewcommand*{\newacronymhook}{%
6666   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
6667     \the\glskeylisttok}%
6668   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
6669 }

```

DefaultNewAcronymDef Modify default style to use access text:

```

6670 \renewcommand*{\DefaultNewAcronymDef}{%
6671   \edef\@do@newglossaryentry{%
6672     \noexpand\newglossaryentry{\the\glslabeltok}%
6673     {%
6674       type=\acronymtype,%
6675       name={\the\glsshorttok},%
6676       description={\the\glslongtok},%
6677       descriptionaccess=\relax,%
6678       text={\the\glsshorttok},%
6679       access={\noexpand\@glo@textaccess},%
6680       sort={\the\glsshorttok},%
6681       short={\the\glsshorttok},%
6682       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6683       shortaccess={\the\glslongtok},%
6684       long={\the\glslongtok},%
6685       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6686       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6687       first={\noexpand\glslongaccessdisplay
6688         {\the\glslongtok}{\the\glslabeltok}\space
6689         (\noexpand\glsshortaccessdisplay
6690           {\the\glsshorttok}{\the\glslabeltok})},%

```

```

6691 plural={\the\glsshorttok\acrpluralsuffix},%
6692 firstplural={\noexpand\glslongpluralaccessdisplay
6693   {\noexpand\@glo@longpl}{\the\glslabeltok}\space
6694   (\noexpand\glsshortpluralaccessdisplay
6695     {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
6696 firstaccess=\relax,
6697 firstpluralaccess=\relax,
6698 textaccess={\noexpand\@glo@shortaccess},%
6699 \the\glskeylisttok
6700 }%
6701 }%
6702 \@do@newglossaryentry
6703 }

```

otnoteNewAcronymDef

```

6704 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
6705   \edef\@do@newglossaryentry{%
6706     \noexpand\newglossaryentry{\the\glslabeltok}%
6707     {%
6708       type=\acronymtype,%
6709       name={\noexpand\acronymfont{\the\glsshorttok}},%
6710       sort={\the\glsshorttok},%
6711       text={\the\glsshorttok},%
6712       short={\the\glsshorttok},%
6713       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6714       shortaccess={\the\glslongtok},%
6715       long={\the\glslongtok},%
6716       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6717       access={\noexpand\@glo@textaccess},%
6718       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6719       symbol={\the\glslongtok},%
6720       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6721       firstpluralaccess=\relax,
6722       textaccess={\noexpand\@glo@shortaccess},%
6723       \the\glskeylisttok
6724     }%
6725   }%
6726   \@do@newglossaryentry
6727 }

```

iptionNewAcronymDef

```

6728 \renewcommand*{\DescriptionNewAcronymDef}{%
6729   \edef\@do@newglossaryentry{%
6730     \noexpand\newglossaryentry{\the\glslabeltok}%
6731     {%
6732       type=\acronymtype,%
6733       name={\noexpand
6734         \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
6735       access={\noexpand\@glo@textaccess},%

```

```

6736     sort={\the\glsshorttok},%
6737     short={\the\glsshorttok},%
6738     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6739     shortaccess={\the\glslongtok},%
6740     long={\the\glslongtok},%
6741     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6742     first={\the\glslongtok},%
6743     firstaccess=\relax,
6744     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6745     text={\the\glsshorttok},%
6746     textaccess={\the\glslongtok},%
6747     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6748     symbol={\noexpand\@glo@text},%
6749     symbolaccess={\noexpand\@glo@textaccess},%
6750     symbolplural={\noexpand\@glo@plural},%
6751     firstpluralaccess=\relax,
6752     textaccess={\noexpand\@glo@shortaccess},%
6753     \the\glskeylisttok}%
6754 }%
6755 \@do@newglossaryentry
6756 }

```

otnoteNewAcronymDef

```

6757 \renewcommand*{\FootnoteNewAcronymDef}{%
6758   \edef\@do@newglossaryentry{%
6759     \noexpand\newglossaryentry{\the\glslabeltok}%
6760     {%
6761       type=\acronymtype,%
6762       name={\noexpand\acronymfont{\the\glsshorttok}},%
6763       sort={\the\glsshorttok},%
6764       text={\the\glsshorttok},%
6765       textaccess={\the\glslongtok},%
6766       access={\noexpand\@glo@textaccess},%
6767       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6768       short={\the\glsshorttok},%
6769       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6770       long={\the\glslongtok},%
6771       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6772       description={\the\glslongtok},%
6773       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6774       \the\glskeylisttok
6775     }%
6776   }%
6777   \@do@newglossaryentry
6778 }

```

\SmallNewAcronymDef

```

6779 \renewcommand*{\SmallNewAcronymDef}{%
6780   \edef\@do@newglossaryentry{%

```



```

6781 \noexpand\newglossaryentry{\the\glslabeltok}%
6782 {%
6783     type=\acronymtype,%
6784     name={\noexpand\acronymfont{\the\glsshorttok}},%
6785     access={\noexpand\@glo@symbolaccess},%
6786     sort={\the\glsshorttok},%
6787     short={\the\glsshorttok},%
6788     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6789     shortaccess={\the\glslongtok},%
6790     long={\the\glslongtok},%
6791     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6792     text={\noexpand\@glo@short},%
6793     textaccess={\noexpand\@glo@shortaccess},%
6794     plural={\noexpand\@glo@shortpl},%
6795     first={\the\glslongtok},%
6796     firstaccess=\relax,%
6797     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6798     description={\noexpand\@glo@first},%
6799     descriptionplural={\noexpand\@glo@firstplural},%
6800     symbol={\the\glsshorttok},%
6801     symbolaccess={\the\glslongtok},%
6802     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6803     \the\glskeylisttok
6804 }%
6805 }%
6806 \@do@newglossaryentry
6807 }

```

The following are kept for compatibility with versions before 3.0:

```

\glsshortaccesskey
6808 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

hortpluralaccesskey
6809 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\glslongaccesskey
6810 \newcommand*{\glslongaccesskey}{\glslongkey access}%

longpluralaccesskey
6811 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

5.5 Debugging Commands

```

\showglongnameaccess
6812 \newcommand*{\showglongnameaccess}[1]{%
6813     \expandafter\show\csname glo@#1@textaccess\endcsname
6814 }

```

```

\showglotextaccess
6815 \newcommand*{\showglotextaccess}[1]{%
6816   \expandafter\show\csname glo@#1@textaccess\endcsname
6817 }

showglopluralaccess
6818 \newcommand*{\showglopluralaccess}[1]{%
6819   \expandafter\show\csname glo@#1@pluralaccess\endcsname
6820 }

\showglofirstaccess
6821 \newcommand*{\showglofirstaccess}[1]{%
6822   \expandafter\show\csname glo@#1@firstaccess\endcsname
6823 }

lofirstpluralaccess
6824 \newcommand*{\showglofirstpluralaccess}[1]{%
6825   \expandafter\show\csname glo@#1@firstpluralaccess\endcsname
6826 }

showglosymbolaccess
6827 \newcommand*{\showglosymbolaccess}[1]{%
6828   \expandafter\show\csname glo@#1@symbolaccess\endcsname
6829 }

osymbolpluralaccess
6830 \newcommand*{\showglosymbolpluralaccess}[1]{%
6831   \expandafter\show\csname glo@#1@symbolpluralaccess\endcsname
6832 }

\showglodescaccess
6833 \newcommand*{\showglodescaccess}[1]{%
6834   \expandafter\show\csname glo@#1@descaccess\endcsname
6835 }

glodescpluralaccess
6836 \newcommand*{\showglodescpluralaccess}[1]{%
6837   \expandafter\show\csname glo@#1@descpluralaccess\endcsname
6838 }

\showgloshortaccess
6839 \newcommand*{\showgloshortaccess}[1]{%
6840   \expandafter\show\csname glo@#1@shortaccess\endcsname
6841 }

loshortpluralaccess
6842 \newcommand*{\showgloshortpluralaccess}[1]{%
6843   \expandafter\show\csname glo@#1@shortpluralaccess\endcsname
6844 }

```

\showglolongaccess

```
6845 \newcommand*{\showglolongaccess}[1]{%
6846   \expandafter\show\csname glo@#1@longaccess\endcsname
6847 }
```

glolongpluralaccess

```
6848 \newcommand*{\showglolongpluralaccess}[1]{%
6849   \expandafter\show\csname glo@#1@longpluralaccess\endcsname
6850 }
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex.

6.1 Babel Captions

Define captions if multi-lingual support is required, but the package is not loaded.

```
6851 \NeedsTeXFormat{LaTeX2e}
6852 \ProvidesPackage{glossaries-babel}[2009/04/16 v1.2 (NLCT)]
```

English:

```
6853 \@ifundefined{captionsenglish}{-}{%
6854   \addto\captionsenglish{%
6855     \renewcommand*{\glossaryname}{Glossary}%
6856     \renewcommand*{\acronymname}{Acronyms}%
6857     \renewcommand*{\entryname}{Notation}%
6858     \renewcommand*{\descriptionname}{Description}%
6859     \renewcommand*{\symbolname}{Symbol}%
6860     \renewcommand*{\pagelistname}{Page List}%
6861     \renewcommand*{\glssymbolsgroupname}{Symbols}%
6862     \renewcommand*{\glsnumbersgroupname}{Numbers}%
6863 }%
6864 }
6865 \@ifundefined{captionsamerican}{-}{%
6866   \addto\captionsamerican{%
6867     \renewcommand*{\glossaryname}{Glossary}%
6868     \renewcommand*{\acronymname}{Acronyms}%
6869     \renewcommand*{\entryname}{Notation}%
6870     \renewcommand*{\descriptionname}{Description}%
6871     \renewcommand*{\symbolname}{Symbol}%
6872     \renewcommand*{\pagelistname}{Page List}%
6873     \renewcommand*{\glssymbolsgroupname}{Symbols}%
6874     \renewcommand*{\glsnumbersgroupname}{Numbers}%
6875 }%
6876 }
6877 \@ifundefined{captionsaustralian}{-}{%
```

```

6878 \addto\captionsaustrian{%
6879 \renewcommand*{\glossaryname}{Glossary}%
6880 \renewcommand*{\acronymname}{Acronyms}%
6881 \renewcommand*{\entryname}{Notation}%
6882 \renewcommand*{\descriptionname}{Description}%
6883 \renewcommand*{\symbolname}{Symbol}%
6884 \renewcommand*{\pagelistname}{Page List}%
6885 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6886 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6887 }%
6888 }
6889 \@ifundefined{captionsbritish}{-}{%
6890 \addto\captionsbritish{%
6891 \renewcommand*{\glossaryname}{Glossary}%
6892 \renewcommand*{\acronymname}{Acronyms}%
6893 \renewcommand*{\entryname}{Notation}%
6894 \renewcommand*{\descriptionname}{Description}%
6895 \renewcommand*{\symbolname}{Symbol}%
6896 \renewcommand*{\pagelistname}{Page List}%
6897 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6898 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6899 }%
6900 \@ifundefined{captionscanadian}{-}{%
6901 \addto\captionscanadian{%
6902 \renewcommand*{\glossaryname}{Glossary}%
6903 \renewcommand*{\acronymname}{Acronyms}%
6904 \renewcommand*{\entryname}{Notation}%
6905 \renewcommand*{\descriptionname}{Description}%
6906 \renewcommand*{\symbolname}{Symbol}%
6907 \renewcommand*{\pagelistname}{Page List}%
6908 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6909 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6910 }%
6911 }
6912 \@ifundefined{captionsnewzealand}{-}{%
6913 \addto\captionnewzealand{%
6914 \renewcommand*{\glossaryname}{Glossary}%
6915 \renewcommand*{\acronymname}{Acronyms}%
6916 \renewcommand*{\entryname}{Notation}%
6917 \renewcommand*{\descriptionname}{Description}%
6918 \renewcommand*{\symbolname}{Symbol}%
6919 \renewcommand*{\pagelistname}{Page List}%
6920 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6921 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6922 }%
6923 }
6924 \@ifundefined{captionsUKenglish}{-}{%
6925 \addto\captionUKenglish{%
6926 \renewcommand*{\glossaryname}{Glossary}%

```

```

6927 \renewcommand*{\acronymname}{Acronyms}%
6928 \renewcommand*{\entryname}{Notation}%
6929 \renewcommand*{\descriptionname}{Description}%
6930 \renewcommand*{\symbolname}{Symbol}%
6931 \renewcommand*{\pagelistname}{Page List}%
6932 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6933 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6934 }%
6935 }
6936 \@ifundefined{captionsUSenglish}{-}{%
6937 \addto\captionsUSenglish{%
6938 \renewcommand*{\glossaryname}{Glossary}%
6939 \renewcommand*{\acronymname}{Acronyms}%
6940 \renewcommand*{\entryname}{Notation}%
6941 \renewcommand*{\descriptionname}{Description}%
6942 \renewcommand*{\symbolname}{Symbol}%
6943 \renewcommand*{\pagelistname}{Page List}%
6944 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6945 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6946 }%
6947 }

```

German (quite a few variations were suggested for German; I settled on the following):

```

6948 \@ifundefined{captionsgerman}{-}{%
6949 \addto\captionsgerman{%
6950 \renewcommand*{\glossaryname}{Glossar}%
6951 \renewcommand*{\acronymname}{Akronyme}%
6952 \renewcommand*{\entryname}{Bezeichnung}%
6953 \renewcommand*{\descriptionname}{Beschreibung}%
6954 \renewcommand*{\symbolname}{Symbol}%
6955 \renewcommand*{\pagelistname}{Seiten}%
6956 \renewcommand*{\glssymbolsgroupname}{Symbole}%
6957 \renewcommand*{\glsnumbersgroupname}{Zahlen}}
6958 }

```

ngerman is identical to German:

```

6959 \@ifundefined{captionsgerman}{-}{%
6960 \addto\captionsgerman{%
6961 \renewcommand*{\glossaryname}{Glossar}%
6962 \renewcommand*{\acronymname}{Akronyme}%
6963 \renewcommand*{\entryname}{Bezeichnung}%
6964 \renewcommand*{\descriptionname}{Beschreibung}%
6965 \renewcommand*{\symbolname}{Symbol}%
6966 \renewcommand*{\pagelistname}{Seiten}%
6967 \renewcommand*{\glssymbolsgroupname}{Symbole}%
6968 \renewcommand*{\glsnumbersgroupname}{Zahlen}}
6969 }

```

Italian:

```

6970 \@ifundefined{captionsitalian}{-}{%

```

```

6971 \addto\captionssitalian{%
6972 \renewcommand*{\glossaryname}{Glossario}%
6973 \renewcommand*{\acronymname}{Acronimi}%
6974 \renewcommand*{\entryname}{Nomenclatura}%
6975 \renewcommand*{\descriptionname}{Descrizione}%
6976 \renewcommand*{\symbolname}{Simbolo}%
6977 \renewcommand*{\pagelistname}{Elenco delle pagine}%
6978 \renewcommand*{\glssymbolsgroupname}{Simboli}%
6979 \renewcommand*{\glsnumbersgroupname}{Numeri}}
6980 }

```

Dutch:

```

6981 \@ifundefined{captionssdutch}{}{%
6982 \addto\captionssdutch{%
6983 \renewcommand*{\glossaryname}{Woordenlijst}%
6984 \renewcommand*{\acronymname}{Acroniemen}%
6985 \renewcommand*{\entryname}{Benaming}%
6986 \renewcommand*{\descriptionname}{Beschrijving}%
6987 \renewcommand*{\symbolname}{Symbool}%
6988 \renewcommand*{\pagelistname}{Pagina's}%
6989 \renewcommand*{\glssymbolsgroupname}{Symbolen}%
6990 \renewcommand*{\glsnumbersgroupname}{Cijfers}}
6991 }

```

Spanish:

```

6992 \@ifundefined{captionssspanish}{}{%
6993 \addto\captionssspanish{%
6994 \renewcommand*{\glossaryname}{Glosario}%
6995 \renewcommand*{\acronymname}{Siglas}%
6996 \renewcommand*{\entryname}{Entrada}%
6997 \renewcommand*{\descriptionname}{Descripci\'on}%
6998 \renewcommand*{\symbolname}{S\'mbolo}%
6999 \renewcommand*{\pagelistname}{Lista de p\'aginas}%
7000 \renewcommand*{\glssymbolsgroupname}{S\'mbolos}%
7001 \renewcommand*{\glsnumbersgroupname}{N\'umeros}}
7002 }

```

French:

```

7003 \@ifundefined{captionssfrench}{}{%
7004 \addto\captionssfrench{%
7005 \renewcommand*{\glossaryname}{Glossaire}%
7006 \renewcommand*{\acronymname}{Acronymes}%
7007 \renewcommand*{\entryname}{Termes}%
7008 \renewcommand*{\descriptionname}{Description}%
7009 \renewcommand*{\symbolname}{Symbole}%
7010 \renewcommand*{\pagelistname}{Pages}%
7011 \renewcommand*{\glssymbolsgroupname}{Symboles}%
7012 \renewcommand*{\glsnumbersgroupname}{Nombres}}
7013 }
7014 \@ifundefined{captionssfrenchb}{}{%
7015 \addto\captionssfrenchb{%

```

```

7016 \renewcommand*{\glossaryname}{Glossaire}%
7017 \renewcommand*{\acronymname}{Acronymes}%
7018 \renewcommand*{\entryname}{Terme}%
7019 \renewcommand*{\descriptionname}{Description}%
7020 \renewcommand*{\symbolname}{Symbole}%
7021 \renewcommand*{\pagelistname}{Pages}%
7022 \renewcommand*{\glssymbolsgroupname}{Symboles}%
7023 \renewcommand*{\glsnumbersgroupname}{Nombres}}
7024 }
7025 \@ifundefined{captionsfrancais}{}{%
7026 \addto\captionsfrancais{%
7027 \renewcommand*{\glossaryname}{Glossaire}%
7028 \renewcommand*{\acronymname}{Acronymes}%
7029 \renewcommand*{\entryname}{Terme}%
7030 \renewcommand*{\descriptionname}{Description}%
7031 \renewcommand*{\symbolname}{Symbole}%
7032 \renewcommand*{\pagelistname}{Pages}%
7033 \renewcommand*{\glssymbolsgroupname}{Symboles}%
7034 \renewcommand*{\glsnumbersgroupname}{Nombres}}
7035 }

```

Danish:

```

7036 \@ifundefined{captionsdanish}{}{%
7037 \addto\captionsdanish{%
7038 \renewcommand*{\glossaryname}{Ordliste}%
7039 \renewcommand*{\acronymname}{Akronymer}%
7040 \renewcommand*{\entryname}{Symbolforklaring}%
7041 \renewcommand*{\descriptionname}{Beskrivelse}%
7042 \renewcommand*{\symbolname}{Symbol}%
7043 \renewcommand*{\pagelistname}{Side}%
7044 \renewcommand*{\glssymbolsgroupname}{Symboler}%
7045 \renewcommand*{\glsnumbersgroupname}{Tal}}
7046 }

```

Irish:

```

7047 \@ifundefined{captionsirish}{}{%
7048 \addto\captionsirish{%
7049 \renewcommand*{\glossaryname}{Gluais}%
7050 \renewcommand*{\acronymname}{Acrainmneacha}%

```

wasn't sure whether to go for Nóta (Note), Ciall ('Meaning', 'sense') or Brí ('Meaning'). In the end I chose Ciall.

```

7051 \renewcommand*{\entryname}{Ciall}%
7052 \renewcommand*{\descriptionname}{Tuairisc}%

```

Again, not sure whether to use Comhartha/Comharthaí or Siombail/Siombaile, so have chosen the former.

```

7053 \renewcommand*{\symbolname}{Comhartha}%
7054 \renewcommand*{\glssymbolsgroupname}{Comhartha\'}{\i}}%
7055 \renewcommand*{\pagelistname}{Leathanaigh}%
7056 \renewcommand*{\glsnumbersgroupname}{Uimhreacha}}

```

7057 }

Hungarian:

```
7058 \@ifundefined{captionsmagyar}{%}{%
7059   \addto\captionsmagyar{%
7060     \renewcommand*{\glossaryname}{Sz\`ojegyz\`ek}%
7061     \renewcommand*{\acronymname}{Bet\H uszavak}%
7062     \renewcommand*{\entryname}{Kifejez\`es}%
7063     \renewcommand*{\descriptionname}{Magyar\`azat}%
7064     \renewcommand*{\symbolname}{Jel\"ol\`es}%
7065     \renewcommand*{\pagelistname}{Oldalsz\`am}%
7066     \renewcommand*{\glssymbolsgroupname}{Jelek}%
7067     \renewcommand*{\glsnumbersgroupname}{Sz\`amjegyek}%
7068   }
7069 }
7070 \@ifundefined{captionshungarian}{%}{%
7071   \addto\captionshungarian{%
7072     \renewcommand*{\glossaryname}{Sz\`ojegyz\`ek}%
7073     \renewcommand*{\acronymname}{Bet\H uszavak}%
7074     \renewcommand*{\entryname}{Kifejez\`es}%
7075     \renewcommand*{\descriptionname}{Magyar\`azat}%
7076     \renewcommand*{\symbolname}{Jel\"ol\`es}%
7077     \renewcommand*{\pagelistname}{Oldalsz\`am}%
7078     \renewcommand*{\glssymbolsgroupname}{Jelek}%
7079     \renewcommand*{\glsnumbersgroupname}{Sz\`amjegyek}%
7080   }
7081 }
```

Polish

```
7082 \@ifundefined{captionspolish}{%}{%
7083   \addto\captionspolish{%
7084     \renewcommand*{\glossaryname}{S{\l}ownik termin\`ow}%
7085     \renewcommand*{\acronymname}{Skr\`ot}%
7086     \renewcommand*{\entryname}{Termin}%
7087     \renewcommand*{\descriptionname}{Opis}%
7088     \renewcommand*{\symbolname}{Symbol}%
7089     \renewcommand*{\pagelistname}{Strony}%
7090     \renewcommand*{\glssymbolsgroupname}{Symbole}%
7091     \renewcommand*{\glsnumbersgroupname}{Liczby}}
7092 }
```

Brazilian

```
7093 \@ifundefined{captionsbrazil}{%}{%
7094   \addto\captionsbrazil{%
7095     \renewcommand*{\glossaryname}{Gloss\`ario}%
7096     \renewcommand*{\acronymname}{Siglas}%
7097     \renewcommand*{\entryname}{Nota\c c\~ao}%
7098     \renewcommand*{\descriptionname}{Descri\c c\~ao}%
7099     \renewcommand*{\symbolname}{S\`imbolo}%
7100     \renewcommand*{\pagelistname}{Lista de P\`aginas}%
7101     \renewcommand*{\glssymbolsgroupname}{S\`imbolos}%

```



```

7102 \renewcommand*{\glsnumbersgroupname}{N\,umeros}%
7103 }%
7104 }

```

6.2 Polyglossia Captions

```

7105 \NeedsTeXFormat{LaTeX2e}
7106 \ProvidesPackage{glossaries-polyglossia}[2009/11/09 v1.0 (NLCT)]

```

English:

```

7107 \@ifundefined{captionsenglish}{}{%
7108 \expandafter\toks@\expandafter{\captionsenglish
7109 \renewcommand*{\glossaryname}{\textenglish{Glossary}}%
7110 \renewcommand*{\acronymname}{\textenglish{Acronyms}}%
7111 \renewcommand*{\entryname}{\textenglish{Notation}}%
7112 \renewcommand*{\descriptionname}{\textenglish{Description}}%
7113 \renewcommand*{\symbolname}{\textenglish{Symbol}}%
7114 \renewcommand*{\pagelistname}{\textenglish{Page List}}%
7115 \renewcommand*{\glssymbolsgroupname}{\textenglish{Symbols}}%
7116 \renewcommand*{\glsnumbersgroupname}{\textenglish{Numbers}}%
7117 }%
7118 \edef\captionsenglish{\the\toks@}%
7119 }

```

German:

```

7120 \@ifundefined{captionsgerman}{}{%
7121 \expandafter\toks@\expandafter{\captionsgerman
7122 \renewcommand*{\glossaryname}{\textgerman{Glossar}}%
7123 \renewcommand*{\acronymname}{\textgerman{Akronyme}}%
7124 \renewcommand*{\entryname}{\textgerman{Bezeichnung}}%
7125 \renewcommand*{\descriptionname}{\textgerman{Beschreibung}}%
7126 \renewcommand*{\symbolname}{\textgerman{Symbol}}%
7127 \renewcommand*{\pagelistname}{\textgerman{Seiten}}%
7128 \renewcommand*{\glssymbolsgroupname}{\textgerman{Symbole}}%
7129 \renewcommand*{\glsnumbersgroupname}{\textgerman{Zahlen}}%
7130 }%
7131 \edef\captionsgerman{\the\toks@}%
7132 }

```

Italian:

```

7133 \@ifundefined{captionsitalian}{}{%
7134 \expandafter\toks@\expandafter{\captionsitalian
7135 \renewcommand*{\glossaryname}{\textitalian{Glossario}}%
7136 \renewcommand*{\acronymname}{\textitalian{Acronimi}}%
7137 \renewcommand*{\entryname}{\textitalian{Nomenclatura}}%
7138 \renewcommand*{\descriptionname}{\textitalian{Descrizione}}%
7139 \renewcommand*{\symbolname}{\textitalian{Simbolo}}%
7140 \renewcommand*{\pagelistname}{\textitalian{Elenco delle pagine}}%
7141 \renewcommand*{\glssymbolsgroupname}{\textitalian{Simboli}}%
7142 \renewcommand*{\glsnumbersgroupname}{\textitalian{Numeri}}%
7143 }%

```

```

7144 \edef\captionssitalian{\the\toks@}%
7145 }

```

Dutch:

```

7146 \@ifundefined{captionsdutch}{}{%
7147   \expandafter\toks@\expandafter{\captionsdutch
7148     \renewcommand*{\glossaryname}{\textdutch{Woordenlijst}}%
7149     \renewcommand*{\acronymname}{\textdutch{Acroniemen}}%
7150     \renewcommand*{\entryname}{\textdutch{Benaming}}%
7151     \renewcommand*{\descriptionname}{\textdutch{Beschrijving}}%
7152     \renewcommand*{\symbolname}{\textdutch{Symbool}}%
7153     \renewcommand*{\pagelistname}{\textdutch{Pagina's}}%
7154     \renewcommand*{\glssymbolsgroupname}{\textdutch{Symbolen}}%
7155     \renewcommand*{\glsnumbersgroupname}{\textdutch{Cijfers}}%
7156   }%
7157   \edef\captionsdutch{\the\toks@}%
7158 }

```

Spanish:

```

7159 \@ifundefined{captionsspanish}{}{%
7160   \expandafter\toks@\expandafter{\captionsspanish
7161     \renewcommand*{\glossaryname}{\textspanish{Glosario}}%
7162     \renewcommand*{\acronymname}{\textspanish{Siglas}}%
7163     \renewcommand*{\entryname}{\textspanish{Entrada}}%
7164     \renewcommand*{\descriptionname}{\textspanish{Descripci'on}}%
7165     \renewcommand*{\symbolname}{\textspanish{S'\{i\}mbolo}}%
7166     \renewcommand*{\pagelistname}{\textspanish{Lista de p'aginas}}%
7167     \renewcommand*{\glssymbolsgroupname}{\textspanish{S'\{i\}mbolos}}%
7168     \renewcommand*{\glsnumbersgroupname}{\textspanish{N'umeros}}%
7169   }%
7170   \edef\captionsspanish{\the\toks@}%
7171 }

```

French:

```

7172 \@ifundefined{captionsfrench}{}{%
7173   \expandafter\toks@\expandafter{\captionsfrench
7174     \renewcommand*{\glossaryname}{\textfrench{Glossaire}}%
7175     \renewcommand*{\acronymname}{\textfrench{Acronymes}}%
7176     \renewcommand*{\entryname}{\textfrench{Terme}}%
7177     \renewcommand*{\descriptionname}{\textfrench{Description}}%
7178     \renewcommand*{\symbolname}{\textfrench{Symbole}}%
7179     \renewcommand*{\pagelistname}{\textfrench{Pages}}%
7180     \renewcommand*{\glssymbolsgroupname}{\textfrench{Symboles}}%
7181     \renewcommand*{\glsnumbersgroupname}{\textfrench{Nombres}}%
7182   }%
7183   \edef\captionsfrench{\the\toks@}%
7184 }

```

Danish:

```

7185 \@ifundefined{captionsdanish}{}{%
7186   \expandafter\toks@\expandafter{\captionsdanish
7187     \renewcommand*{\glossaryname}{\textdanish{Ordliste}}%

```

```

7188 \renewcommand*{\acronymname}{\textdanish{Akronymer}}}%
7189 \renewcommand*{\entryname}{\textdanish{Symbolforklaring}}}%
7190 \renewcommand*{\descriptionname}{\textdanish{Beskrivelse}}}%
7191 \renewcommand*{\symbolname}{\textdanish{Symbol}}}%
7192 \renewcommand*{\pagelistname}{\textdanish{Side}}}%
7193 \renewcommand*{\glssymbolsgroupname}{\textdanish{Symboler}}}%
7194 \renewcommand*{\glsnumbersgroupname}{\textdanish{Tal}}}%
7195 }%
7196 \edef\captionsdanish{\the\toks@}%
7197 }

Irish:
7198 \@ifundefined{captionsirish}{}{%
7199 \expandafter\toks@\expandafter{\captionsirish
7200 \renewcommand*{\glossaryname}{\textirish{Gluais}}}%
7201 \renewcommand*{\acronymname}{\textirish{Acrainmneacha}}}%
7202 \renewcommand*{\entryname}{\textirish{Ciall}}}%
7203 \renewcommand*{\descriptionname}{\textirish{Tuirisc}}}%
7204 \renewcommand*{\symbolname}{\textirish{Comhartha}}}%
7205 \renewcommand*{\glssymbolsgroupname}{\textirish{Comhartha\,'{i}}}%
7206 \renewcommand*{\pagelistname}{\textirish{Leathanaigh}}}%
7207 \renewcommand*{\glsnumbersgroupname}{\textirish{Uimhreacha}}}%
7208 }%
7209 \edef\captionsirish{\the\toks@}%
7210 }

Hungarian:
7211 \@ifundefined{captionsmagyar}{}{%
7212 \expandafter\toks@\expandafter{\captionsmagyar
7213 \renewcommand*{\glossaryname}{\textmagyar{Sz\,'ojegy\,'ek}}}%
7214 \renewcommand*{\acronymname}{\textmagyar{Bet\H uszavak}}}%
7215 \renewcommand*{\entryname}{\textmagyar{Kifejez\,'es}}}%
7216 \renewcommand*{\descriptionname}{\textmagyar{Magyar\,'azat}}}%
7217 \renewcommand*{\symbolname}{\textmagyar{Jel\,"ol\,'es}}}%
7218 \renewcommand*{\pagelistname}{\textmagyar{Oldalsz\,'am}}}%
7219 \renewcommand*{\glssymbolsgroupname}{\textmagyar{Jelek}}}%
7220 \renewcommand*{\glsnumbersgroupname}{\textmagyar{Sz\,'amjegyek}}}%
7221 }%
7222 \edef\captionsmagyar{\the\toks@}%
7223 }

Polish
7224 \@ifundefined{captionspolish}{}{%
7225 \expandafter\toks@\expandafter{\captionspolish
7226 \renewcommand*{\glossaryname}{\textpolish{S{\l}ownik termin\,'ow}}}%
7227 \renewcommand*{\acronymname}{\textpolish{Skr\,'ot}}}%
7228 \renewcommand*{\entryname}{\textpolish{Termin}}}%
7229 \renewcommand*{\descriptionname}{\textpolish{Opis}}}%
7230 \renewcommand*{\symbolname}{\textpolish{Symbol}}}%
7231 \renewcommand*{\pagelistname}{\textpolish{Strony}}}%
7232 \renewcommand*{\glssymbolsgroupname}{\textpolish{Symbole}}}%

```

```

7233 \renewcommand*{\glsnumbersgroupname}{\textpolish{Liczby}}%
7234 }%
7235 \edef\captionspolish{\the\toks@}%
7236 }

```

Portugues

```

7237 \@ifundefined{captionoportuges}{}{%
7238 \expandafter\toks@\expandafter{\captionoportuges
7239 \renewcommand*{\glossaryname}{\textportuges{Gloss\`ario}}%
7240 \renewcommand*{\acronymname}{\textportuges{Siglas}}%
7241 \renewcommand*{\entryname}{\textportuges{Nota\c c\~ao}}%
7242 \renewcommand*{\descriptionname}{\textportuges{Descri\c c\~ao}}%
7243 \renewcommand*{\symbolname}{\textportuges{S\`imbolo}}%
7244 \renewcommand*{\pagelistname}{\textportuges{Lista de P\`aginas}}%
7245 \renewcommand*{\glssymbolsgroupname}{\textportuges{S\`imbolos}}%
7246 \renewcommand*{\glsnumbersgroupname}{\textportuges{N\`umeros}}%
7247 }%
7248 \edef\captionoportuges{\the\toks@}%
7249 }

```

6.3 Brazilian Dictionary

This is a dictionary file provided by Thiago de Melo for use with the package.

```

7250 \ProvidesDictionary{glossaries-dictionary}{Brazilian}

```

Provide Brazilian translations:

```

7251 \providetranslation{Glossary}{Gloss\`ario}
7252 \providetranslation{Acronyms}{Siglas}
7253 \providetranslation{Notation (glossaries)}{Nota\c c\~ao}
7254 \providetranslation{Description (glossaries)}{Descri\c c\~ao}
7255 \providetranslation{Symbol (glossaries)}{S\`imbolo}
7256 \providetranslation{Page List (glossaries)}{Lista de P\`aginas}
7257 \providetranslation{Symbols (glossaries)}{S\`imbolos}
7258 \providetranslation{Numbers (glossaries)}{N\`umeros}

```

6.4 Danish Dictionary

This is a dictionary file provided for use with the package.

```

7259 \ProvidesDictionary{glossaries-dictionary}{Danish}

```

Provide Danish translations:

```

7260 \providetranslation{Glossary}{Ordliste}
7261 \providetranslation{Acronyms}{Akronymer}
7262 \providetranslation{Notation (glossaries)}{Symbolforklaring}
7263 \providetranslation{Description (glossaries)}{Beskrivelse}
7264 \providetranslation{Symbol (glossaries)}{Symbol}
7265 \providetranslation{Page List (glossaries)}{Side}
7266 \providetranslation{Symbols (glossaries)}{Symboler}
7267 \providetranslation{Numbers (glossaries)}{Tal}

```

6.5 Dutch Dictionary

This is a dictionary file provided for use with the package.

```
7268 \ProvidesDictionary{glossaries-dictionary}{Dutch}
```

Provide Dutch translations:

```
7269 \providetranslation{Glossary}{Woordenlijst}
7270 \providetranslation{Acronyms}{Acroniemen}
7271 \providetranslation{Notation (glossaries)}{Benaming}
7272 \providetranslation{Description (glossaries)}{Beschrijving}
7273 \providetranslation{Symbol (glossaries)}{Symbool}
7274 \providetranslation{Page List (glossaries)}{Pagina's}
7275 \providetranslation{Symbols (glossaries)}{Symbolen}
7276 \providetranslation{Numbers (glossaries)}{Cijfers}
```

6.6 English Dictionary

This is a dictionary file provided for use with the package.

```
7277 \ProvidesDictionary{glossaries-dictionary}{English}
```

Provide English translations:

```
7278 \providetranslation{Glossary}{Glossary}
7279 \providetranslation{Acronyms}{Acronyms}
7280 \providetranslation{Notation (glossaries)}{Notation}
7281 \providetranslation{Description (glossaries)}{Description}
7282 \providetranslation{Symbol (glossaries)}{Symbol}
7283 \providetranslation{Page List (glossaries)}{Page List}
7284 \providetranslation{Symbols (glossaries)}{Symbols}
7285 \providetranslation{Numbers (glossaries)}{Numbers}
```

6.7 French Dictionary

This is a dictionary file provided for use with the package.

```
7286 \ProvidesDictionary{glossaries-dictionary}{French}
```

Provide French translations:

```
7287 \providetranslation{Glossary}{Glossaire}
7288 \providetranslation{Acronyms}{Acronymes}
7289 \providetranslation{Notation (glossaries)}{Terme}
7290 \providetranslation{Description (glossaries)}{Description}
7291 \providetranslation{Symbol (glossaries)}{Symbole}
7292 \providetranslation{Page List (glossaries)}{Pages}
7293 \providetranslation{Symbols (glossaries)}{Symboles}
7294 \providetranslation{Numbers (glossaries)}{Nombres}
```

6.8 German Dictionary

This is a dictionary file provided for use with the package.

```
7295 \ProvidesDictionary{glossaries-dictionary}{German}
```

Provide German translations (quite a few variations were suggested for German; I settled on the following):

```
7296 \providetranslation{Glossary}{Glossar}
7297 \providetranslation{Acronyms}{Akronyme}
7298 \providetranslation{Notation (glossaries)}{Bezeichnung}
7299 \providetranslation{Description (glossaries)}{Beschreibung}
7300 \providetranslation{Symbol (glossaries)}{Symbol}
7301 \providetranslation{Page List (glossaries)}{Seiten}
7302 \providetranslation{Symbols (glossaries)}{Symbole}
7303 \providetranslation{Numbers (glossaries)}{Zahlen}
```

6.9 Irish Dictionary

This is a dictionary file provided for use with the package.

```
7304 \ProvidesDictionary{glossaries-dictionary}{Irish}
```

Provide Irish translations:

```
7305 \providetranslation{Glossary}{Gluais}
7306 \providetranslation{Acronyms}{Acrainmneacha}
7307 \providetranslation{Notation (glossaries)}{Ciall}
7308 \providetranslation{Description (glossaries)}{Tuairisc}
7309 \providetranslation{Symbol (glossaries)}{Comhartha}
7310 \providetranslation{Page List (glossaries)}{Leathanaigh}
7311 \providetranslation{Symbols (glossaries)}{Comhartha'\{i}}
7312 \providetranslation{Numbers (glossaries)}{Uimhreacha}
```

6.10 Italian Dictionary

This is a dictionary file provided for use with the package.

```
7313 \ProvidesDictionary{glossaries-dictionary}{Italian}
```

Provide Italian translations:

```
7314 \providetranslation{Glossary}{Glossario}
7315 \providetranslation{Acronyms}{Acronimi}
7316 \providetranslation{Notation (glossaries)}{Nomenclatura}
7317 \providetranslation{Description (glossaries)}{Descrizione}
7318 \providetranslation{Symbol (glossaries)}{Simbolo}
7319 \providetranslation{Page List (glossaries)}{Elenco delle pagine}
7320 \providetranslation{Symbols (glossaries)}{Simboli}
7321 \providetranslation{Numbers (glossaries)}{Numeri}
```

6.11 Magyar Dictionary

This is a dictionary file provided for use with the package.

```
7322 \ProvidesDictionary{glossaries-dictionary}{Magyar}
```

Provide translations:

```
7323 \providetranslation{Glossary}{Sz\'ojegy\'ek}
7324 \providetranslation{Acronyms}{Bet\'H uszavak}
```

```

7325 \providetranslation{Notation (glossaries)}{Kifejez\`es}
7326 \providetranslation{Description (glossaries)}{Magyar\`azat}
7327 \providetranslation{Symbol (glossaries)}{Jel\`ol\`es}
7328 \providetranslation{Page List (glossaries)}{Oldalsz\`am}
7329 \providetranslation{Symbols (glossaries)}{Jelek}
7330 \providetranslation{Numbers (glossaries)}{Sz\`amjegyek}

```

6.12 Polish Dictionary

This is a dictionary file provided for use with the package.

```
7331 \ProvidesDictionary{glossaries-dictionary}{Polish}
```

Provide Polish translations:

```

7332 \providetranslation{Glossary}{S{\l}ownik termin\`ow}
7333 \providetranslation{Acronyms}{Skr\`ot}
7334 \providetranslation{Notation (glossaries)}{Termin}
7335 \providetranslation{Description (glossaries)}{Opis}
7336 \providetranslation{Symbol (glossaries)}{Symbol}
7337 \providetranslation{Page List (glossaries)}{Strony}
7338 \providetranslation{Symbols (glossaries)}{Symbole}
7339 \providetranslation{Numbers (glossaries)}{Liczby}

```

6.13 Serbian Dictionary

This dictionary was provided by Zoran Filipovic.

```

7340 \ProvidesDictionary{glossaries-dictionary}{Serbian}
7341 \providetranslation{Glossary}{Mali re\`v cnik}
7342 \providetranslation{Acronyms}{Skra\` cenice}
7343 \providetranslation{Notation (glossaries)}{Oznaka}
7344 \providetranslation{Description (glossaries)}{Opis}
7345 \providetranslation{Symbol (glossaries)}{Simbol}
7346 \providetranslation{Page List (glossaries)}{Stranica}
7347 \providetranslation{Symbols (glossaries)}{Simboli}
7348 \providetranslation{Numbers (glossaries)}{Brojevi}

```

6.14 Spanish Dictionary

This is a dictionary file provided for use with the package.

```
7349 \ProvidesDictionary{glossaries-dictionary}{Spanish}
```

Provide Spanish translations:

```

7350 \providetranslation{Glossary}{Glosario}
7351 \providetranslation{Acronyms}{Siglas}
7352 \providetranslation{Notation (glossaries)}{Entrada}
7353 \providetranslation{Description (glossaries)}{Descripci\`on}
7354 \providetranslation{Symbol (glossaries)}{S\`{\i}mbolo}
7355 \providetranslation{Page List (glossaries)}{Lista de p\`aginas}
7356 \providetranslation{Symbols (glossaries)}{S\`{\i}mbolos}
7357 \providetranslation{Numbers (glossaries)}{N\`umeros}

```

Glossary

`makeindex` An indexing application. 17

`xindy` An flexible indexing application with multilingual support written in Perl. 17

Change History

- 1.01
 - General: Added range facility in format key 61
 - `\writeist`: Added spaces after `'delimN` and `'delimR` in ist file 118
- 1.03
 - `\makefirstuc`: changed `'protected@edef` to `'def` 171
- 1.04
 - General: Added `'glstextformat` ... 57
- 1.05
 - `\glossarysection`: added `'@mk-` both to `'glossarysection` 26
 - `\glsmakefirstuc`: new 171
 - `\newglossaryentry`: Changed the default value of the sort key to just the value of the name key 48
- 1.06
 - General: now requires `etoolbox` . 170
 - `\capitalisewords`: new 172
 - `\xcapitalisewords`: new 172
- 1.07
 - `\@gls@link`: fixed bug caused by `\theglsentrycounter` setting the page number too soon 59
 - `\glsadd`: fixed bug caused by `\theglsentrycounter` setting the page number too soon 116
- 1.08
 - General: Added babel support ... 21
 - `listgroup`: changed `listgroup` style to use `\glsgetgrouptitle` 177
 - `altlistgroup`: changed `altlistgroup` style to use `\glsgetgrouptitle` 178
- 1.1
 - `\newglossaryentry`: Fixed error message to say “description key” rather than “desc key” .. 47
- 1.1
 - `\@glossarysection`: numbered sections and auto label added 28
 - `\@gls@tmpb`: changed `\toksdef` to `\newtoks` 64
 - `\@gls@toc`: numberline added .. 29
 - `\@p@glossarysection`: numbered sections and auto label added 28
 - General: Added support for translator package 22
 - `amsgen` now loaded (`\new@ifnextchar` needed) 3
 - `translate`: `translate` option added 16
 - `\setglossarysection`: new ... 28
 - `numberedsection`: `numberedsection` package option added . 6
 - `numberline`: `numberline` option added 5
- 1.12
 - `\@GLSpl`: now uses `'glsentrydescplural` and `'glsentrysymbolplural` instead of `'glsentrydesc` and `'glsentrysymbol` 78
 - `\@Glspl@`: now uses `'glsentrydescplural` and `'glsentrysymbolplural` instead of `'glsentrydesc` and `'glsentrysymbol` 76
 - `\@glspl@`: now uses `'glsentrydescplural` and `'glsentrysymbolplural` instead of `'glsentrydesc` and `'glsentrysymbol` 75

General: added check for 'hypertarget separate to 'hyperlink (memoir defines 'hyperlink but not 'hypertarget) ...	69	fixed bug ('glstext shouldn't use 'gls@⟨type⟩@display)	80
fixed bug ('GLSdesc shouldn't use 'gls@⟨type⟩@display)	89	descriptionplural: new	43
fixed bug ('Glsdesc shouldn't use 'gls@⟨type⟩@display)	88	\Glsentrydescplural: New ..	110
fixed bug ('glsdesc shouldn't use 'gls@⟨type⟩@display)	88	\glsentrydescplural: New ..	110
fixed bug ('GLSfirst shouldn't use 'gls@⟨type⟩@display)	83	\Glsentrysymbolplural: New	111
fixed bug ('Glsfirst shouldn't use 'gls@⟨type⟩@display)	82	\glsentrysymbolplural: New	111
fixed bug ('glsfirst shouldn't use 'gls@⟨type⟩@display)	82	\newglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended)	48
fixed bug ('GLSfirstplural shouldn't use 'gls@⟨type⟩@display)	86	descriptionplural support added	48
fixed bug ('Glsfirstplural shouldn't use 'gls@⟨type⟩@display)	85	symbolplural support added ..	48
fixed bug ('glsfirstplural shouldn't use 'gls@⟨type⟩@display)	85	\SetDescriptionFootnoteAcronymStyle: Added 'protect before 'footnote and 'glslink	153
fixed bug ('GLSname shouldn't use 'gls@⟨type⟩@display)	87	\SetFootnoteAcronymStyle: Added 'protect before 'footnote and 'glslink	157
fixed bug ('glsname shouldn't use 'gls@⟨type⟩@display) ..	86, 87	symbolplural: new	44
fixed bug ('GLSplural shouldn't use 'gls@⟨type⟩@display)	84	1.13	
fixed bug ('Glsplural shouldn't use 'gls@⟨type⟩@display)	84	General: Add Polish support	248, 251
fixed bug ('glsplural shouldn't use 'gls@⟨type⟩@display)	83	fixed bug that ignores 3rd parameter	80–93
fixed bug ('GLSymbol shouldn't use 'gls@⟨type⟩@display)	92	\ACRfullpl: new	149
fixed bug ('Glsymbol shouldn't use 'gls@⟨type⟩@display)	91	\Acrfullpl: new	149
fixed bug ('glssymbol shouldn't use 'gls@⟨type⟩@display)	91	\acrfullpl: new	149
fixed bug ('glssymbolplural shouldn't use 'gls@⟨type⟩@display)	92	\acrpluralsuffix: New	147
fixed bug ('GLStext shouldn't use 'gls@⟨type⟩@display)	81	\newacronym: Removed restriction on only using 'newacronym in the preamble	147
fixed bug ('Glstext shouldn't use 'gls@⟨type⟩@display)	81	\newglossaryentry: Changed default first value	48
		Changed default firstplural value	48
		Removed restriction on only using 'newglossaryentry in the preamble	52
		1.14	
		\@gls@hypergroup: new	174
		General: added nonnumberlist key to 'printglossary	136
		added numberedsection key to 'printglossary	136
		\firstacronymfont: new	150
		\glsautoprefix: new	6
		\glsnavhyperlink: changed 'edef to 'protected@edef ...	173

<code>\glsnavhypertarget:</code>	added	checking if footnote option has	
	write to aux file	been used	75
<code>\glsnavigation:</code>	changed to	<code>\@glstarget:</code>	raised the hyper-
	only use labels for groups that		target so the target text doesn't
	are present		scroll off the top of the page
	174		69
1.15		<code>\newglossaryentry:</code>	Changed
<code>\@gls@link:</code>	added 'glslabel		def to let
 59		48
General:	Added 'glssettoctitle		Changed if to ifx
	... 22		50
<code>\gls@hypergroup</code>	new	1.17	
	173	<code>\@do@wrglossary:</code>	new
<code>\glsnavhypertarget:</code>	added		128
	check if rerun required	<code>\do@seeglossary:</code>	new
 173		131
<code>\glssettoctitle:</code>	new	<code>\@glo@storeentry:</code>	new
	20		53
<code>\newglossaryentry:</code>	check for	<code>\@glossary:</code>	changed defini-
	'@glo@first in description		tion to use \index instead of
	... 51		\@index
	check for '@glo@text in symbol		127
	52	<code>\@glsdefaultplural:</code>	new
<code>\printglossary:</code>	changed the		47
	way the TOC title is set	<code>\@glsdefaultsort:</code>	new
 133		47
1.16		<code>\@glshypernumber:</code>	new
<code>\@GLS@:</code>	Test glossary type is		144
	'acronymtype in addition to	<code>\@glsnoname:</code>	new
	checking if footnote option has		47
	been used	<code>\@glsnonextpages:</code>	new
	74, 232		137
<code>\@GLSpl:</code>	Test glossary type is	<code>\@wrglossary:</code>	modified to allow
	'acronymtype in addition to		for xindy support
	checking if footnote option has		127
	been used	General:	added Brazilian dictio-
	78		nary
<code>\@Gls@:</code>	Test glossary type is		252
	'acronymtype in addition to		Added Brazilian support
	checking if footnote option has	 248
	been used		added xindy support
	72		17
<code>\@GLspl@:</code>	Test glossary type is	parent:	new
	'acronymtype in addition to		45
	checking if footnote option has	see:	new
	been used		45
	77	<code>\gls@suffixF:</code>	new
<code>\@gls@:</code>	Test glossary type is		24
	'acronymtype in addition to	<code>\gls@suffixFF:</code>	new
	checking if footnote option has		25
	been used	<code>\glshyperlink:</code>	new
	71		116
<code>\@gls@pl@:</code>	Test glossary type is	<code>\glshypernumber:</code>	modified to
	'acronymtype in addition to		allow material to be attached
	checking if footnote option has		to location
	been used		143
	233	<code>\glsnavhyperlink:</code>	replaced 'hy-
<code>\@glsdisp:</code>	Test glossary type is		perlink to '@glslink
	'acronymtype in addition to		173
	checking if footnote option has	<code>\glsnavhypertarget:</code>	replaced
	been used		'hypertarget to '@glstarget
	80		173
<code>\@glspl@:</code>	Test glossary type is	<code>\glssee:</code>	new
	'acronymtype in addition to		131
		<code>\glsseeformat:</code>	new
			131
		<code>\glsSetSuffixF:</code>	new
			25
		<code>\glsSetSuffixFF:</code>	new
			25
		<code>\ifglsxindy:</code>	new
			17
		<code>\istfilename:</code>	added xindy sup-
			port
			23
		<code>\newglossaryentry:</code>	added non-
			umberlist key
			48
			added parent key
			48
			added see key
			48

Stored main part of entry format	1.2	General: fixed bug in ngerman	
when entry is defined	52	captions	245
\newglossarystyle: made 'new-		2.01	
glossarystyle long	142	\@gls@link: moved \@do@wrglossary	
\nopostdesc: new	23	before term is displayed to pre-	
nonumberlist: new	45	vent unwanted whatsit	60
\printglossary: added check to		General: added nomain package	
determine if 'printglossary is		option	12
already defined	133	\forall glossaries: replaced	
added print language to aux file	133	\ifthenelse with \ifx	38
order: order package option		\forall glossentries: replaced	
added	17	\ifthenelse with \ifx	38
\writeist: added xindy support	118	\glsdefmain: new	11
1.18		\glsdescwidth: changed	
\@gls@loadlist: new	7	\linewidth to \hsize . 180, 195	
\@gls@loadlong: new	7	\glslistdottedwidth: changed	
\@gls@loadsuper: new	7	\linewidth to \hsize	179
\@gls@loadtree: new	8	\glspagelistwidth: changed	
\glstarget: new	140	\linewidth to \hsize . 180, 195	
\newglossaryentry: Changed		\writeist: removed item_02 - no	
default value of sort to '@gls-		such makeindex key	122
defaultsort	48	2.02	
moved sort sanitization to 'new-		General: Changed Brazil to Brazil-	
glossaryentry	51	ian	252
\oldacronym: new	146	false will prevent automatic	
nolist: new	7	loading of translator package	19
nolong: new	7	\glossarymark: New	27
sort: moved sanitization to 'new-		\glossarysection: changed	
glossaryentry	43	'@mkboth to 'glossarymark . .	26
nostyles: new	8	\printglossary: suppressed	
nosuper: new	7	warning globally rather than	
notree: new	8	locally	135
1.19		2.03	
\glsclearpage: new	29	\@GLS@: Added check for hyper-	
\glsdisp: new	79	first	74
\SetDescriptionAcronymStyle:		\@GLSpl: Added check for hyper-	
changed 'acronymfont to use		first	78
'textsmaller instead of 'smaller	156	\@Gls@: Added check for hyper-	
\SetDescriptionFootnoteAcronymStyle		first	72
changed 'acronymfont to use		\@Glspl@: Added check for hyper-	
'textsmaller instead of 'smaller	153	first	77
\SetFootnoteAcronymStyle:		\@gls@: Added check for hyper-	
changed 'acronymfont to use		first	71
'textsmaller instead of 'smaller	157	\@gls@link: new	59
\SetSmallAcronymStyle:		\@gls@link: added \leavevmode	
changed 'acronymfont to use		60
'textsmaller instead of 'smaller	159	Moved entry existence check to	
		avoid duplicate code	60

\@glsdisp: Added check for hyperfirst	80	\glsentryuseriv:new	112
\@glspl@: Added check for hyperfirst	75	\Glsentryuseriv:new	113
\glossarymark: Added check to see if it's already defined	27	\glsentryuseriv:new	113
hyperfirst:new	16	\glsentryuseriv:new	113
2.04		\newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	41
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	74	\newglossaryentry: added user1-6 keys	48
\@GLSpl@: Changed test to check if glossary type has been identified as a list of acronyms	78	\SetAcronymLists:new	13
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	72	\SetDefaultAcronymDisplayStyle:new	150
\@GLSpl@: Changed test to check if glossary type has been identified as a list of acronyms ...	77	\SetDefaultAcronymStyle:new	151
\@glossaryentryfield:new ..	53	\SetDescriptionAcronymDisplayStyle:new	155
\@glossarysubentryfield:new	53	\SetDescriptionDUAACronymDisplayStyle:new	154
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	71	\SetDescriptionFootnoteAcronymDisplayStyle:new	152
\@glsacronymlists:new	12	\SetDUADisplayStyle:new ..	160
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms ...	80	\SetFootnoteAcronymDisplayStyle:new	156
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms ...	75	\SetSmallAcronymDisplayStyle:new	158
\@newglossaryentryposthook:new	52	2.05	
\@newglossaryentryprehook:new	52	\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	79
acronymlists:new	13	Removed spurious brace. Patch provided by Sergiu Dotenco ..	80
\DeclareAcronymList:new ...	12	\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	123
\DefineAcronymSynonyms:new	163	2.06	
\glsadd: fixed bug that ignored counter	116	\altnewglossary:new	42
\Glsentryuseri:new	112	\CustomAcronymFields:new ..	162
\Glsentryuseri:new	112	\CustomNewAcronymDef:new ..	162
\Glsentryuserii:new	112	\SetCustomDisplayStyle:new	162
\Glsentryuserii:new	112	\SetCustomStyle:new	162
\Glsentryuseriii:new	112	2.07	
\Glsentryuseriii:new	112	General: glssadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	116
\Glsentryuseriv:new	113		

3.0

\@do@wrglossary: added check for hyper location prefix ...	129	\addglossarytocaptions: replaced \@ifundefined with \ifcsundef	21
modified to use new format ..	128	savewrites: new	18
\@glossarysec: replaced \@ifundefined with \ifcsundef	5	see: added \@glo@seeautonumberlist	45
\@do@seeglossary: Sanitize and escape cross-referencing information	131	seeautonumberlist: new	7
\@gls@counterwithin: new	9	\glossarymark: replaced \@ifundefined with \ifcsundef	27
\@gls@ifinlist: new	30	\glossarysection: replaced \@ifundefined with \ifcsundef	26
\@gls@link: added \@gls@saveentrycounter	60	\glossarystyle: replaced \@ifundefined with \ifcsundef	142
added \@gls@setsort	60	\gls@codepage: replaced \@ifundefined with \ifcsundef	18
\@gls@saveentrycounter: new	60	\gls@doclearpage: replaced \@ifundefined with \ifcsundef	28
\@gls@setupsort@def: new ...	10	\glsadd: added \@gls@saveentrycounter	116
\@gls@setupsort@standard: new	9	\GlsAddXdyCounters: new	30
\@gls@setupsort@use: new ...	10	\glentrycounterlabel: new	139
\@gls@xdy@locationlist: new	33	\glentryitem: new	139
\@glslink: replaced \@ifundefined with \ifcsundef	69	\glentrylong: new	114
\@glsnextpages: new	137	\glentrylong: new	113
\@makeglossary: Added check for savewrites	124	\glentrylongpl: new	114
\@set@glo@numformat: added 4th argument	62	\glentrylongpl: new	114
\@wrglossary: modified to take into account savewrites	127	\glentryshort: new	113
\@xdyattributelist: new	30	\glentryshort: new	113
General: added prefix to hyperlink	145	\glentryshortpl: new	113
etoolbox now loaded	4	\glentryshortpl: new	113
replaced \@ifundefined with \ifcsundef	19, 58, 136	\glsgetgrouptitle: replaced \@ifundefined with \ifcsundef	141
\acrfootnote: new	152	\glshyperlink: changed default from \glentryname to \glentrytext	116
\ACRfull: added starred version	148	\glshypernumber: replaced \@ifundefined with \ifcsundef	143
\Acrfull: added starred version	148	\glsnumberformat: replaced \@ifundefined with \ifcsundef	25
\acrfull: added starred version	147	\glsrefentry: new	139
\ACRfullpl: added starred version	150		
\Acrfullpl: added starred version	149		
\acrfullpl: added starred version	149		
\acrlinkfootnote: new	152		
\acrnolinkfootnote: new ...	152		

<code>\glsresetsubentrycounter:</code>	<code>counter:replaced \@ifundefined</code>
new 138	with <code>\ifcsundef</code> 45
<code>\glsseeitem:</code> hyperlink uses	<code>\printglossary:</code> added
<code>\glsseeitemformat</code> instead	<code>\currentglossary</code> 134
of <code>\glsentryname</code> 132	added <code>\glsnextpages</code> 134
<code>\glsseeitemformat:new</code> 132	make toctitle default to title .. 134
<code>\glssortnumberfmt:new</code> 9	replaced <code>\@ifundefined</code> with
<code>\glsstepentry:new</code> 138	<code>\ifcsundef</code> 133, 135
<code>\glsstepsubentry:new</code> 138	<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>
<code>\glssubentrycounterlabel:</code>	expanded options link op-
new 139	tions 152
<code>\glssubentryitem:new</code> 139	<code>\setentrycounter:</code> added op-
<code>theglossary:replaced \@ifundefined</code>	tional argument 142
with <code>\ifcsundef</code> 139	<code>\showacronymlists:new</code> 169
<code>short:new</code> 46	<code>\showglocounter:new</code> 166
<code>shortplural:new</code> 46	<code>\showglodesc:new</code> 168
<code>\ifglossaryexists:</code> re-	<code>\showglodescplural:new</code> ... 168
placed <code>\@ifundefined</code> with	<code>\showglofirst:new</code> 166
<code>\ifcsundef</code> 39	<code>\showglofirstpl:new</code> 166
<code>\ifglsentryexists:</code> re-	<code>\showgloflag:new</code> 168
placed <code>\@ifundefined</code> with	<code>\showgloindex:new</code> 168
<code>\ifcsundef</code> 39	<code>\showglolevel:new</code> 165
<code>\istfile:deprecated</code> 126	<code>\showgloname:new</code> 167
<code>glossaryentry:new</code> 137	<code>\showgloparent:new</code> 165
<code>glossarysubentry:new</code> 138	<code>\showgloplural:new</code> 166
<code>\newglossary:added \@gls@defsortcount</code>	<code>\showglosort:new</code> 168
..... 42	<code>\showglossaries:new</code> 169
replaced <code>\@ifundefined</code> with	<code>\showglossarycounter:new</code> . 169
<code>\ifcsundef</code> 41	<code>\showglossaryentries:new</code> . 170
<code>\newglossaryentry:</code> added	<code>\showglossaryin:new</code> 169
<code>\@gls@defsort</code> 51	<code>\showglossaryout:new</code> 169
added short and long keys 48	<code>\showglossarytitle:new</code> ... 169
replaced <code>\@ifundefined</code> with	<code>\showglosymbol:new</code> 168
<code>\ifcsundef</code> 49	<code>\showglosymbolplural:new</code> . 168
replaced <code>\DeclareRobustCommand</code>	<code>\showglotext:new</code> 166
with <code>\newrobustcmd</code> 47	<code>\showglotype:new</code> 166
<code>\newglossarystyle:</code> re-	<code>\showglouseri:new</code> 167
placed <code>\@ifundefined</code> with	<code>\showglouserii:new</code> 167
<code>\ifcsundef</code> 142	<code>\showglouseriii:new</code> 167
<code>entrycounter:new</code> 8	<code>\showglouseriv:new</code> 167
<code>entrycounterwithin:new</code> 8	<code>\showglouserv:new</code> 167
<code>\oldacronym:replaced \@ifundefined</code>	<code>\showglouservi:new</code> 167
with <code>\ifcsundef</code> 146	<code>subentrycounter:new</code> 9
<code>compatible-2.07: compatible-</code>	<code>\writeist:</code> added xindy-only
2.07 option added 18	macro definitions to glossary
<code>long:new</code> 46	open tag 120
<code>longplural:new</code> 46	modified to support new for-
<code>nonumberlist:now boolean</code> ... 45	mat 118
<code>sort:new</code> 9	

3.01		
General: made robust	73
\ACRfull: made robust	148
\Acrfull: made robust	148
\acrfull: made robust	147
\acrfullformat:	removed	
\acronymfont as it should already be set in the second argument.	148
\ACRfullpl: made robust	149
\Acrfullpl: made robust	149
\acrfullpl: made robust	149
\ACRlong: made robust	107
\Acrlong: made robust	106
\acrlong: made robust	106
\ACRlongpl: made robust	109
\Acrlongpl: made robust	108
\acrlongpl: made robust	108
\ACRshort: made robust	104
\Acrshort: made robust	103
\acrshort: made robust	102
\ACRshortpl: made robust	105
\Acrshortpl: made robust	105
\acrshortpl: made robust	104
\Gls: made robust	72
\glsadd: made robust	116
\glsaddall: made robust	116
\GLSdesc: made robust	88
\Glsdesc: made robust	88
\glsdesc: made robust	87
\GLSdescplural: made robust	..	90
\Glsdescplural: made robust	..	89
\glsdescplural: made robust	..	89
\glsfirst: made robust	82
\GLSfirstplural: made robust	..	85
\Glsfirstplural: made robust	..	85
\glsfirstplural: made robust	..	84
\glslink: made robust	59
\GLSname: made robust	87
\Glsname: made robust	86
\glsname: made robust	86
\GLSpl: made robust	77
\Glspl: made robust	76
\glspl: made robust	74
\GLSplural: made robust	84
\GLSsymbol: made robust	91
\Glsymbol: made robust	91
\glsymbol: made robust	90
\GLSsymbolplural: made robust	93	
\Glsymbolplural: made robust	92	
\glsymbolplural: made robust	92	
\Glstext: made robust	81
\glstext: made robust	80
\GLSuseri: made robust	94
\Glsuseri: made robust	94
\glsuseri: made robust	93
\GLSuserii: made robust	96
\Glsuserii: made robust	95
\glsuserii: made robust	95
\GLSuseriii: made robust	97
\Glsuseriii: made robust	97
\glsuseriii: made robust	96
\GLSuseriv: made robust	99
\Glsuseriv: made robust	98
\glsuseriv: made robust	98
\GLSuserv: made robust	100
\Glsuserv: made robust	100
\glsuserv: made robust	99
\GLSuservi: made robust	102
\Glsuservi: made robust	101
\glsuservi: made robust	101
\glswritefiles: added check for empty glossaries	126
3.02		
\@@do@wrglossary: changed		
\@glslocref to \theglentrycounter	130
\@do@wrglossary: changed		
\@do@wr@glossary to test for indexonlyfirst option; put old		
\@do@wr@glossary code into		
\@@do@wrglossary	128
\@gls@missingnumberlist:		
new	47
\@wrglossary: added check for glossary file defined	127
General: added check for polyglossia	19
reversed order of package check		23
savenumberlist: new	7
ucmark: new	8
\gls@save@numberlist: new	..	133
\glsdisplaynumberlist: new	114	
\glentrycounter: set default value	60
\Glsentryfull: fixed bug (replaced \glentryshortpl with \glentryshort)	114

<code>\glsentryfullpl</code> : fixed bug (replaced <code>\glsentryshort</code> with <code>\glsentryshortpl</code>)	114	<code>longragged3col</code> : added check for <code>glsnogroupskip</code>	188
<code>\glsentrynumberlist</code> : new ..	114	<code>nopostdot</code> : new	8
<code>\glsmoveentry</code> : new	52	<code>\printglossary</code> : allow title to override default <code>toctitle</code>	133
<code>\glsnumlistlastsep</code> : new ...	115	<code>tree</code> : added check for <code>glsnogroupskip</code>	209
<code>\glsnumlistsep</code> : new	115	<code>treenoname</code> : added check for <code>glsnogroupskip</code>	211
<code>\glsresetsubentrycounter</code> : new	138	<code>super</code> : added check for <code>glsnogroupskip</code>	196
<code>\glswritefiles</code> : added check for existence of token in case <code>\makeglossaries</code> has been omitted	126	<code>super3col</code> : added check for <code>glsnogroupskip</code>	197
<code>\ifglshaschildren</code> : new	40	<code>super4col</code> : added check for <code>glsnogroupskip</code>	199
<code>\ifglshasparent</code> : new	40	<code>superragged</code> : added check for <code>glsnogroupskip</code>	202
<code>\makeglossaries</code> : added list parser	125	<code>superragged3col</code> : added check for <code>glsnogroupskip</code>	204
<code>indexonlyfirst</code> : new	16		
<code>\newglossaryentry</code> : added numberlist element	51		
<code>\printglossary</code> : add a way to fetch current entry label ...	134	<code>3.04</code>	
<code>\renewglossarystyle</code> : new ..	143	<code>\@@do@wrglossary</code> : changed <code>\theglsentrycounter</code> back to <code>\@glslocref</code>	130
<code>\showglossaryentries</code> : fixed misspelt command	170	modified to compensate for possible incorrect page number	129
<code>\SmallNewAcronymDef</code> : fixed broken short and long plural	158	<code>\@gls@escbsdq</code> : unsanitize <code>\gls@numberpage</code> , <code>\gls@alphpage</code> , <code>\gls@Alphpage</code> and <code>\gls@romanpage</code>	63
<code>3.03</code>		<code>General</code> : Added check for doc package	4
<code>\@gls@sanitizesort</code> : new	15	added <code>datatool-base</code> as a required package	3
<code>\@gls@setupsort@standard</code> : used <code>\@gls@sanitizesort</code> ..	9	added local key	59
<code>General</code> : allow title to set <code>toctitle</code>	136	<code>\changes</code> : new	4
<code>altlongragged4col</code> : added check for <code>glsnogroupskip</code> ..	190	<code>\gls@Alphpage</code> : new	128
<code>altsuperragged4col</code> : added check for <code>glsnogroupskip</code> ..	205	<code>\gls@alphpage</code> : new	128
<code>alttree</code> : added check for <code>glsnogroupskip</code>	214	<code>\gls@disablepagerefexpansion</code> : new	128
<code>index</code> : added check for <code>glsnogroupskip</code>	208	<code>\gls@numberpage</code> : new	128
<code>nogroupskip</code> : new	8	<code>\gls@protected@pagefmts</code> : new	128
<code>long</code> : added check for <code>glsnogroupskip</code>	181	<code>\gls@romanpage</code> : new	128
<code>long3col</code> : added check for <code>glsnogroupskip</code>	182	<code>\glsdefmain</code> : added check for doc package	11
<code>long4col</code> : added check for <code>glsnogroupskip</code>	183	<code>\glsorg@endtheglossary</code> : new .	5
<code>longragged</code> : added check for <code>glsnogroupskip</code>	187	<code>\glsorg@glossary</code> : new	4
		<code>\glsorg@theglossary</code> : new	5

\glsorg@wrglossary: new	4	\gls@Romanpage: new	128
altlist: replaced \newline with paragraph break	178	\GlsDeclareNoHyperList: new	14
\PrintChanges: new	5	\glsgetgrouplabel: fixed bug (typo in \equal)	142
\printglossary: Moved aux write to end of document to prevent unwanted whatsit oc- curring here.	135	\nopostdesc: made robust	23
3.05		nohypertypes: new	14
\@@do@wrglossary: add Roman case. Fixed bugs in the else statements	129	3.06	
\@gls@link: added check for “no- hypertypes”	60	\@xdy@main@language: Changed back to using \language name	17
\@gls@nohyperlist: new	14	\findrootlanguage: Obsoleted	36
mcolalmtree: replaced ‘2’ with \glsmcols	194	3.07	
mcolindex: replaced ‘2’ with \glsmcols	191	\@gls@link: fixed bug that failed to find entry in list	60
mcoltree: replaced ‘2’ with \glsmcols	192	\glossarypreamble: modified to work with \setglossarypreamble	26
mcoltreename: replaced ‘2’ with \glsmcols	193	\gls@doclearpage: added check for openright	28
\gls@protected@pagefmts: added Roman to list	128	\glspostdescription: Added spacefactor code	8
		\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	155
		\setglossarypreamble: new ..	26

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
\@do@wrglossary	128
\@glossarysec	5
\@glossaryseclabel	6
\@glossarysecstar	6
\@ACRlong	237
\@ACRshort	236
\@Acrlong	236
\@Acrshort	236
\@GLS@	73, 231
\@GLSpl	78
\@GLSpl@	234
\@Gls@	72, 230
\@Glspl@	76, 233
\@acrlong	236
\@acrshort	235
\@addtoacronymlists	12
\@delimN	144
\@delimR	144
\@disable@onlypremakeg	20
\@disable@premakecs	20
\@disabled@glsgladdxdycounters ..	31
\@do@seeglossary	131
\@do@wrglossary	128, 216
\@glo@seeautonumberlist	7
\@glo@storeentry	53
\@glo@types	40
\@glossary	127
\@glossary@default@style	6
\@glossaryentryfield	53, 237
\@glossarysection	28
\@glossarysubentryfield ..	53, 237
\@gls	70
\@gls@	71, 229
\@gls@@link	59
\@gls@addpredefinedattributes ..	32
\@gls@checkactual	67
\@gls@checkbar	66
\@gls@checkescactual	65
\@gls@checkescbar	65
\@gls@checkesclevel	66
\@gls@checkescquote	64
\@gls@checklevel	67
\@gls@checkmkidxchars	63
\@gls@checkquote	64
\@gls@codepage	37
\@gls@counterwithin	9
\@gls@escbsdq	62
\@gls@fixbraces	131
\@gls@getcounter	42
\@gls@getcounterprefix	130
\@gls@hypergroup	174
\@gls@ifinlist	30
\@gls@link	59
\@gls@loadlist	7
\@gls@loadlong	7
\@gls@loadsuper	7
\@gls@loadtree	8
\@gls@missingnumberlist	47
\@gls@noaccess	223
\@gls@nohyperlist	14
\@gls@onlypremakeg	19
\@gls@pl@	232
\@gls@renewglossary	127
\@gls@sanitizedesc	14
\@gls@sanitizename	14
\@gls@sanitizesort	15
\@gls@sanitizesymbol	14
\@gls@saveentrycounter	60
\@gls@setcounter	42
\@gls@setupsort@def	10
\@gls@setupsort@standard	9
\@gls@setupsort@use	10
\@gls@tmpb	64
\@gls@toc	29
\@gls@updatechecked	64
\@gls@xdy@Lclass@Alpha-page-numbers	34
\@gls@xdy@Lclass@Appendix-page-numbers	34
\@gls@xdy@Lclass@Roman-page-numbers	33
\@gls@xdy@Lclass@alpha-page-numbers	34
\@gls@xdy@Lclass@arabic-page-numbers	34

\@gls@xdy@Lclass@arabic-section-numbers	A	
.....	34	\Ac 164
\@gls@xdy@Lclass@roman-page-numbers		\ac 164
.....	33	access (key) 222
\@gls@xdy@locationlist	33	accsupp package 221
\@gls@xdy@checkbackslash	68	\accsuppglossaryentryfield .. 238
\@gls@xdy@checkquote	68	\accsuppglossarysubentryfield 238
\@gls@Alphacompositor	24, 34	\Acf 164
\@gls@acronymlists	12	\acf 164
\@gls@defaultplural	47	\Acfp 164
\@gls@defaultsort	47	\acfp 164
\@gls@disp	79	\Acl 164
\@gls@firstletter	117	\acl 163
\@gls@hypernumber	144	\Aclp 164
\@gls@link	69	\aclp 164
\@gls@minrange	118	\Acp 164
\@gls@nextpages	137	\acp 164
\@gls@noname	47	\acrfootnote 152
\@gls@nonextpages	137	\ACRfull 148
\@gls@openfile	124	\Acrfull 148
\@gls@order	17	\acrfull 147, 148
\@gls@pl@	75	\acrfullformat 148
\@gls@target	69	\ACRfullpl 149
\@gls@widestname	212	\Acrfullpl 149
\@ist@filename	24	\acrfullpl 149
\@make@glossary	124	\acrlinkfootnote 152
\@new@glossary	42	\acrlinkfullformat 148
\@new@glossaryentryposthook	52	\ACRlong 107
\@new@glossaryentryprehook	52	\Acrlong 106
\@nopostdesc	23	\acrlong 106
\@only@premakeg	20	\ACRlongpl 109
\@p@@glossarysection	28	\Acrlongpl 108
\@set@glo@numformat	61, 217	\acrlongpl 108
\@sgls	70, 79	\acrnameformat 150, 156
\@sgls@link	59	\acrno linkfootnote 152
\@wrglossary	127	acronym (option) 12
\@xdy@main@language	17	\acronymfont . 150, 153, 156, 157, 159
\@xdy@attributelist	30	acronymlists (option) 13
\@xdy@attributes	29	\acronymname 20
\@xdy@language	37	\acronymtype 12, 146
\@xdy@lettergroups	38	\acrpluralsuffix 147
\@xdy@locationclassorder	35	\ACRshort 104
\@xdy@locref	30	\Acrshort 103
\@xdy@requiredstyles	36	\acrshort 102
\@xdy@sortrules	36	\ACRshortpl 105
\@xdy@useralphabets	32	\Acrshortpl 105
\@xdy@userlocationdefs	34	\acrshortpl 104
\@xdy@userlocationnames	34	\Acs 163
		\acs 163

<code>\Acsp</code>	163	<code>\defglsdisplay</code>	41, 43, 44, 57, 58
<code>\acsp</code>	163	<code>\defglsdisplayfirst</code>	41, 43, 44, 57, 58
<code>\addglossarytocaptions</code>	21	<code>\DefineAcronymSynonyms</code>	163
<code>\addto</code>	21	<code>\delimN</code>	25, 143
<code>align (environment)</code>	60	<code>\delimR</code>	25, 143
<code>altlist (style)</code>	178	<code>description (environment)</code>	177
<code>altlistgroup (style)</code>	178	<code>description (key)</code>	43
<code>altlisthypergroup (style)</code>	179	<code>description (option)</code>	16
<code>atlong4col (style)</code>	184	<code>descriptionaccess (key)</code>	223
<code>atlong4colborder (style)</code>	185	<code>\DescriptionDUANewAcronymDef</code>	154
<code>atlong4colheader (style)</code>	185	<code>\DescriptionFootnoteNewAcronymDef</code>	152, 239
<code>atlong4colheaderborder (style)</code>	185	<code>\descriptionname</code>	21
<code>atlongragged4col (style)</code>	189	<code>\DescriptionNewAcronymDef</code>	155, 239
<code>atlongragged4colborder (style)</code>	190	<code>descriptionplural (key)</code>	43
<code>atlongragged4colheader (style)</code>	190	<code>descriptionpluralaccess (key)</code> .	223
<code>atlongragged4colheaderborder</code>		<code>doc package</code>	4, 11
<code>(style)</code>	190	<code>dua (option)</code>	17
<code>\altnewglossary</code>	42	<code>\DUANewAcronymDef</code>	160
<code>altsuper4col (style)</code>	200		
<code>altsuper4colborder (style)</code>	200	E	
<code>altsuper4colheader (style)</code>	200	<code>entrycounter (option)</code>	8
<code>altsuper4colheaderborder (style)</code>	201	<code>entrycounterwithin (option)</code>	8
<code>altsuperragged4col (style)</code>	205	<code>\entryname</code>	20
<code>altsuperragged4colborder (style)</code>	206	<code>environments:</code>	
<code>altsuperragged4colheader (style)</code>	206	<code>align</code>	60
<code>altsuperragged4colheaderborder</code>		<code>description</code>	177
<code>(style)</code>	206	<code>longtable</code>	7, 165, 180–191
<code>alttree (style)</code>	212	<code>multicols</code>	191
<code>alttreegroup (style)</code>	214	<code>supertabular</code>	7, 165, 195–206
<code>alttreehypergroup (style)</code>	214	<code>theglossary</code>	4, 5,
<code>amsgen package</code>	4, 58	25, 26, 139, 140, 142, 193, 209–212	
<code>\andname</code>	21	<code>theindex</code>	207
<code>array package</code>	186, 201	<code>equation counter</code>	60, 61
<code>article class</code>	130	<code>etoolbox package</code>	4, 170
B		F	
<code>babel package</code>	19, 20, 22, 37, 243	<code>file types</code>	
		<code>.aux</code>	135
C		<code>.glo</code>	53
<code>\capitalisewords</code>	172	<code>.ist</code>	117, 123, 124
<code>\changes</code>	4	<code>.toc</code>	29
<code>compatible-2.07 (option)</code>	18	<code>.xdy</code>	24
<code>counter (key)</code>	45	<code>glo</code>	170
<code>counter (option)</code>	13	<code>\findrootlanguage</code>	36
<code>\CustomAcronymFields</code>	162	<code>first (key)</code>	44
<code>\CustomNewAcronymDef</code>	162	<code>firstaccess (key)</code>	222
		<code>\firstacronymfont</code>	150
D		<code>firstplural (key)</code>	44
<code>\DeclareAcronymList</code>	12	<code>firstpluralaccess (key)</code>	222
<code>\DefaultNewAcronymDef</code> ...	151, 238		

altsuperragged4colheaderborder	206	longragged	186, 187
altsuperragged4colheaderborder	206	longragged	186
alttree	194, 211, 212, 214	longragged3col	187, 188
alttree	212	longragged3col	187
alttreegroup	214	longragged3colborder	188
alttreegroup	214	longragged3colborder	188
alttreehypergroup	214	longragged3colheader	188
alttreehypergroup	214	longragged3colheader	188
index	191, 207, 208	longragged3colheaderborder	189
index	207	longragged3colheaderborder	189
indexgroup	208	longraggedborder	187
indexgroup	208	longraggedborder	187
indexhypergroup	208	longraggedheader	187
indexhypergroup	208	longraggedheader	187
inline	175	longraggedheaderborder	187
list	6, 177–179	longraggedheaderborder	187
list	177	mcolalttree	194
listdotted	179, 180	mcolalttree	194
listdotted	179	mcolalttreegroup	194
listgroup	177, 178	mcolalttreegroup	194
listgroup	177	mcolalttreehypergroup	194
listhypergroup	178	mcolalttreehypergroup	194
listhypergroup	178	mcolindex	191
long	180–182, 186	mcolindex	191
long	180	mcolindexgroup	192
long3col	182	mcolindexgroup	191
long3col	182	mcolindexhypergroup	192
long3colborder	182	mcolindexhypergroup	192
long3colborder	182	mcoltree	192
long3colborder	182	mcoltree	192
long3colheader	182	mcoltreegroup	192
long3colheader	182	mcoltreehypergroup	192
long3colheaderborder	183	mcoltreehypergroup	192
long3colheaderborder	183	mcoltreenoname	193
long4col	183, 184	mcoltreenoname	193
long4col	183	mcoltreenonamegroup	193
long4colborder	184	mcoltreenonamegroup	193
long4colborder	184	mcoltreenonamehypergroup	193
long4colheader	184	mcoltreenonamehypergroup	193
long4colheader	184	sublistdotted	180
long4colheaderborder	184	super	195–197, 203
long4colheaderborder	184	super	195
longborder	181	super3col	197, 198
longborder	181	super3col	197
longheader	181	super3colborder	197
longheader	181	super3colborder	197
longheaderborder	181	super3colheader	197
longheaderborder	181	super3colheader	197
		super3colheaderborder	198

super3colheaderborder	198	glossary-mcols package	191
super4col	198–200	glossary-super package	
super4col	198		7, 180, 195, 201, 205
super4colborder	199	glossary-superragged package	201
super4colborder	199	glossary-tree package	8, 207
super4colheader	199	glossaryentry (counter)	137
super4colheader	199	glossaryentry counter	8, 138, 139
super4colheaderborder	199	\glossaryentryfield	44, 140, 142, 143
super4colheaderborder	199	\glossaryentrynumber	137
superborder	196	\glossaryentrynumbers	6, 25, 133, 135
superborder	196	\glossaryheader	140, 142
superheader	196	\glossarymark	8, 27
superheader	196	\glossaryname	20, 21
superheaderborder	196	\glossarypostamble	26, 142
superheaderborder	196	\glossarypreamble	26, 142
superragged	202, 203	\glossarysection	5, 26, 41
superragged	202	\glossarystyle	142, 165
superragged3col	203–205	glossarysubentry (counter)	138
superragged3col	203	glossarysubentry counter	9, 138, 139
superragged3colborder	204	\GLS	73
superragged3colborder	204	\Gls	72, 76, 170
superragged3colheader	204	\gls	4, 44, 56, 57, 59,
superragged3colheader	204		70, 73, 74, 80, 82–84, 86, 87, 89,
superragged3colheaderborder	205		90, 92, 93, 95, 96, 98, 99, 101, 139
superraggedborder	202	\gls@Alphpage	128
superraggedborder	202	\gls@alphpage	128
superraggedheader	203	\gls@checkisacronymlist	13
superraggedheader	203	\gls@codepage	18
superraggedheaderborder	203	\gls@disablepagerefexpansion	128
superraggedheaderborder	203	\gls@doclearpage	28
superraggedheaderborder	203	\gls@hypergroup prerun	173
superraggedright3colheaderborder	205	\gls@level	47
tree	192, 208–210, 212	\gls@numberpage	128
tree	208	\gls@protected@pagefmts	128
treegroup	192, 210	\gls@Romanpage	128
treegroup	209	\gls@romanpage	128
treehypergroup	210	\gls@save@numberlist	133
treehypergroup	210	\gls@suffixF	24
treenoname	193, 210, 211	\gls@suffixFF	25
treenoname	210	\gls@accessdisplay	229
treenonamegroup	211	\gls@sacccsupp	226
treenonamegroup	211	\gls@sadd	56, 116, 142
treenonamehypergroup	211	\gls@sadd options	
treenonamehypergroup	211	counter	116
		format	116, 143
glossary-hypernav package	117	\gls@saddall	56, 116
glossary-list package	6, 7, 177	\gls@saddall options	
glossary-long package	7, 180, 189, 195	types	116
glossary-longragged package	186	\GlsAddLetterGroup	38

<code>\GlsAddSortRule</code>	36	<code>\glstentryfull</code>	114
<code>\GlsAddXdyAlphabet</code>	32	<code>\Glsentryfullpl</code>	114
<code>\GlsAddXdyAttribute</code>	31, 215	<code>\glstentryfullpl</code>	114
<code>\GlsAddXdyCounters</code>	30, 215	<code>\glstentryitem</code>	139
<code>\GlsAddXdyLocation</code>	34, 216	<code>\Glsentrylong</code>	114
<code>\GlsAddXdyStyle</code>	36	<code>\glstentrylong</code>	113
<code>\glsclearprefix</code>	6	<code>\glstentrylongaccess</code>	226
<code>\glsclearpage</code>	29	<code>\Glsentrylongpl</code>	114
<code>\glsclosebrace</code>	117	<code>\glstentrylongpl</code>	114
<code>\glscpositor</code>	24, 34	<code>\glstentrylongpluralaccess</code> ..	226
<code>\glscounter</code>	13, 42	<code>\Glsentryname</code>	110
<code>\GlsDeclareNoHyperList</code>	14	<code>\glstentryname</code>	110, 132
<code>\glstdefaulttype</code>	11	<code>\glstentrynumberlist</code>	114
<code>\glstdefmain</code>	11	<code>\Glsentryplural</code>	111
<code>\GLSdesc</code>	88	<code>\glstentryplural</code>	111
<code>\Glsdesc</code>	88	<code>\glstentrypluralaccess</code>	225
<code>\glstdesc</code>	87, 88	<code>\Glsentryshort</code>	113
<code>\GLSdescplural</code>	90	<code>\glstentryshort</code>	113
<code>\Glsdescplural</code>	89	<code>\glstentryshortaccess</code>	226
<code>\glstdescplural</code>	89, 90	<code>\Glsentryshortpl</code>	113
<code>\glstdescriptionaccessdisplay</code>	228	<code>\glstentryshortpl</code>	113
<code>\glstdescriptionpluralaccessdisplay</code>	228	<code>\glstentryshortpluralaccess</code> ..	226
<code>\glstdescwidth</code>	180, 186, 195, 201	<code>\glstentrysort</code>	112
<code>\glstdisablehyper</code>	70	<code>\Glsentrysymbol</code>	111
<code>\glstdisp</code>	79	<code>\glstentrysymbol</code>	111
<code>\glstdisplay</code>	14, 43, 44, 57, 70	<code>\glstentrysymbolaccess</code>	225
<code>\glstdisplayfirst</code> ..	43, 44, 57, 58, 70	<code>\Glsentrysymbolplural</code>	111
<code>\glstdisplaynumberlist</code>	114	<code>\glstentrysymbolplural</code>	111
<code>\glstdoifexists</code>	39	<code>\glstentrysymbolpluralaccess</code> ..	225
<code>\glstdoifnoexists</code>	40	<code>\Glsentrytext</code>	110
<code>\glstdoparenifnotempty</code>	158	<code>\glstentrytext</code>	110, 132
<code>\glstenablehyper</code>	70	<code>\glstentrytextaccess</code>	225
<code>\glstentryaccess</code>	225	<code>\glstentrytype</code>	112
<code>\glstentrycounter</code>	60	<code>\Glsentryuseri</code>	112
<code>\glstentrycounterlabel</code>	139	<code>\glstentryuseri</code>	112
<code>\Glsentrydesc</code>	110	<code>\glstentryuserii</code>	112
<code>\glstentrydesc</code>	14, 110	<code>\glstentryuseriii</code>	112
<code>\glstentrydescaccess</code>	225	<code>\glstentryuseriii</code>	112
<code>\Glsentrydescplural</code>	110	<code>\Glsentryuseriv</code>	113
<code>\glstentrydescplural</code>	110	<code>\glstentryuseriv</code>	112
<code>\glstentrydescpluralaccess</code> ..	226	<code>\Glsentryuserv</code>	113
<code>\Glsentryfirst</code>	111	<code>\glstentryuserv</code>	113
<code>\glstentryfirst</code>	111	<code>\Glsentryuservi</code>	113
<code>\glstentryfirstaccess</code>	225	<code>\glstentryuservi</code>	113
<code>\Glsentryfirstplural</code>	112	<code>\GLSfirst</code>	83
<code>\glstentryfirstplural</code>	111	<code>\Glsfirst</code>	82
<code>\glstentryfirstpluralaccess</code> ..	225	<code>\glstfirst</code>	82
<code>\Glsentryfull</code>	114	<code>\glstfirstaccessdisplay</code>	227

<code>\GLSfirstplural</code>	85	<code>\glsnameaccessdisplay</code>	226
<code>\Glsfirstplural</code>	85	<code>\glsnamefont</code>	143
<code>\glsfirstplural</code>	84, 85	<code>\glsnavhyperlink</code>	173
<code>\glsfirstpluralaccessdisplay</code>	227	<code>\glsnavhypertarget</code>	173
<code>\glsgetgrouplabel</code>	142	<code>\glsnavigation</code>	174
<code>\glsgetgrouptitle</code>	117, 141	<code>\glsnextpages</code>	137
<code>\glsgroupheading</code>	141, 142	<code>\glsnonextpages</code>	137
<code>\glsgroupskip</code>	141, 142, 177	<code>\glsnoxindywarning</code>	29
<code>\glshyperlink</code>	116	<code>\glsnumberformat</code>	25
<code>\glshypernavsep</code>	174	<code>\glsnumbersgroupname</code> ..	21, 117, 141
<code>\glshypernumber</code>	25, 143	<code>\glsnumlistlastsep</code>	115
<code>\glsIfListOfAcronyms</code>	12	<code>\glsnumlistsep</code>	115
<code>\glsinlinedescformat</code>	176	<code>\glsopenbrace</code>	117
<code>\glsinlinedopostchild</code> ...	175, 176	<code>\glsorder</code>	17
<code>\glsinlineemptydescformat</code> ...	176	<code>\glsorg@endtheglossary</code>	5
<code>\glsinlinenameformat</code>	176	<code>\glsorg@glossary</code>	4
<code>\glsinlineparentchildseparator</code>	176	<code>\glsorg@theglossary</code>	5
<code>\glsinlinepostchild</code>	176	<code>\glsorg@wrglossary</code>	4
<code>\glsinlineseparator</code>	176	<code>\glspagelistwidth</code>	180, 186, 195, 202
<code>\glsinlinesubdescformat</code>	177	<code>\glspar</code>	23
<code>\glsinlinesubnameformat</code>	177	<code>\GLSpl</code>	77
<code>\glsinlinesubseparator</code>	176	<code>\Glspl</code>	76, 170
<code>\glskeylisttok</code>	150	<code>\glspl</code>	56, 57, 74, 76, 77
<code>\glslabeltok</code>	150	<code>\GLSplural</code>	84
<code>\glslink</code> ..	56, 57, 59, 70, 116, 142, 143	<code>\Glsplural</code>	83
<code>\glslink options</code>		<code>\glsplural</code>	83, 84
counter	58, 70, 170	<code>\glspluralaccessdisplay</code>	227
format	58, 70, 143	<code>\glspluralsuffix</code>	21, 44
hyper	58, 70	<code>\glspostdescription</code>	8
local	59	<code>\glspostinline</code>	176
<code>\glslistdottedwidth</code>	179	<code>\glsquote</code>	117
<code>\glslocalreset</code>	55	<code>\glsrefentry</code>	139
<code>\glslocalresetall</code>	56	<code>\glsreset</code>	55
<code>\glslocalunset</code>	55	<code>\glsresetall</code>	56
<code>\glslocalunsetall</code>	56	<code>\glsresetentrylist</code>	137
<code>\glslongaccessdisplay</code>	229	<code>\glsresetsubentrycounter</code>	138
<code>\glslongaccesskey</code>	241	<code>\glssee</code>	131
<code>\glslongkey</code>	147	<code>\glsseeformat</code>	119, 131
<code>\glslongpluralaccessdisplay</code> ..	229	<code>\glsseeitem</code>	132
<code>\glslongpluralaccesskey</code>	241	<code>\glsseeitemformat</code>	132
<code>\glslongpluralkey</code>	147	<code>\glsseelastsep</code>	132
<code>\glslongtok</code>	150	<code>\glsseelist</code>	131
<code>\glsmakefirsttuc</code>	171	<code>\glsseesep</code>	132
<code>\glsmcols</code>	191	<code>\glsSetAlphaCompositor</code>	24
<code>\glsmoveentry</code>	52	<code>\glsSetCompositor</code>	24
<code>\GLSname</code>	87	<code>\glsSetSuffixF</code>	25
<code>\Glsname</code>	86	<code>\glsSetSuffixFF</code>	25
<code>\glsname</code>	86, 87	<code>\glssettoctitle</code>	20
		<code>\glssetwidest</code>	211

shortplural	113, 223	entrycounter	137
shortpluralaccess	226, 228	true	8
sort	14, 15, 43, 112, 140, 141	entrycounter	8
symbol		entrycounterwithin	8
14, 15, 43, 44, 57, 90, 111, 153,		footnote	71, 72, 74, 75,
154, 156, 159, 183, 198, 222, 223		77, 78, 80, 153, 155–157, 230–235	
symbolaccess	225, 227	footnote	16
symbolplural	92, 222	hyperfirst	
symbolpluralaccess	225, 228	false	71, 72, 74, 75, 77, 78, 80
text 44, 57, 70, 80, 110, 153, 156, 222		hyperfirst	16
textaccess	225, 227	indexonlyfirst	263
type	11, 45, 56, 112	indexonlyfirst	16
user1	93, 112, 223	makeindex	120, 170
user2	95, 112	nogroupskip	8
user3	96, 112	nohypertypes	14
user4	98, 112	nolist	165
user5	99, 113	nolist	7
user6	101, 113, 223	nolong	165, 180
\newglossarystyle	142	nolong	7
nogroupskip (option)	8	nomain	11, 12
nohypertypes (option)	14	nonumberlist	7
\noist	123, 170, 221	nonumberlist	7
nolist (option)	7	noposteddot	8
nolong (option)	7	nostyles	8
nonumberlist (key)	45	nosuper	165
nonumberlist (option)	7	nosuper	7
\nopostdesc	23	notree	165
noposteddot (option)	8	notree	8
nostyles (option)	8	numberedsection	6
nosuper (option)	7	numberline	5
notree (option)	8	numberline	5
numberedsection (option)	6	order	17
numberline (option)	5	sanitize	14, 15, 43, 110, 111
		sanitize	15
O		savenumberlist	7
\oldacronym	146	savewrites	18, 261
order (option)	17	false	124
		true	126
P		savewrites	18
package options:		section	5, 27
acronym 11, 12, 20, 135, 146, 147		section	5
acronym	12	seeautonumberlist	7
acronymlists	13	shotcuts	17
compatible-2.07	18	smallcaps	16
counter	13	smaller	17
counter	13	sort	
description	156	def	9
description	16	standard	9
dua	155, 156	use	9
dua	17		

sort	9	\SetDefaultAcronymStyle	151
style	6, 165	\SetDescriptionAcronymDisplayStyle	155
style	6	\SetDescriptionAcronymStyle ..	156
subentrycounter	138	\SetDescriptionDUAACronymDisplayStyle	154
subentrycounter	9	\SetDescriptionDUAACronymStyle	154
toc	5	\SetDescriptionFootnoteAcronymDisplayStyle	152
true	5	\SetDescriptionFootnoteAcronymStyle	153
toc	5	\SetDUADisplayStyle	160
translate	16	\SetDUASStyle	160
translate	16	\setentrycounter	142
ucmark	8	\SetFootnoteAcronymDisplayStyle	156
xindy	17, 120, 170	\SetFootnoteAcronymStyle	157
\pagelistname	21	\setglossarypreamble	26
parent (key)	45	\setglossarysection	28
\phantomsection	26, 28	\SetSmallAcronymDisplayStyle ..	158
plural (key)	44	\SetSmallAcronymStyle	159
pluralaccess (key)	222	\setStyleFile	23
polyglossia package	19, 21	short (key)	46
\PrintChanges	5	shortaccess (key)	223
\printglossaries		shortplural (key)	46
.. 11, 26, 40, 43, 125, 133, 136, 173		shortpluralaccess (key)	223
\printglossary		shotcuts (option)	17
26, 41, 125, 133, 135, 136, 172, 173		\showacronymlists	169
\printglossary options		\showglocounter	166
nonumberlist	136	\showglodesc	168
numberedsection	136	\showglodescaccess	242
style	136	\showglodescplural	168
title	136	\showglodescpluralaccess	242
toctitle	136	\showglofirst	166
type	11, 132, 136	\showglofirstaccess	242
\protect	14	\showglofirststpl	166
		\showglofirststpluralaccess ...	242
		\showgloflag	168
		\showgloindex	168
		\showglolevel	165
		\showglolongaccess	243
		\showglolongpluralaccess	243
		\showgloname	167
		\showglonameaccess	241
		\showgloparent	165
		\showgloplural	166
		\showglopluralaccess	242
		\showgloshortaccess	242
R			
\renewglossarystyle	143		
\roman	33		
S			
sanitize (option)	15		
savenumberlist (option)	7		
savewrites (option)	18		
section (option)	5		
see (key)	45		
seeautonumberlist (option)	7		
\seename	21		
\SetAcronymLists	13		
\SetAcronymStyle	12, 161		
\SetCustomDisplayStyle	162		
\SetCustomStyle	162		
\SetDefaultAcronymDisplayStyle	150		

<code>\showgloshortpluralaccess</code> ...	242	<code>superraggedheaderborder (style)</code>	203
<code>\showglosort</code>	168	<code>superraggedright3colheaderborder</code>	
<code>\showglossaries</code>	169	(style)	205
<code>\showglossarycounter</code>	169	<code>supertabular (environment)</code>	
<code>\showglossaryentries</code>	170	7, 165, 195–206
<code>\showglossaryin</code>	169	<code>supertabular package</code> ...	7, 165, 195, 201
<code>\showglossaryout</code>	169	<code>symbol (key)</code>	44
<code>\showglossarytitle</code>	169	<code>symbolaccess (key)</code>	222
<code>\showglosymbol</code>	168	<code>\symbolname</code>	21
<code>\showglosymbolaccess</code>	242	<code>symbolplural (key)</code>	44
<code>\showglosymbolplural</code>	168	<code>symbolpluralaccess (key)</code>	222
<code>\showglosymbolpluralaccess</code> ..	242		
<code>\showglotext</code>	166	T	
<code>\showglotextaccess</code>	242	<code>text (key)</code>	44
<code>\showglotype</code>	166	<code>textaccess (key)</code>	222
<code>\showglouseri</code>	167	<code>\theequation</code>	130
<code>\showglouserii</code>	167	<code>theglossary (environment)</code> ..	4, 5,
<code>\showglouseriii</code>	167	25, 26, 139, 140, 142, 193, 209–212	
<code>\showglouseriv</code>	167	<code>\theHequation</code>	130
<code>\showglouserv</code>	167	<code>theindex (environment)</code>	207
<code>\showglouservi</code>	167	<code>toc (option)</code>	5
<code>smallcaps (option)</code>	16	<code>\translate</code>	21
<code>smaller (option)</code>	17	<code>translate (option)</code>	16
<code>\SmallNewAcronymDef</code>	158, 240	<code>translator package</code>	
<code>sort (key)</code>	43	19, 21, 22, 133, 243, 252–255
<code>sort (option)</code>	9	<code>tree (style)</code>	208
<code>style (option)</code>	6	<code>treegroup (style)</code>	209
<code>subentrycounter (option)</code>	9	<code>treehypergroup (style)</code>	210
<code>\subitem</code>	207	<code>treenoname (style)</code>	210
<code>sublistdotted (style)</code>	180	<code>treenonamegroup (style)</code>	211
<code>\subsubitem</code>	207	<code>treenonamehypergroup (style)</code> ...	211
<code>super (style)</code>	195	<code>type (key)</code>	45
<code>super3col (style)</code>	197		
<code>super3colborder (style)</code>	197	U	
<code>super3colheader (style)</code>	197	<code>ucmark (option)</code>	8
<code>super3colheaderborder (style)</code> ..	198	<code>user1 (key)</code>	45
<code>super4col (style)</code>	198	<code>user2 (key)</code>	46
<code>super4colborder (style)</code>	199	<code>user3 (key)</code>	46
<code>super4colheader (style)</code>	199	<code>user4 (key)</code>	46
<code>super4colheaderborder (style)</code> ..	199	<code>user5 (key)</code>	46
<code>superborder (style)</code>	196	<code>user6 (key)</code>	46
<code>superheader (style)</code>	196		
<code>superheaderborder (style)</code>	196	W	
<code>superragged (style)</code>	202	<code>\warn@nomakeglossaries</code>	124
<code>superragged3col (style)</code>	203	<code>\warn@noprintglossary</code>	133
<code>superragged3colborder (style)</code> ..	204	<code>\writeist</code> .	23, 30, 32, 35, 118, 215, 217
<code>superragged3colheader (style)</code> ..	204		
<code>superraggedborder (style)</code>	202	X	
<code>superraggedheader (style)</code>	203	<code>\xcapitalisewords</code>	172
		<code>\xglaccsupp</code>	226
		<code>xindy</code>	256

xindy	17, 18, 23, 24, 29, 32,	\xmakefirstuc	172
	34, 36–38, 53, 54, 68, 117–119,		
	129, 130, 135, 140, 141, 170, 216	xspace package	4, 146