

Documented Code For glossaries v4.25

Nicola L.C. Talbot

Dickimaw Books

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

2016-06-09

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 v4.25: 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.

The user level commands described in the user manual ([glossaries-user.pdf](#)) may be considered “future-proof”. Even if they become deprecated, they should still work for old documents (although they may not work in a document that also contains new commands introduced since the old commands were deprecated, and you may need to specify a compatibility mode).

The internal commands in *this* document that aren’t documented in the *user manual* should not be considered future-proof and are liable to change. If you want a new user level command, you can post a feature request at <http://www.dickimaw-books.com/feature-request.html>. If you are a package writer wanting to integrate your package with glossaries, it’s better to request a new user level command than to hack these internals.

Contents

1	Main Package Code	4
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	30
1.4	Xindy	40
1.5	Loops and conditionals	49
1.6	Defining new glossaries	55
1.7	Defining new entries	60
1.8	Resetting and unsetting entry flags	85
1.9	Keeping Track of How Many Times an Entry Has Been Unset	88
1.10	Loading files containing glossary entries	93
1.11	Using glossary entries in the text	93
1.12	Adding an entry to the glossary without generating text	153
1.13	Creating associated files	155
1.14	Writing information to associated files	173
1.15	Glossary Entry Cross-References	180
1.16	Displaying the glossary	182
1.17	Acronyms	211
1.18	Predefined acronym styles	215
1.19	Predefined Glossary Styles	247
1.20	Debugging Commands	247
1.21	Compatibility with version 2.07 and below	253
2	Prefix Support (glossaries-prefix Code)	254
3	Glossary Styles	261
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	261
3.2	In-line Style (glossary-inline.sty)	263
3.3	List Style (glossary-list.sty)	265
3.4	Glossary Styles using longtable (the glossary-long package)	269
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	275
3.6	Glossary Styles using longtable (the glossary-longragged package)	279
3.7	Glossary Styles using multicol (glossary-mcols.sty)	285
3.8	Glossary Styles using supertabular environment (glossary-super package)	291
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	297
3.10	Tree Styles (glossary-tree.sty)	303

4 Backwards Compatibility	313
4.1 glossaries-compatible-207	313
4.2 glossaries-compatible-307	319
5 Accessibility Support (glossaries-accsupp Code)	333
5.1 Defining Replacement Text	334
5.2 Accessing Replacement Text	337
5.3 Displaying the Glossary	353
5.4 Acronyms	354
5.5 Debugging Commands	369
6 Multi-Lingual Support	371
6.1 Polyglossia Captions	371
Glossary	373
Change History	374
Index	396

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2016/06/09 v4.25 (NLCT)]
```

Required packages:

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

The textcase package has much better case changing handling, so use `\MakeTextUppercase` instead of `\MakeUppercase`

```
6 \RequirePackage{textcase}
7 \renewcommand*{\mfirstucMakeUppercase}{\MakeTextUppercase}%
8 \RequirePackage{xfor}
```

```
9 \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.)

```
10 \RequirePackage{amsgen}
```

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

```
11 \RequirePackage{etoolbox}
```

Check if doc has been loaded.

```
f@gls@docloaded
```

```
12 \newif\if@gls@docloaded
13 \@ifpackageloaded{doc}%
14 {%
15   \@gls@docloadedtrue
16 }%
17 {%
18   \@ifclassloaded{nlctdoc}{\@gls@docloadedtrue}{\@gls@docloadedfalse}%
19 }
20 \if@gls@docloaded
```

\doc has been loaded, so some modifications need to be made to ensure both packages can work together. The amount of conflict has been reduced as from v4.11 and no longer involves patching internal commands.

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

org@theglossary

```
21 \let\glsorg@theglossary\theglossary
```

@endtheglossary

```
22 \let\glsorg@endtheglossary\endtheglossary
```

\PrintChanges

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

```
23 \let\glsorg@PrintChanges\PrintChanges
24 \renewcommand{\PrintChanges}{%
25   \begingroup
26     \let\theglossary\glsorg@theglossary
27     \let\endtheglossary\glsorg@endtheglossary
28     \glsorg@PrintChanges
29   \endgroup
30 }
```

End of doc stuff.

```
31 \fi
```

1.2 Package Options

debug

Switch on debug mode. This will also cancel the nowarn option.

```
32 \define@boolkey{glossaries.sty}[@gls@]{debug}[true]{%
33   \if@gls@debug
34     \renewcommand*\GlossariesWarning}[1]{%
35       \PackageWarning{glossaries}{##1}%
36     }%
37     \renewcommand*\GlossariesWarningNoLine}[1]{%
38       \PackageWarningNoLine{glossaries}{##1}%
39     }%
40     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
41   \else
42     \PackageInfo{glossaries}{debug mode OFF}%
43   \fi
44 }
```

Determine what to do if the see key is used before \makeglossaries. The default is to produce an error.

gls@see@noindex

```
45 \newcommand*\@gls@see@noindex{%
46   \PackageError{glossaries}{
```

```

47 {'see' key may only be used after \string\makeglossaries\space
48 or \string\makenoidxglossaries}%
49 {You must use \string\makeglossaries\space
50 or \string\makenoidxglossaries\space before defining
51 any entries that have a 'see' key}%
52 }

```

seenoinde

```

53 \define@choicekey{glossaries.sty}{seenoinde}[\val\nr]{error,warn,ignore}{%
54 \ifcase\nr
55 \renewcommand*{\@gls@see@noindex}{%
56 \PackageError{glossaries}%
57 {'see' key may only be used after \string\makeglossaries\space
58 or \string\makenoidxglossaries}%
59 {You must use \string\makeglossaries\space
60 or \string\makenoidxglossaries\space before defining
61 any entries that have a 'see' key}%
62 }%
63 \or
64 \renewcommand*{\@gls@see@noindex}{%
65 \GlossariesWarning{'see' key ignored}%
66 }%
67 \or
68 \renewcommand*{\@gls@see@noindex}{}%
69 \fi
70 }

```

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.

```
71 \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.

```
72 \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.

```

73 \ifcsundef{chapter}%
74 {\newcommand*{\@@glossarysec}{section}}%
75 {\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.

```

76 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
77 subsection,subsubsection,paragraph,subparagraph}[section]{%
78 \renewcommand*{\@@glossarysec}{#1}}

```

Determine whether or not to use numbered sections.

`glossarysecstar`

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

`glossaryseclabel`

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

`\glsautoprefix`

Prefix to add before label if automatically generated:

```
81 \newcommand*\glsautoprefix}{}
```

`numberedsection`

```
82 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
83 false,nolabel,autolabel,nameref}[nolabel]{%
84   \ifcase\nr\relax
85     \renewcommand*\@@glossarysecstar}{*}%
86     \renewcommand*\@@glossaryseclabel}{}%
87   \or
88     \renewcommand*\@@glossarysecstar}{}%
89     \renewcommand*\@@glossaryseclabel}{}%
90   \or
91     \renewcommand*\@@glossarysecstar}{}%
92     \renewcommand*\@@glossaryseclabel}{%
93       \label{\glsautoprefix\@glo@type}}%
94   \or
95     \renewcommand*\@@glossarysecstar}{*}%
96     \renewcommand*\@@glossaryseclabel}{%
97       \protected@edef\@currentlabelname{\glossarytoctitle}%
98       \label{\glsautoprefix\@glo@type}}%
99   \fi
100 }
```

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 [section 1.19](#).)

`y@default@style`

```
101 \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 [section 1.19](#).

```
102 \define@key{glossaries.sty}{style}{%
103   \renewcommand*\@glossary@default@style}{#1}%
104 }
```

Each `\DeclareOptionX` needs a corresponding `\DeclareOption` so that it can be passed as a document class option, so define a command that will implement both.

s@declareoption

```
105 \newcommand*{\@gls@declareoption}[2]{%
106   \DeclareOptionX{#1}{#2}%
107   \DeclareOption{#1}{#2}%
108 }
```

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”:

aryentrynumbers

```
109 \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).

```
110 \@gls@declareoption{nonumberlist}{%
111   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
112 }
```

savenumberlist

Provide means to store the number list for entries.

```
113 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{%
114   \glssavenumberlistfalse
```

eautionumberlist

```
115 \newcommand*\@glo@seeautionumberlist{}
```

eautionumberlist

Automatically activates number list for entries containing the see key.

```
116 \@gls@declareoption{seeautionumberlist}{%
117   \renewcommand*\@glo@seeautionumberlist}{%
118     \def\@glo@prefix{\glsnextpages}%
119   }%
120 }
```

\@gls@loadlong

```
121 \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.

```
122 \@gls@declareoption{nolong}{\renewcommand*\@gls@loadlong{}}
```

\@gls@loadsuper

The package isn't loaded if isn't installed.

```
123 \IfFileExists{supertabular.sty}{%
124   \newcommand*\@gls@loadsuper{\RequirePackage{glossary-super}}}{%
125   \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.

```
126 \@gls@declareoption{nosuper}{\renewcommand*{\@gls@loadsuper}{}}
```

`\@gls@loadlist`

```
127 \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.

```
128 \@gls@declareoption{nolist}{\renewcommand*{\@gls@loadlist}{}}
```

`\@gls@loadtree`

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

`notree` 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.

```
130 \@gls@declareoption{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).

```
131 \@gls@declareoption{nostyles}{%
132 \renewcommand*{\@gls@loadlong}{}%
133 \renewcommand*{\@gls@loadsuper}{}%
134 \renewcommand*{\@gls@loadlist}{}%
135 \renewcommand*{\@gls@loadtree}{}%
136 \let\@glossary@default@style\relax
137 }
```

`postdescription` 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 adjusted in case the description ends with an upper case letter. (Patch provided by Michael Pock.)

```
138 \newcommand*{\glspostdescription}{%
139 \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
140 }
```

`nopostdot` Boolean option to suppress post description dot

```
141 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
142 \glsnopostdotfalse
```

`nogroupskip` Boolean option to suppress vertical space between groups in the pre-defined styles.

```
143 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
144 \glsnogroupskipfalse
```

`ucmark` Boolean option to determine whether or not to use upper case in definition of `\gls glossarymark`

```
145 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
```

```

146 \@ifclassloaded{memoir}
147 {%
148   \glsucmarktrue
149 }%
150 {%
151   \glsucmarkfalse
152 }

```

`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`.

```

153 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
154 \glsentrycounterfalse

```

`entrycounterwithin` This option can be used to set a parent counter for `glossaryentry`. This option automatically sets `entrycounter=true`.

```

155 \define@key{glossaries.sty}{counterwithin}{%
156   \renewcommand*\@gls@counterwithin{#1}%
157   \glsentrycountertrue
158 }

```

`entrycounterwithin` The default value is no parent counter:

```

159 \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`.

```

160 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
161 \glsesubentrycounterfalse

```

`default@sorttype` Initialise default sort for `\printnoidxglossary`

```

162 \newcommand*\@glo@default@sorttype{standard}

```

`sort` Define the sort method: `sort=standard` (default), `sort=def` (order of definition) or `sort=use` (order of use).

```

163 \define@choicekey{glossaries.sty}{sort}{standard,def,use}{%
164   \renewcommand*\@glo@default@sorttype{#1}%
165   \csname @gls@setupsort@#1\endcsname
166 }

```

`prestandardsort`

```
\glsprestandardsort{<sort cs>}{<type>}{<label>}
```

Allow user to hook into sort mechanism. The first argument `<sort cs>` is the temporary control sequence containing the sort value before it has been sanitized and had `makeindex/xindy` special characters escaped.

```

167 \newcommand*\glsprestandardsort[3]{%
168   \glsdosanitizesort
169 }

```

`upsort@standard` Set up the macros for default sorting.

```

170 \newcommand*{\@gls@setupsort@standard}{%
  Store entry information when it's defined.
171 \def\do@glo@storeentry{\@glo@storeentry}%
  No count register required for standard sort.
172 \def\@gls@defsortcount##1{%
  Sort according to sort key (\@glo@sort) if provided otherwise sort according to the entry's
  name (\@glo@name). (First argument glossary type, second argument entry label.)
173 \def\@gls@defsort##1##2{%
174 \ifx\@glo@sort\@gls@defaultsort
175 \let\@glo@sort\@glo@name
176 \fi
177 \let\glsdosanitizesort\@gls@sanitizesort
178 \glsprestandardsort{\@glo@sort}{##1}{##2}%
179 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
180 }%
  Don't need to do anything when the entry is used.
181 \def\@gls@setsort##1{%
182 }
  Set standard sort as the default:
183 \@gls@setupsort@standard
  
```

`lssortnumberfmt` Format the number used as the sort key by `sort=def` and `sort=use`. Defaults to six digit numbering.

```

184 \newcommand*\glsortnumberfmt[1]{%
185 \ifnum#1<100000 0\fi
186 \ifnum#1<10000 0\fi
187 \ifnum#1<1000 0\fi
188 \ifnum#1<100 0\fi
189 \ifnum#1<10 0\fi
190 \number#1%
191 }
  
```

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

```

192 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
193 \def\do@glo@storeentry{\@glo@storeentry}%
  Defined count register associated with the glossary.
194 \def\@gls@defsortcount##1{%
195 \expandafter\global
196 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
197 }%
  
```

Increment count register associated with the glossary and use as the sort key.

```
198 \def\@gls@defsort##1##2{%
199   \expandafter\global\expandafter
200   \advance\csname glossary@##1@sortcount\endcsname by 1\relax
201   \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
202     \expandafter\glssortnumberfmt
203     {\csname glossary@##1@sortcount\endcsname}}%
204 }%
```

Don't need to do anything when the entry is used.

```
205 \def\@gls@setsort##1{%
206 }
```

s@setupsort@use Set up the macros for order of use sorting.

```
207 \newcommand*{\@gls@setupsort@use}{%
```

Don't store entry information when it's defined.

```
208 \let\do@glo@storeentry\@gobble
```

Defined count register associated with the glossary.

```
209 \def\@gls@defsortcount##1{%
210   \expandafter\global
211   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
212 }%
```

Initialise the sort key to empty.

```
213 \def\@gls@defsort##1##2{%
214   \expandafter\gdef\csname glo@##2@sort\endcsname{}%
215 }%
```

If the sort key hasn't been set, increment the counter associated with the glossary and set the sort key.

```
216 \def\@gls@setsort##1{%
```

Get the parent, if one exists

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

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

```
218   \ifx\@glo@parent\@empty
219   \else
220     \expandafter\@gls@setsort\expandafter{\@glo@parent}%
221   \fi
```

Set index information for this entry

```
222   \edef\@glo@type{\csname glo@##1@type\endcsname}%
223   \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
224   \ifx\@gls@tmp\@empty
225     \expandafter\global\expandafter
226     \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
227     \expandafter\protected@xdef\csname glo@##1@sort\endcsname{%
228       \expandafter\glssortnumberfmt
229       {\csname glossary@\@glo@type @sortcount\endcsname}}%
```

```

230     \glo@storeentry{##1}%
231     \fi
232 }%
233 }

```

`\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.)

```

234 \newcommand*\glsdefmain{%
235     \if@gls@docloaded
236     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
237     \else
238     \newglossary{main}{gls}{glo}{\glossaryname}%
239     \fi

```

Define hook to set the toc title when translator is in use.

```

240 \newcommand*\gls@tr@set@main@toctitle{%
241     \translatelet{\glossarytoctitle}{Glossary}%
242 }%
243 }

```

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 [section 1.10](#)).

`\glsdefaulttype`

```

244 \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`

```

245 \newcommand*\acronymtype{\glsdefaulttype}

```

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

```

246 \@gls@declareoption{nomain}{%
247     \let\glsdefaulttype\relax
248     \renewcommand*\glsdefmain}{}%
249 }

```

`acronym` The `acronym` option sets an associated conditional which is used in [section 1.17](#) to determine whether or not to define a separate glossary for acronyms.

```

250 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
251     \ifglsacronym

```

```

252 \renewcommand{\@gls@do@acronymsdef}{%
253 \DeclareAcronymList{acronym}%
254 \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
255 \renewcommand*{\acronymtype}{acronym}%

```

Define hook to set the toc title when translator is in use.

```

256 \newcommand*{\gls@tr@set@acronym@toctitle}{%
257 \translatelet{\glossarytoctitle}{Acronyms}%
258 }%
259 }%
260 \else
261 \let\@gls@do@acronymsdef\relax
262 \fi
263 }

```

`\printacronyms` Define `\printacronyms` at the start of the document if `acronym` is set and compatibility mode isn't on and `\printacronyms` hasn't already been defined.

```

264 \AtBeginDocument{%
265 \ifglsacronym
266 \ifbool{glscompatible-3.07}%
267 {}%
268 {%
269 \providecommand*{\printacronyms}[1] []{%
270 \printglossary[type=\acronymtype,#1]}%
271 }%
272 \fi
273 }

```

`@do@acronymsdef` Set default value

```

274 \newcommand*{\@gls@do@acronymsdef}{}

```

`acronyms` Provide a synonym for `acronym=true` that can be passed via the document class options.

```

275 \@gls@declareoption{acronyms}{%
276 \glsacronymtrue
277 \renewcommand{\@gls@do@acronymsdef}{%
278 \DeclareAcronymList{acronym}%
279 \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
280 \renewcommand*{\acronymtype}{acronym}%

```

Define hook to set the toc title when translator is in use.

```

281 \newcommand*{\gls@tr@set@acronym@toctitle}{%
282 \translatelet{\glossarytoctitle}{Acronyms}%
283 }%
284 }%
285 }

```

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

```

286 \newcommand*{\@glsacronymlists}{}

```

dtoacronymlists

```
287 \newcommand*{\@addtoacronymlists}[1]{%
288   \ifx\@glsacronymlists\empty
289   \protected@xdef\@glsacronymlists{#1}%
290   \else
291   \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
292   \fi
293 }
```

lareAcronymList 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.)

```
294 \newcommand*{\DeclareAcronymList}[1]{%
295   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
296 }
```

IfListOfAcronyms

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

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

```
297 \newcommand{\glsIfListOfAcronyms}[1]{%
298   \edef\@do@gls@islistofacronyms{%
299     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
300   \@do@gls@islistofacronyms
301 }
```

Internal command requires label and list to be expanded:

```
302 \newcommand{\@gls@islistofacronyms}[4]{%
303   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
304     \def\@before{##1}\def\@after{##2}}%
305   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
306   \ifx\@after\@nnil
```

Not found

```
307   #4%
308   \else
```

Found

```
309   #3%
310   \fi
311 }
```

lsglsacronymlist Convenient boolean.

```
312 \newif\if@lsglsacronymlist
```

cklsglsacronymlist Sets the above boolean if argument is a label representing a list of acronyms.

```
313 \newcommand*{\gls@cklsglsacronymlist}[1]{%
314   \glsIfListOfAcronyms{#1}%
315   {\@lsglsacronymlisttrue}{\@lsglsacronymlistfalse}%
316 }
```

`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.)

```
317 \newcommand*\SetAcronymLists[1]{%
318   \renewcommand*\@glsacronymlists{#1}%
319 }
```

`acronymlists`

```
320 \define@key{glossaries.sty}{acronymlists}{%
321   \DeclareAcronymList{#1}%
322 }
```

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 [section 1.6](#)).

`\glscounter`

```
323 \newcommand{\glscounter}{page}
```

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

```
324 \define@key{glossaries.sty}{counter}{%
325   \renewcommand*\glscounter{#1}%
326 }
```

`gls@nohyperlist`

```
327 \newcommand*\@gls@nohyperlist{}
```

`lareNoHyperList`

```
328 \newcommand*\GlsDeclareNoHyperList[1]{%
329   \ifdefempty\@gls@nohyperlist
330     {%
331       \renewcommand*\@gls@nohyperlist{#1}%
332     }%
333     {%
334       \appto\@gls@nohyperlist{,#1}%
335     }%
336 }
```

`nohypertypes`

```
337 \define@key{glossaries.sty}{nohypertypes}{%
338   \GlsDeclareNoHyperList{#1}%
339 }
```

`glossariesWarning` Prints a warning message.

```
340 \newcommand*\GlossariesWarning[1]{%
341   \PackageWarning{glossaries}{#1}%
342 }
```


esWarningNoLine Prints a warning message without the line number.

```
343 \newcommand*\GlossariesWarningNoLine}[1]{%
344 \PackageWarningNoLine{glossaries}{#1}%
345 }
```

nowarn Define package option to suppress warnings

```
346 \@gls@declareoption{nowarn}{%
347 \ifgls@debug
348 \GlossariesWarning{Warnings can't be suppressed in debug mode}%
349 \else
350 \renewcommand*\GlossariesWarning}[1]{}%
351 \renewcommand*\GlossariesWarningNoLine}[1]{}%
352 \fi
353 }
```

nonglossdefined Issue a warning if overriding \printglossary

```
354 \newcommand*\@gls@warnonglossdefined}{%
355 \GlossariesWarning{Overriding \string\printglossary}%
356 }
```

theglossdefined Issue a warning if overriding theglossary

```
357 \newcommand*\@gls@warnontheglossdefined}{%
358 \GlossariesWarning{Overriding 'theglossary' environment}%
359 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```
360 \@gls@declareoption{noredefwarn}{%
361 \renewcommand*\@gls@warnonglossdefined}{}%
362 \renewcommand*\@gls@warnontheglossdefined}{}%
363 }
```

As from version 3.08a, the only information written to the external glossary files are the label and sort values. Therefore, now, the only sanitize option that makes sense is the one for the sort key. so the sanitize option is now deprecated and there is only a sanitizesort option.

ls@sanitizedesc

```
364 \newcommand*\@gls@sanitizedesc}{%
365 }
```

lssetexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```
366 \newcommand*\glssetexpandfield}[1]{%
367 \csdef{gls@assign@#1@field}##1##2{%
368 \@gls@expand@field{##1}{#1}{##2}%
369 }%
370 }
```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```
371 \newcommand*{\glssetnoexpandfield}[1]{%
372   \csdef{gls@assign@#1@field}##1##2{%
373     \@gls@noexpand@field{##1}{#1}{##2}%
374   }%
375 }
```

sign@type@field The type must always be expandable.

```
376 \glssetexpandfield{type}
```

sign@desc@field The description is not expanded by default:

```
377 \glssetnoexpandfield{desc}
```

escplural@field

```
378 \glssetnoexpandfield{descplural}
```

ls@sanitizename

```
379 \newcommand*{\@gls@sanitizename}{}
```

sign@name@field Don't expand name by default.

```
380 \glssetnoexpandfield{name}
```

@sanitizesymbol

```
381 \newcommand*{\@gls@sanitizesymbol}{}
```

gn@symbol@field Don't expand symbol by default.

```
382 \glssetnoexpandfield{symbol}
```

bolplural@field

```
383 \glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```
384 \newcommand*{\@gls@sanitizesort}{%
385   \ifglssanitizesort
386     \@gls@sanitizesort
387   \else
388     \@gls@nosanitizesort
389   \fi
390 }
```

ls@sanitizesort

```
391 \newcommand*\@gls@sanitizesort{%
392   \@onelevel@sanitize\@glo@sort
393 }
```

@nosanitizesort

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

dx@sanitizesort Remove braces around first character (if present) before sanitizing.

```
395 \newcommand*\@gls@noidx@sanitizesort{%
396   \ifdefvoid\@glo@sort
397   {}%
398   {%
399     \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
400   }%
401 }
402 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
403   \def\@glo@sort{#1#2}%
404   \@onelevel@sanitize\@glo@sort
405 }
```

@nosanitizesort

```
406 \newcommand*{\@@gls@noidx@nosanitizesort}{%
407   \ifdefvoid\@glo@sort
408   {}%
409   {%
410     \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
411   }%
412 }
413 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
414   \bgroup
415     \glsnoidxstripaccents
416     \protected@xdef\@glo@sort{#1#2}%
417   \egroup
418   \let\@glo@sort\@glo@sort
419 }
```

idxstripaccents

```
420 \newcommand*\glsnoidxstripaccents{%
421   \let\IeC\@firstofone
422   \let\' \@firstofone
423   \let\` \@firstofone
424   \let\~ \@firstofone
425   \let\" \@firstofone
426   \let\u \@firstofone
427   \let\t \@firstofone
428   \let\d \@firstofone
429   \let\r \@firstofone
430   \let\= \@firstofone
431   \let\.\@firstofone
432   \let\~ \@firstofone
433   \let\v \@firstofone
434   \let\H \@firstofone
435   \let\c \@firstofone
```

```

436 \let\b\@firstofone
437 \def\AE{AE}%
438 \def\ae{ae}%
439 \def\OE{OE}%
440 \def\oe{oe}%
441 \def\AA{AA}%
442 \def\aa{aa}%
443 \def\L{L}%
444 \def\l{l}%
445 \def\O{O}%
446 \def\o{o}%
447 \def\SS{SS}%
448 \def\ss{ss}%
449 \def\th{th}%
450 }

```

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.

```

451 \define@boolkey[glS]{sanitize}{description}[true]{%
452 \GlossariesWarning{sanitize={description} package option deprecated}%
453 \ifglS@sanitize@description
454 \glSsetnoexpandfield{desc}%
455 \glSsetnoexpandfield{descplural}%
456 \else
457 \glSsetexpandfield{desc}%
458 \glSsetexpandfield{descplural}%
459 \fi
460 }

461 \define@boolkey[glS]{sanitize}{name}[true]{%
462 \GlossariesWarning{sanitize={name} package option deprecated}%
463 \ifglS@sanitize@name
464 \glSsetnoexpandfield{name}%
465 \else
466 \glSsetexpandfield{name}%
467 \fi
468 }

469 \define@boolkey[glS]{sanitize}{symbol}[true]{%
470 \GlossariesWarning{sanitize={symbol} package option deprecated}%
471 \ifglS@sanitize@symbol
472 \glSsetnoexpandfield{symbol}%
473 \glSsetnoexpandfield{symbolplural}%
474 \else
475 \glSsetexpandfield{symbol}%
476 \glSsetexpandfield{symbolplural}%
477 \fi
478 }

```

`sanitizesort`

```

479 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
480   \ifglssanitizesort
481     \glsssetnoexpandfield{sortvalue}%
482     \renewcommand*{\@gls@noidx@setsanitizesort}{%
483       \glssanitizesorttrue
484       \glsssetnoexpandfield{sortvalue}%
485     }%
486   \else
487     \glsssetexpandfield{sortvalue}%
488     \renewcommand*{\@gls@noidx@setsanitizesort}{%
489       \glssanitizesortfalse
490       \glsssetexpandfield{sortvalue}%
491     }%
492   \fi
493 }

```

Default setting:

```

494 \glssanitizesorttrue
495 \glsssetnoexpandfield{sortvalue}%

```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```

496 \newcommand*{\@gls@noidx@setsanitizesort}{%
497   \glssanitizesortfalse
498   \glsssetexpandfield{sortvalue}%
499 }

```

```

500 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
501   \setbool{glssanitizesort}{#1}%
502   \ifglssanitizesort
503     \glsssetnoexpandfield{sortvalue}%
504   \else
505     \glsssetexpandfield{sortvalue}%
506   \fi
507   \GlossariesWarning{sanitize={sort} package option
508     deprecated. Use sanitizesort instead}%
509 }

```

sanitize

```

510 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
511   \ifthenelse{\equal{#1}{none}}{%
512     {%
513       \GlossariesWarning{sanitize package option deprecated}%
514       \glsssetexpandfield{name}%
515       \glsssetexpandfield{symbol}%
516       \glsssetexpandfield{symbolplural}%
517       \glsssetexpandfield{desc}%
518       \glsssetexpandfield{descplural}%
519     }%
520   }%
521   \setkeys[gls]{sanitize}{#1}%

```

```

522 }%
523 }

\ifglstranslate As from version 3.13a, the translator package option is a choice rather than boolean option
so now need to define conditional:
524 \newif\ifglstranslate

notranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator
525 \newcommand*\@gls@usetranslator{%
polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
polyglossia as well.
526 \@ifpackageloaded{polyglossia}%
527 {%
528   \let\glsifusetranslator\@secondoftwo
529 }%
530 {%
531   \@ifpackageloaded{babel}%
532   {%
533     \IfFileExists{translator.sty}%
534     {%
535       \RequirePackage{translator}%
536       \let\glsifusetranslator\@firstoftwo
537     }%
538   }%
539 }%
540 {}%
541 }%
542 }

dtranslatordict Checks if given translator dictionary has been loaded.
543 \newcommand{\glsifusedtranslatordict}[3]{%
544   \glsifusetranslator
545   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
546   {#3}%
547 }

notranslate Provide a synonym for translate=false that can be passed via the document class.
548 \@gls@declareoption{notranslate}{%
549   \glstranslatefalse
550   \let\@gls@usetranslator\relax
551   \let\glsifusetranslator\@secondoftwo
552 }

translate Define translate option. If false don't set up multi-lingual support.
553 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
554 {true,false,babel}[true]%

```

```

555 {%
556   \ifcase\nr\relax
557     \glstranslatetrue
558     \renewcommand*\@gls@usetranslator{%
559       \@ifpackageloaded{polyglossia}%
560       {%
561         \let\glsifusetranslator\@secondoftwo
562       }%
563       {%
564         \@ifpackageloaded{babel}%
565         {%
566           \IfFileExists{translator.sty}%
567           {%
568             \RequirePackage{translator}%
569             \let\glsifusetranslator\@firstoftwo
570           }%
571         }%
572       }%
573     }%
574   }%
575 }%
576 \or
577   \glstranslatefalse
578   \let\@gls@usetranslator\relax
579   \let\glsifusetranslator\@secondoftwo
580 \or
581   \glstranslatetrue
582   \let\@gls@usetranslator\relax
583   \let\glsifusetranslator\@secondoftwo
584 \fi
585 }

```

Set the default value:

```

586 \glstranslatefalse
587 \let\glsifusetranslator\@secondoftwo
588 \@ifpackageloaded{translator}%
589 {%
590   \glstranslatetrue
591   \let\glsifusetranslator\@firstoftwo
592 }%
593 {%
594   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
595   {
596     \@ifpackageloaded{\gls@thissty}%
597     {%
598       \glstranslatetrue
599       \@endfortrue
600     }%
601   }%

```

```
602 }
603 }
```

indexonlyfirst Set whether to only index on first use.

```
604 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
605 \glsindexonlyfirstfalse
```

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

```
606 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
607 \glshyperfirsttrue
```

gls@setacrstyle Keep track of whether an acronym style has been set (for the benefit of `\setupglossaries`):

```
608 \newcommand*{\@gls@setacrstyle}{}
```

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

```
609 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}
610 \ifbool{glsacrdescription}%
611 {}%
612 {%
613   \renewcommand*{\@gls@sanitizedesc}{}%
614 }%
615 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
616 }
```

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

```
617 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}
618 \renewcommand*{\@gls@sanitizesymbol}{}%
619 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
620 }
```

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

```
621 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}
622 \renewcommand*{\@gls@sanitizesymbol}{}%
623 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
624 }
```

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

```
625 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}
626 \renewcommand*{\@gls@sanitizesymbol}{}%
627 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
628 }
```

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

```
629 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{}
630 \renewcommand*{\@gls@sanitizesymbol}{}%
631 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
632 }
```


`shortcuts` Define acronym shortcuts.
633 `\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.
634 `\newcommand*{\glsorder}{word}`

`\@glsorder` The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.
635 `\newcommand*{\@glsorder}[1]{}`

`order`
636 `\define@choicekey{glossaries.sty}{order}{word,letter}{%`
637 `\def\glsorder{#1}}`

`\ifglsxindy` Provide boolean to determine whether `xindy` or `makeindex` will be used to sort the glossaries.
638 `\newif\ifglsxindy`
The default is `makeindex`:
639 `\glsxindyfalse`

`makeindex` Define package option to specify that `makeindex` will be used to sort the glossaries:
640 `\@gls@declareoption{makeindex}{\glsxindyfalse}`
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.
641 `\define@boolkey[gls]{xindy}{glsnumbers}[true]{}`
642 `\gls@xindy@glsnumberstrue`

`y@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.)
643 `\def\@xdy@main@language{\languagename}%`
Define key to set the language
644 `\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.
645 `\ifcsundef{inputencodingname}{%`
646 `\def\gls@codepage{}}{%`
647 `\def\gls@codepage{\inputencodingname}`
648 `}`
Define a key to set the code page.
649 `\define@key[gls]{xindy}{codepage}{\def\gls@codepage{#1}}`

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

```
650 \define@key{glossaries.sty}{xindy}[]{%
651   \glsxindytrue
652   \setkeys[glS]{xindy}{#1}%
653 }
```

`xindygloss` Provide a synonym for `xindy` that can be passed via the document class options.

```
654 \@glS@declareoption{xindygloss}{%
655   \glsxindytrue
656 }
```

`xindynoglsnumbers` Provide a synonym for `xindy=glSnumbers=false` that can be passed via the document class options.

```
657 \@glS@declareoption{xindynoglsnumbers}{%
658   \glsxindytrue
659   \glS{xindy@glSnumbersfalse}
660 }
```

`automake` If this setting is on, automatically run `makeindex/xindy` at the end of the document. Must be used with `\makeglossaries`. Default is false.

```
661 \define@boolkey{glossaries.sty}[glS]{automake}[true]{%
662   \ifglSautomake
663     \renewcommand*\@glS@doautomake{%
664       \PackageError{glossaries}{You must use
665       \string\makeglossaries\space with automake=true}
666       {%
667         Either remove the automake=true setting or
668         add \string\makeglossaries\space to your document preamble.%
669       }%
670     }%
671   \else
672     \renewcommand*\@glS@doautomake{}%
673   \fi
674 }
675 \glSautomakefalse
```

`@glS@doautomake`

```
676 \newcommand*\@glS@doautomake{}
677 \AtEndDocument{\@glS@doautomake}
```

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

```
678 \define@boolkey{glossaries.sty}[glS]{savewrites}[true]{%
679   \ifglSsavewrites
680     \renewcommand*\glSwritefiles{\@glSwritefiles}%
681   \else
682     \let\glSwritefiles\@empty
683   \fi
684 }
```

Set default:

```
685 \glssavewritesfalse
686 \let\glswritefiles\@empty
```

compatible-3.07

```
687 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{%
688 \boolfalse{glscompatible-3.07}}
```

compatible-2.07

```
689 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
```

Also set 3.07 compatibility if this option is set.

```
690 \ifbool{glscompatible-2.07}%
691 {%
692   \booltrue{glscompatible-3.07}%
693 }%
694 {}%
695 }
696 \boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

```
697 \@gls@declareoption{symbols}{%
698 \let\@gls@do@symbolsdef\@gls@symbolsdef
699 }
```

Default is not to define the symbols glossary:

```
700 \newcommand*{\@gls@do@symbolsdef}{}
```

@gls@symbolsdef

```
701 \newcommand*{\@gls@symbolsdef}{%
702 \newglossary[slg]{symbols}{sls}{slo}{\glsymbolsgroupname}%
703 \newcommand*{\printsymbols}[1][\printglossary[type=symbols,##1]]%
```

Define hook to set the toc title when translator is in use.

```
704 \newcommand*{\gls@tr@set@symbols@toctitle}{%
705 \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
706 }%
707 }%
```

numbers Create a “symbols” glossary type

```
708 \@gls@declareoption{numbers}{%
709 \let\@gls@do@numbersdef\@gls@numbersdef
710 }
```

Default is not to define the numbers glossary:

```
711 \newcommand*{\@gls@do@numbersdef}{}
```

@gls@numbersdef

```
712 \newcommand*{\@gls@numbersdef}{%
713 \newglossary[nlg]{numbers}{nls}{nlo}{\glsnumbersgroupname}%
714 \newcommand*{\printnumbers}[1][\printglossary[type=numbers,##1]]%
```

Define hook to set the toc title when translator is in use.

```
715 \newcommand*\gls@tr@set@numbers@toctitle}{%
716   \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
717 }%
718 }%
```

index Create an “index” glossary type

```
719 \@gls@declareoption{index}{%
720   \let\@gls@do@indexdef\@gls@indexdef
721 }
```

Default is not to define index glossary:

```
722 \newcommand*\@gls@do@indexdef{}
```

\@gls@indexdef \indexname isn't set by glossaries.

```
723 \newcommand*\@gls@indexdef{%
724   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
725   \newcommand*\printindex}[1] [] {\printglossary[type=index,##1]}%
726   \newcommand*\newterm}[2] [] {%
727     \newglossaryentry{##2}%
728     {type={index},name={##2},description={\nopostdesc},##1}
729 }%
```

Process package options. First process any options that have been passed via the document class.

```
730 \@for\CurrentOption :=\@declaredoptions\do{%
731   \ifx\CurrentOption\@empty
732   \else
733     \@expandtwoargs
734     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
735     \ifin@
736     \@use@ption
737     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
738   \fi
739 \fi
740 }
```

Now process options passed to the package:

```
741 \ProcessOptionsX
```

Load backward compatibility stuff:

```
742 \RequirePackage{glossaries-compatible-307}
```

setupglossaries Provide way to set options after package has been loaded. However, some options must be set before \ProcessOptionsX, so they have to be disabled:

```
743 \disable@keys{glossaries.sty}{compatible-2.07,%
744 xindy,xindygloss,xindynoglsnumbers,makeindex,%
745 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,nomain}
```

Now define `\setupglossaries`:

```

746 \newcommand*{\setupglossaries}[1]{%
747   \renewcommand*{\@gls@setacrstyle}{}%
748   \ifglsacrshortcuts
749     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
750   \else
751     \def\@gls@setupshortcuts{%
752       \ifglsacrshortcuts
753         \DefineAcronymSynonyms
754       \fi
755     }%
756   \fi
757   \glsacrshortcutsfalse
758   \let\@gls@do@numbersdef\relax
759   \let\@gls@do@symbolssdef\relax
760   \let\@gls@do@indexdef\relax
761   \let\@gls@do@acronymsdef\relax
762   \setkeys{glossaries.sty}{#1}%
763   \@gls@setacrstyle
764   \@gls@setupshortcuts
765   \@gls@do@acronymsdef
766   \@gls@do@numbersdef
767   \@gls@do@symbolssdef
768   \@gls@do@indexdef
769 }
```

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}`).

```

770 \ifthenelse{\equal{\glscounter}{section}}{%
771   {%
772     \ifcsundef{chapter}{}%
773     {%
774       \let\@gls@old@chapter\@chapter
775       \def\@chapter[#1]#2{\@gls@old@chapter[#1]}{#2}%
776       \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}}{}}%
777     }%
778   }%
779 }
```

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

```

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

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

```
781 \newcommand*{\@onlypremakeg}[1]{%
782   \ifx\@gls@onlypremakeg\@empty
783     \def\@gls@onlypremakeg{#1}%
784   \else
785     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
786     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
787   \fi
788 }
```

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

```
789 \newcommand*{\@disable@onlypremakeg}{%
790 \@for\@thiscs:=\@gls@onlypremakeg\do{%
791   \expandafter\@disable@premakecs\@thiscs%
792 }}
```

`\sable@premakecs` Disables the given command.

```
793 \newcommand*{\@disable@premakecs}[1]{%
794   \def#1{\PackageError{glossaries}{\string#1\space may only be
795     used before \string\makeglossaries}{You can't use
796     \string#1\space after \string\makeglossaries}}%
797 }
```

1.3 Predefined Text

Set up default textual tags 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`

```
798 \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`

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

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

```
800 \newcommand*{\glssettoctitle}[1]{%
801 \def\glossarytoctitle{\csname @gls@#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
802 \providecommand*\entryname{Notation}

descriptionname
803 \providecommand*\descriptionname{Description}

\symbolname
804 \providecommand*\symbolname{Symbol}

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

Labels for makeindex's symbol and number groups:

symbolsgroupname
806 \providecommand*\glssymbolsgroupname{Symbols}

numbersgroupname
807 \providecommand*\glsnumbersgroupname{Numbers}

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

acrpluralsuffix Default plural suffix for acronyms
809 \newcommand*\glsacrpluralsuffix{\glspluralsuffix}

acrpluralsuffix
810 \newcommand*\glsupacrpluralsuffix{\glstextup{\glsacrpluralsuffix}}

\seename
811 \providecommand*\seename{see}

\andname
812 \providecommand*\andname{\&}

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

eGlossariesLang
813 \newcommand*\RequireGlossariesLang[1]{%
814 \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
815 }

sGlossariesLang
816 \newcommand*\ProvidesGlossariesLang[1]{%
817 \ProvidesFile{glossaries-#1.ldf}%
818 }

```

ssarytocaptions Does nothing if translator hasn't been loaded.

```
819 \newcommand*{\addglossarytocaptions}[1]{}
```

As from v4.12, multilingual support has been split off into independently-maintained language modules.

```
820 \ifglstranslate
```

Load tracklang

```
821 \RequirePackage{tracklang}
```

Load translator if required.

```
822 \@gls@usetranslator
```

If using `,` `\glossaryname` should be defined in terms of `\translate`, but if `babel` is also loaded, it will redefine `\glossaryname` whenever the language is set, so override it. (Don't use `\addto` as doesn't define it.)

```
823 \@ifpackageloaded{translator}
```

```
824 {%
```

If the language options have been specified through the document class, then translator can pick them up. If not, translator will default to English and any language option passed to `babel` won't be detected, so if `\trans@languages` is just English and `\bbl@loaded` isn't simply `english`, then don't use the translator dictionaries.

```
825 \ifboolexpr
```

```
826 {
```

```
827 test {\ifdefstring{\trans@languages}{English}}
```

```
828 and not
```

```
829 test {\ifdefstring{bbl@loaded}{english}}
```

```
830 }
```

```
831 {%
```

```
832 \let\glsifusetranslator\@secondoftwo
```

```
833 }%
```

```
834 {%
```

```
835 \usedictionary{glossaries-dictionary}%
```

```
836 \renewcommand*{\addglossarytocaptions}[1]{%
```

```
837 \ifcsundef{captions#1}{}%
```

```
838 {%
```

```
839 \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
```

```
840 \expandafter\toks@\expandafter{\@gls@tmp
```

```
841 \renewcommand*{\glossaryname}{\translate{Glossary}}}%
```

```
842 }%
```

```
843 \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
```

```
844 }%
```

```
845 }%
```

```
846 }%
```

```
847 }%
```

```
848 }%
```

Check for tracked languages

```
849 \AnyTrackedLanguages
```



```

850  {%
851    \ForEachTrackedDialect{\this@dialect}{%
852      \IfTrackedLanguageFileExists{\this@dialect}%
853      {glossaries-}% prefix
854      {.ldf}%
855      {%
856        \RequireGlossariesLang{\CurrentTrackedTag}%
857      }%
858      {%
859        \PackageWarningNoLine{glossaries}%
860        {No language module detected for ‘\this@dialect’.\MessageBreak
861        Language modules need to be installed separately.\MessageBreak
862        Please check on CTAN for a bundle called\MessageBreak
863        ‘glossaries-\CurrentTrackedLanguage’ or similar}%
864      }%
865    }%
866  }%
867  {}%

```

if using translator use translator interface.

```

868  \glsifusetranslator
869  {%
870    \renewcommand*\glssettoctitle}[1]{%
871      \ifcsdef{gls@tr@set@#1@toctitle}%
872      {%
873        \csuse{gls@tr@set@#1@toctitle}%
874      }%
875      {%
876        \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
877      }%
878    }%
879    \renewcommand*\glossaryname{\translate{Glossary}}%
880    \renewcommand*\acronymname{\translate{Acronyms}}%
881    \renewcommand*\entryname{\translate{Notation (glossaries)}}%
882    \renewcommand*\descriptionname{%
883      \translate{Description (glossaries)}}%
884    \renewcommand*\symbolname{\translate{Symbol (glossaries)}}%
885    \renewcommand*\pagelistname{%
886      \translate{Page List (glossaries)}}%
887    \renewcommand*\glssymbolsgroupname{%
888      \translate{Symbols (glossaries)}}%
889    \renewcommand*\glsnumbersgroupname{%
890      \translate{Numbers (glossaries)}}%
891  }{}%
892 \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.

```
893 \DeclareRobustCommand*\nopostdesc{}
```

`\@nopostdesc` Suppress next description terminator.

```

894 \newcommand*\@nopostdesc}{%
895   \let\org@glspostdescription\glspostdescription
896   \def\glspostdescription{%
897     \let\glspostdescription\org@glspostdescription}%
898 }
```

`\@no@post@desc` Used for comparison purposes.

```

899 \newcommand*\@no@post@desc}{\nopostdesc}
```

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

```

900 \newcommand{\glspar}{\par}
```

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

```

901 \newcommand{\setStyleFile}[1]{%
902   \renewcommand*\gl@istfilebase}{#1}%
   Just in case \istfilename has been modified.
903   \ifglsxindy
904     \def\istfilename{\gl@istfilebase.xdy}
905   \else
906     \def\istfilename{\gl@istfilebase.ist}
907   \fi
908 }
```

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

```

909 \@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`

```

910 \ifglsxindy
911   \def\istfilename{\gl@istfilebase.xdy}
912 \else
913   \def\istfilename{\gl@istfilebase.ist}
914 \fi
```

`gl@istfilebase`

```

915 \newcommand*\gl@istfilebase}{\jobname}
```

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 \LaTeX , `\@istfilename` ignores its argument.

`\@istfilename`

```

916 \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`

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

`\glsSetCompositor` Sets the compositor.

```
918 \newcommand*\glsSetCompositor}[1]{%
919 \renewcommand*\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
920 \@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 `<letter><compositor><number>`. 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.

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

`\glsAlphaCompositor` Sets the alpha compositor.

```
922 \ifglsxindy
923 \newcommand*\glsSetAlphaCompositor[1]{%
924 \renewcommand*\@glsAlphacompositor}{#1}}
925 \else
926 \newcommand*\glsSetAlphaCompositor[1]{%
927 \glsnoxindywarning\glsSetAlphaCompositor}
928 \fi
```

Can only be used before `\makeglossaries`

```
929 \@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.

```
930 \newcommand*\gls@suffixF}{}
```

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

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

Only has an effect when used before `\makeglossaries`

```
933 \@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.

```
934 \newcommand*{\gls@suffixFF}{}
```

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

```
935 \newcommand*{\glsSetSuffixFF}[1]{%
936   \renewcommand*{\gls@suffixFF}{#1}%
937 }
```

`\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 `\glsnumberformat`, otherwise it will simply display its argument "as is".

```
938 \ifcsundef{hyperlink}%
939 {%
940   \newcommand*{\glsnumberformat}[1]{#1}%
941 }%
942 {%
943   \newcommand*{\glsnumberformat}[1]{\glsnumberformat{#1}}%
944 }
```

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`

```
945 \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`

```
946 \newcommand{\delimR}{--}
```

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

`\glossarypreamble`

```
947 \newcommand*{\glossarypreamble}{%
948   \csuse{@glossarypreamble@currentglossary}%
949 }
```

glossary preamble

```
\setglossarypreamble[<type>]{<text>}
```

Code provided by Michael Pock.

```
950 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
951   \ifglossaryexists{#1}{%
952     \csgdef{@glossarypreamble@#1}{#2}%
953   }{%
954     \GlossariesWarning{%
955       Glossary ‘#1’ is not defined%
956     }%
957   }%
958 }
```

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

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

glossary postamble

```
959 \newcommand*{\glossarypostamble}{}
```

glossary section

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

```
960 \newcommand*{\glossarysection}[2][\@gls@title]{%
961   \def\@gls@title{#2}%
962   \ifcsundef{phantomsection}%
963     {%
964       \@glossarysection{#1}{#2}%
965     }%
966     {%
967       \p@glossarysection{#1}{#2}%
968     }%

969   \gls@glossarymark{\glossarytoctitle}%
970 }
```

gls glossary mark

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

```
971 \ifcsundef{glossarymark}%
972   {%
973     \newcommand{\gls@glossarymark}[1]{\glossarymark{#1}}
974   }%
975   {}%
```

```

976 \@ifclassloaded{memoir}
977 {%
978   \newcommand{\glsglossarymark}[1]{%
979     \ifglsucmark
980       \markboth{\memUHead{#1}}{\memUHead{#1}}%
981     \else
982       \markboth{#1}{#1}%
983     \fi
984   }
985 }%
986 {%
987   \newcommand{\glsglossarymark}[1]{%
988     \ifglsucmark
989       \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
990     \else
991       \@mkboth{#1}{#1}%
992     \fi
993   }
994 }
995 }

```

`\glossarymark` Provided for backward compatibility:

```

996 \providecommand{\glossarymark}[1]{%
997   \ifglsucmark
998     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
999   \else
1000     \@mkboth{#1}{#1}%
1001   \fi
1002 }

```

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.

`glossarysection`

```

1003 \newcommand*{\setglossarysection}[1]{%
1004 \setkeys{glossaries.sty}{section=#1}}

```

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

`glossarysection`

```

1005 \newcommand*{\@glossarysection}[2]{%
1006   \ifdefempty\@glossarysecstar
1007   {%
1008     \csname\@glossarysec\endcsname[#1]{#2}%
1009   }%
1010   {%

```

```

1011 \csname\@glossarysec\endcsname*{#2}%
1012 \gls@toc{#1}{\@glossarysec}%
1013 }%

```

Do automatic labelling if required

```

1014 \@glossaryseclabel
1015 }

```

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.

`glossarysection`

```

1016 \newcommand*\@pglossarysection [2]{%
1017 \glsclearpage
1018 \phantomsection
1019 \ifdefempty\@glossarysecstar
1020 {%
1021 \csname\@glossarysec\endcsname{#2}%
1022 }%
1023 {%
1024 \gls@toc{#1}{\@glossarysec}%
1025 \csname\@glossarysec\endcsname*{#2}%
1026 }%

```

Do automatic labelling if required

```

1027 \@glossaryseclabel
1028 }

```

`gls@docclearpage` The `\gls@docclearpage` 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.

```

1029 \newcommand*\gls@docclearpage{%
1030 \ifthenelse{\equal{\@glossarysec}{chapter}}%
1031 {%
1032 \ifcsundef{cleardoublepage}%
1033 {%
1034 \clearpage
1035 }%
1036 }%
1037 \ifcsdef{if@openright}%
1038 {%
1039 \if@openright
1040 \cleardoublepage
1041 \else
1042 \clearpage
1043 \fi
1044 }%
1045 }%
1046 \cleardoublepage

```

```

1047     }%
1048   }%
1049 }%
1050 {}%
1051 }

```

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

```
1052 \newcommand*\glsclearpage{\gls@doclearpage}
```

The glossary is added to the table of contents if `glostoc` 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`

```

1053 \newcommand*\@gls@toc[2]{%
1054   \ifglstoc
1055     \ifglsnumberline
1056       \addcontentsline{toc}{#2}{\protect\numberline{}#1}%
1057     \else
1058       \addcontentsline{toc}{#2}{#1}%
1059     \fi
1060   \fi
1061 }

```

1.4 Xindy

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

`\glsnoxywarning` Issues a warning if `xindy` hasn't been specified. These warnings can be suppressed by re-defining `\glsnoxywarning` to ignore its argument

```

1062 \newcommand*\glsnoxywarning[1]{%
1063   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1064 }

```

`\glsnoindexwarning` Reverse for commands that may only be used with `makeindex`.

```

1065 \newcommand*\glsnoindexwarning[1]{%
1066   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1067 }

```

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

```

1068 \ifglsxindy
1069   \edef\@xdyattributes{\string"default\string"}%
1070 \fi

```


dyattributelist Comma-separated list of attributes.

```
1071 \ifglxindy
1072 \edef\xdyattributelist{}%
1073 \fi
```

\@xdylocref Define list of markup location references.

```
1074 \ifglxindy
1075 \def\xdylocref{}
1076 \fi
```

\@gls@ifinlist

```
1077 \newcommand*\@gls@ifinlist[4]{%
1078 \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
1079 \def@gls@listsuffix{##2}%
1080 \ifx@gls@listsuffix\@empty
1081 #4%
1082 \else
1083 #3%
1084 \fi
1085 }%
1086 \@do@ifinlist,#2,#1,\end@do@ifinlist
1087 }
```

sAddXdyCounters 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.

```
1088 \ifglxindy
1089 \newcommand*\@xdycounters{\glscounter}
1090 \newcommand*\GlsAddXdyCounters[1]{%
1091 \@for@gls@ctr:=#1\do{%
    Check if already in list before adding.
1092 \edef\@do@addcounter{%
1093 \noexpand@gls@ifinlist{\gls@ctr}{\@xdycounters}{}%
1094 {%
1095 \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1096 \noexpand@gls@ctr}%
1097 }%
1098 }%
1099 \@do@addcounter
1100 }
1101 }
```

Only has an effect before \writeist:

```
1102 \@onlypremakeg\GlsAddXdyCounters
1103 \else
1104 \newcommand*\GlsAddXdyCounters[1]{%
1105 \glsnoxindywarning\GlsAddXdyAttribute
1106 }
1107 \fi
```

saddxdycounters Counters must all be identified before adding attributes.

```
1108 \newcommand*\@disabled@glssaddxdycounters{%
1109   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1110   can't be used after \string\GlsAddXdyAttribute}{Move all
1111   occurrences of \string\GlsAddXdyCounters\space before the first
1112   instance of \string\GlsAddXdyAttribute}%
1113 }
```

AddXdyAttribute Adds an attribute.

```
1114 \ifglxsindy
```

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

```
1115 \newcommand*\@glssaddxdyattribute[2]{%
```

Add to xindy attribute list

```
1116   \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1117   \string"#2#1\string"}%
```

Add to xindy markup location.

```
1118   \expandafter\toks@\expandafter{\@xdylocref}%
1119   \edef\@xdylocref{\the\toks@ ^^J%
1120   (markup-locref
1121   :open \string"\glstildechar n%
1122   \expandafter\string\csname glsX#2X#1\endcsname
1123   \string" ^^J
1124   :close \string"\string" ^^J
1125   :attr \string"#2#1\string")}%
```

Define associated attribute command $\text{glsX}\langle counter \rangle X\langle attribute \rangle \{ \langle Hprefix \rangle \} \{ \langle n \rangle \}$

```
1126   \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1127     \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1128   }%
1129 }
```

High-level command:

```
1130 \newcommand*\GlsAddXdyAttribute[1]{%
```

Add to comma-separated attribute list

```
1131   \ifx\@xdyattributelist\@empty
1132     \edef\@xdyattributelist{#1}%
1133   \else
1134     \edef\@xdyattributelist{\@xdyattributelist,#1}%
1135   \fi
```

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

```
1136   \@for\@this@counter:=\@xdycounters\do{%
1137     \protected@edef\gls@do@addxdyattribute{%
1138       \noexpand\@glssaddxdyattribute{#1}{\@this@counter}%
1139     }
1140     \gls@do@addxdyattribute
1141   }%
```

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

```
1142 \let\GlsAddXdyCounters\@disabled@glssaddxdycounters
1143 }
```

Only has an effect before `\writeist`:

```
1144 \@onlypremakeg\GlsAddXdyAttribute
1145 \else
1146 \newcommand*\GlsAddXdyAttribute[1]{%
1147 \glsnoxindywarning\GlsAddXdyAttribute}
1148 \fi
```

`definedattributes` Add known attributes for all defined counters

```
1149 \ifglsxindy
1150 \newcommand*\@gls@addpredefinedattributes{%
1151 \GlsAddXdyAttribute{glsnumberformat}
1152 \GlsAddXdyAttribute{textrm}
1153 \GlsAddXdyAttribute{textsf}
1154 \GlsAddXdyAttribute{texttt}
1155 \GlsAddXdyAttribute{textbf}
1156 \GlsAddXdyAttribute{textmd}
1157 \GlsAddXdyAttribute{textit}
1158 \GlsAddXdyAttribute{textup}
1159 \GlsAddXdyAttribute{textsl}
1160 \GlsAddXdyAttribute{textsc}
1161 \GlsAddXdyAttribute{emph}
1162 \GlsAddXdyAttribute{glshypernumber}
1163 \GlsAddXdyAttribute{hyperrm}
1164 \GlsAddXdyAttribute{hypersf}
1165 \GlsAddXdyAttribute{hypertt}
1166 \GlsAddXdyAttribute{hyperbf}
1167 \GlsAddXdyAttribute{hypermd}
1168 \GlsAddXdyAttribute{hyperit}
1169 \GlsAddXdyAttribute{hyperup}
1170 \GlsAddXdyAttribute{hypersl}
1171 \GlsAddXdyAttribute{hypersc}
1172 \GlsAddXdyAttribute{hyperemph}

1173 \GlsAddXdyAttribute{glsignore}
1174 }
1175 \else
1176 \let\@gls@addpredefinedattributes\relax
1177 \fi
```

`dyuseralphabets` List of additional alphabets

```
1178 \def\@xdyuseralphabets{}
```

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

```
1179 \ifglsxindy
```

```

1180 \newcommand*\GlsAddXdyAlphabet}[2]{%
1181 \edef\xdyuseralphabets{%
1182   \xdyuseralphabets ^^J
1183   (define-alphabet "#1" (#2))}}
1184 \else
1185 \newcommand*\GlsAddXdyAlphabet}[2]{%
1186   \glsnoxindywarning\GlsAddXdyAlphabet}
1187 \fi

```

This code is only required for xindy:

```
1188 \ifglsxindy
```

`\gls@locationlist` List of predefined location names.

```

1189 \newcommand*\@gls@xdy@locationlist}{%
1190   roman-page-numbers,%
1191   Roman-page-numbers,%
1192   arabic-page-numbers,%
1193   alpha-page-numbers,%
1194   Alpha-page-numbers,%
1195   Appendix-page-numbers,%
1196   arabic-section-numbers%
1197 }

```

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

`\gls@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.

```

1198 \protected@edef\@gls@roman{\@roman{0}\string"
1199   \string"roman-numbers-lowercase\string" :sep \string"}%
1200 \@onelevel@sanitize\@gls@roman
1201 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1202   :sep \string"}%
1203 \@onelevel@sanitize\@tmp
1204 \ifx\@tmp\@gls@roman
1205   \expandafter
1206     \edef\cename @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1207       \string"roman-numbers-lowercase\string"%
1208     }%
1209 \else
1210   \expandafter
1211     \edef\cename @gls@xdy@Lclass@roman-page-numbers\endcsname{
1212       :sep \string"\@gls@roman\string"%
1213     }%
1214 \fi

```

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

```

1215 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1216   \string"roman-numbers-uppercase\string"%
1217 }%

```

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

```

1218 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1219   \string"arabic-numbers\string"%
1220 }%

```

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

```

1221 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1222   \string"alpha\string"%
1223 }%

```

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

```

1224 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1225   \string"ALPHA\string"%
1226 }%

```

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

```

1227 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1228   \string"ALPHA\string"
1229   :sep \string"\@glsAlphacompositor\string"
1230   \string"arabic-numbers\string"%
1231 }

```

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

```

1232 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1233   \string"arabic-numbers\string"
1234   :sep \string"\glscompositor\string"
1235   \string"arabic-numbers\string"%
1236 }%

```

erlocationdefs List of additional location definitions (separated by ^^J)

```

1237 \def\@xdyuserlocationdefs{}

```

erlocationnames List of additional user location names

```

1238 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

1239 \fi

```

sAddXdyLocation `\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.)

```

1240 \ifglsxindy

```

```

1241 \newcommand*{\GlsAddXdyLocation}[3][[]]{%
1242   \def\@gls@tmp{#1}%
1243   \ifx\@gls@tmp\@empty
1244     \edef\@xdyuserlocationdefs{%
1245       \@xdyuserlocationdefs ^^J%
1246       (define-location-class \string"#2\string"^^J\space\space
1247       \space(:sep \string"{} \glsopenbrace\string" #3
1248         :sep \string"\glsclosebrace\string"))
1249     }%
1250   \else
1251     \edef\@xdyuserlocationdefs{%
1252       \@xdyuserlocationdefs ^^J%
1253       (define-location-class \string"#2\string"^^J\space\space
1254       \space(:sep "\glsopenbrace"
1255         #1
1256         :sep "\glsclosebrace\glsopenbrace" #3
1257         :sep "\glsclosebrace"))
1258     }%
1259   \fi
1260   \edef\@xdyuserlocationnames{%
1261     \@xdyuserlocationnames^^J\space\space\space
1262     \string"#1\string"}%
1263 }

```

Only has an effect before `\writeist`:

```

1264 \@onlypremakeg\GlsAddXdyLocation
1265 \else
1266 \newcommand*{\GlsAddXdyLocation}[2]{%
1267   \glsnoxindywarning\GlsAddXdyLocation}
1268 \fi

```

`ationclassorder` Define location class order

```

1269 \ifglxindy
1270 \edef\@xdylocationclassorder{^^J\space\space\space
1271   \string"roman-page-numbers\string"^^J\space\space\space
1272   \string"arabic-page-numbers\string"^^J\space\space\space
1273   \string"arabic-section-numbers\string"^^J\space\space\space
1274   \string"alpha-page-numbers\string"^^J\space\space\space
1275   \string"Roman-page-numbers\string"^^J\space\space\space
1276   \string"Alpha-page-numbers\string"^^J\space\space\space
1277   \string"Appendix-page-numbers\string"
1278   \@xdyuserlocationnames^^J\space\space\space
1279   \string"see\string"
1280 }
1281 \fi

```

Change the location order.

`ationClassOrder`

```

1282 \ifglxindy

```

```

1283 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1284   \def\@xdylocationclassorder{#1}}
1285 \else
1286 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1287   \glsnoxindywarning\GlsSetXdyLocationClassOrder}
1288 \fi

```

`\@xdysortrules` Define sort rules

```

1289 \ifglxindy
1290 \def\@xdysortrules{}
1291 \fi

```

`\GlsAddSortRule` Add a sort rule

```

1292 \ifglxindy
1293 \newcommand*\GlsAddSortRule[2]{%
1294   \expandafter\toks@\expandafter{\@xdysortrules}%
1295   \protected@edef\@xdysortrules{\the\toks@ ^^J
1296     (sort-rule \string"#1\string" \string"#2\string")}%
1297   }
1298 \else
1299 \newcommand*\GlsAddSortRule[2]{%
1300   \glsnoxindywarning\GlsAddSortRule}
1301 \fi

```

`\@xdyrequiredstyles` Define list of required styles (this should be a comma-separated list of xindy styles)

```

1302 \ifglxindy
1303 \def\@xdyrequiredstyles{tex}
1304 \fi

```

`\GlsAddXdyStyle` Add a xindy style to the list of required styles

```

1305 \ifglxindy
1306 \newcommand*\GlsAddXdyStyle[1]{%
1307   \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1308 \else
1309 \newcommand*\GlsAddXdyStyle[1]{%
1310   \glsnoxindywarning\GlsAddXdyStyle}
1311 \fi

```

`\GlsSetXdyStyles` Reset the list of required styles

```

1312 \ifglxindy
1313 \newcommand*\GlsSetXdyStyles[1]{%
1314   \edef\@xdyrequiredstyles{#1}}
1315 \else
1316 \newcommand*\GlsSetXdyStyles[1]{%
1317   \glsnoxindywarning\GlsSetXdyStyles}
1318 \fi

```

`\indrootlanguage` 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` is no longer available. Now provide a command that does nothing (in case it's been patched), but this may be removed completely in the future.

```
1319 \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.

```
1320 \def\xdylanguage#1#2{}
```

`sSetXdyLanguage` 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.

```
1321 \ifglxindy
1322 \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1323 \ifglossaryexists{#1}{%
1324 \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1325 }{%
1326 \PackageError{glossaries}{Can't set language type for
1327 glossary type '#1' --- no such glossary}{%
1328 You have specified a glossary type that doesn't exist}}
1329 \else
1330 \newcommand*\GlsSetXdyLanguage[2][]{%
1331 \glsnoxindywarning\GlsSetXdyLanguage}
1332 \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.

```
1333 \def\gls@codepage#1#2{}
```

`sSetXdyCodePage` Define command to set the code page.

```
1334 \ifglxindy
1335 \newcommand*\GlsSetXdyCodePage[1]{%
1336 \renewcommand*\gls@codepage{#1}%
1337 }
```

Suggested by egreg:

```
1338 \AtBeginDocument{%
1339 \ifx\gls@codepage\@empty
1340 \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{ }%
1341 \fi
1342 }
1343 \else
1344 \newcommand*\GlsSetXdyCodePage[1]{%
1345 \glsnoxindywarning\GlsSetXdyCodePage}
1346 \fi
```

`xdylettergroups` Store letter group definitions.

```
1347 \ifglxindy
```



```

1348 \ifgls@xindy@glsnumbers
1349   \def\@xdylettergroups{(define-letter-group
1350     \string"glsnumbers\string"^^J\space\space\space
1351     :prefixes (\string"0\string" \string"1\string"
1352     \string"2\string" \string"3\string" \string"4\string"
1353     \string"5\string" \string"6\string" \string"7\string"
1354     \string"8\string" \string"9\string")^^J\space\space\space
1355     :before \string"\@glsfirstletter\string")}
1356 \else
1357   \def\@xdylettergroups{}
1358 \fi
1359 \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.

```

1360 \newcommand*\GlsAddLetterGroup[2]{%
1361   \expandafter\toks@\expandafter{\@xdylettergroups}%
1362   \protected@edef\@xdylettergroups{\the\toks@^^J%
1363   (define-letter-group \string"#1\string"^^J\space\space#2)}%
1364 }%

```

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.

```

1365 \newcommand*\forallglossaries}[3][\@glo@types]{%
1366   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1367 }

```

`\forallacronyms`

```

1368 \newcommand*\forallacronyms}[2]{%
1369   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1370 }

```

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

```
\forallglsentries[<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.

```

1371 \newcommand*\forallglsentries}[3][\glsdefaulttype]{%
1372   \edef\@glo@list{\csname glolist@#1\endcsname}%

```

```

1373 \@for#2:=\@glo@list\do
1374 {%
1375   \ifdefempty{#2}{#3}%
1376 }%
1377 }

```

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

```
\forallglsentries[⟨glossary list⟩]{⟨cmd⟩}{⟨code⟩}
```

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

```

1378 \newcommand*\forallglsentries}[3][\@glo@types]{%
1379   \expandafter\forallglossaries\expandafter[#1]{\@@this@glo@}%
1380   {%
1381     \forglsentries[\@@this@glo@]{#2}{#3}%
1382   }%
1383 }

```

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

```
\ifglossaryexists{⟨type⟩}{⟨true-text⟩}{⟨false-text⟩}
```

where `⟨type⟩` is the glossary's label.

```

1384 \newcommand*\ifglossaryexists}[3]{%
1385   \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1386 }

```

Since the label is used to form the name of control sequences, by default UTF8 etc characters can't be used in the label. A possible workaround is to use `\scantokens`, but commands such as `\glsentrytext` will no longer be usable in sectioning, caption etc commands. If the user really wants to be able to construct a label with UTF8 characters, allow them the means to do so (but on their own head be it, if they then use entries in `\section` etc). This can be done via:

```
\renewcommand*\glsdetoklabel}[1]{\scantokens{#1\noexpand}}
```

(Note, don't use `\detokenize` or it will cause commands like `\glsaddall` to fail.) Since re-defining `\glsdetoklabel` can cause things to go badly wrong, I'm not going to mention it in the main user guide. Only advanced users who know what they're doing ought to attempt it.

`\glsdetoklabel`

```
1387 \newcommand*\glsdetoklabel}[1]{#1}
```

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

```
\ifglsentryexists{⟨label⟩}{⟨true text⟩}{⟨false text⟩}
```

where *⟨label⟩* is the entry's label.

```
1388 \newcommand{\ifglsentryexists}[3]{%
1389   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1390 }
```

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

```
\ifglsused{⟨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⟩*.

```
1391 \newcommand*{\ifglsused}[3]{%
1392   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1393 }
```

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

```
\glsdoifexists \glsdoifexists{⟨label⟩}{⟨code⟩}
```

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

```
1394 \newcommand{\glsdoifexists}[2]{%
1395   \ifglsentryexists{#1}{#2}{%
1396     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1397     has not been defined}{You need to define a glossary entry before you
1398     can use it.}}%
1399 }
```

```
\glsdoifnoexists \glsdoifnoexists{⟨label⟩}{⟨code⟩}
```

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

```
1400 \newcommand{\glsdoifnoexists}[2]{%
1401   \ifglsentryexists{#1}{#2}{%
1402     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’ has already
1403     been defined}{}}{#2}%
1404 }
```

```
\glsdoifexistsorwarn \glsdoifexistsorwarn{⟨label⟩}{⟨code⟩}
```

Generate a warning if entry specified by *⟨label⟩* doesn't exist, otherwise do *⟨code⟩*.

```
1405 \newcommand{\glsdoifexistsorwarn}[2]{%
1406   \ifglsentryexists{#1}{#2}{%
1407     \GlossariesWarning{Glossary entry ‘\glsdetoklabel{#1}’
1408     has not been defined}%
1409   }%
1410 }
```

`\glsdoifexistsordo{<label>}{<code>}{<undef code>}`

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

```
1411 \newcommand{\glsdoifexistsordo}[3]{%
1412   \ifglsentryexists{#1}{#2}{%
1413     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1414     has not been defined}{You need to define a glossary entry before you
1415     can use it.}%
1416     #3%
1417   }%
1418 }
```

`\doifglossarynoexistsordo{<label>}{<code>}{<else code>}`

If glossary given by *<label>* doesn't exist do *<code>* otherwise generate an error and do *<else code>*.

```
1419 \newcommand{\doifglossarynoexistsordo}[3]{%
1420   \ifglossaryexists{#1}%
1421   {%
1422     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1423     #3%
1424   }%
1425   {#2}%
1426 }
```

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

```
1427 \newcommand{\ifglshaschildren}[3]{%
1428   \glsdoifexists{#1}%
1429   {%
1430     \def\do@glshaschildren{#3}%
1431     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1432     \expandafter\forglentries\expandafter
1433     [\csname glo@\@gls@thislabel @type\endcsname]
1434     {\glo@label}%
1435     {%
1436       \letcs\glo@parent{glo@\glo@label @parent}%
1437       \ifdefequal\@gls@thislabel\glo@parent
1438       {%
1439         \def\do@glshaschildren{#2}%
1440         \@endfortrue
1441       }%
1442     }%
1443   }%
1444   \do@glshaschildren
1445 }%
1446 }
```

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

```
1447 \newcommand{\ifglshasparent}[3]{%
1448   \glsdoifexists{#1}%
1449   {%
1450     \ifcseempty{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1451   }%
1452 }
```

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

```
1453 \newcommand*{\ifglshasdesc}[3]{%
1454   \ifcseempty{glo@\glsdetoklabel{#1}@desc}{%
1455     {#3}%
1456     {#2}%
1457 }
```

```
sdescsuppressed \ifglsdescsuppressed{<label>}{<true part>}{<>false part>} Does <true part> if the descrip-
tion is just \nopostdesc otherwise does <>false part>.
```

```
1458 \newcommand*{\ifglsdescsuppressed}[3]{%
1459   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}{%
1460     {#2}%
1461     {#3}%
1462 }
```

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

```
1463 \newcommand*{\ifglshassymbol}[3]{%
1464   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1465   \ifdefempty\@glo@symbol
1466   {#3}%
1467   {%
1468     \ifdefequal\@glo@symbol\@gls@default@value
1469     {#3}%
1470     {#2}%
1471   }%
1472 }
```

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

```
1473 \newcommand*{\ifglshaslong}[3]{%
1474   \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1475   \ifdefempty\@glo@long
1476   {#3}%
1477   {%
1478     \ifdefequal\@glo@long\@gls@default@value
1479     {#3}%
1480     {#2}%
1481   }%
1482 }
```

```

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1483 \newcommand*{\ifglshasshort}[3]{%
1484 \letcs{\@glo@short}{glo@\glsdetoklabel{#1}@short}%
1485 \ifdefempty\@glo@short
1486 {#3}%
1487 {%
1488 \ifdequal\@glo@short\@gls@default@value
1489 {#3}%
1490 {#2}%
1491 }%
1492 }

```

```

\ifglshasfield \ifglshasfield{<field>}{<label>}{<true part>}{<false part>}

```

```

1493 \newcommand*{\ifglshasfield}[4]{%
1494 \glsdoifexists{#2}%
1495 {%
1496 \letcs{\@glo@thisvalue}{glo@\glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1497 \ifdef\@glo@thisvalue
1498 {%

```

Is defined, so now check if empty.

```

1499 \ifdefempty\@glo@thisvalue
1500 {%

```

Is empty, so doesn't have field set.

```

1501 #4%
1502 }%
1503 {%

```

Not empty, so check if set to \@gls@default@value

```

1504 \ifdequal\@glo@thisvalue\@gls@default@value
1505 {%

```

Value is set to the default value.

```

1506 #4%
1507 }%
1508 {%

```

Non-empty, non-default value. Allow user to access this value through \glscurrentfieldvalue.

```

1509 \let\glscurrentfieldvalue\@glo@thisvalue
1510 #3%
1511 }%
1512 }%
1513 }%
1514 {%

```

Field given isn't defined, so check if mapping exists.

```
1515 \@gls@fetchfield{\@gls@thisfield}{#1}%
```

If \@gls@thisfield is defined, we've found a map. If not, the field supplied doesn't exist.

```
1516 \ifdef\@gls@thisfield
1517 {%
```

Is defined, so now check if empty.

```
1518 \letcs{\@glo@thisvalue}{glo@glsetoklabel{#2}@@gls@thisfield}%
1519 \ifdefempty\@glo@thisvalue
1520 {%
```

Is empty so field hasn't been set.

```
1521 #4%
1522 }%
1523 {%
```

Isn't empty so check if it's been set to \@gls@default@value.

```
1524 \ifdequal\@glo@thisvalue\@gls@default@value
1525 {%
```

Value is set to the default value.

```
1526 #4%
1527 }%
1528 {%
```

Non-empty, non-default value. Allow user to access this value through \glscurrentfieldvalue.

```
1529 \let\glscurrentfieldvalue\@glo@thisvalue
1530 #3%
1531 }%
1532 }%
1533 }%
1534 {%
```

Not defined.

```
1535 \GlossariesWarning{Unknown entry field '#1'}%
1536 #4%
1537 }%
1538 }%
1539 }%
1540 }
```

urrentfieldvalue

```
1541 \newcommand*{\glscurrentfieldvalue}{}
```

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`

```
1542 \newcommand*{\@glo@types}{,}
```

`\ide@newglossary` If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
1543 \newcommand*\@gls@provide@newglossary{%
```

```
1544 \protected@write\@auxout{}{\string\providecommand\string\@newglossary[4]{}}%
```

Only need to do this once.

```
1545 \let\@gls@provide@newglossary\relax
```

```
1546 }
```

`\defglsentryfmt` Allow different glossaries to have different display styles.

```
1547 \newcommand*\defglsentryfmt}[2][\glsdefaultttype]{%
```

```
1548 \csgdef{gls@#1@entryfmt}{#2}%
```

```
1549 }
```

`\gls@doentryfmt`

```
1550 \newcommand*\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}
```

`\gls@forbidtextext` As a security precaution, don't allow the user to specify a 'tex' extension for any of the glossary files. (Just in case a seriously confused novice user doesn't know what they're doing.) The argument must be a control sequence whose replacement text is the requested extension.

```
1551 \newcommand*\@gls@forbidtextext}[1]{%
```

```
1552 \ifboolexpr{test {\ifdefstring{#1}{tex}}}
```

```
1553 or test {\ifdefstring{#1}{TEX}}}
```

```
1554 {%
```

```
1555 \def#1{nottex}%
```

```
1556 \PackageError{glossaries}%
```

```
1557 {Forbidden '.tex' extension replaced with '.nottex'}%
```

```
1558 {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
```

```
1559 Don't use '.tex' as an extension for a temporary file.}%
```

```
1560 }%
```

```
1561 {%
```

```
1562 }%
```

```
1563 }
```

`\gls@gobbleopt` Discard optional argument.

```
1564 \newcommand*\gls@gobbleopt{\new@ifnextchar[{\@gls@gobbleopt}{}]}
```

```
1565 \def\@gls@gobbleopt[#1]{}
```

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`

```
1566 \newcommand*{\newglossary}{\@ifstar\s@newglossary\@ns@newglossary}
```

`\s@newglossary` The starred version will construct the extension based on the label.

```
1567 \newcommand*{\s@newglossary}[2]{%
1568 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1569 }
```

`\ns@newglossary` Define the unstarred version.

```
1570 \newcommand*{\ns@newglossary}[5][glg]{%
1571 \doifglossarynoexistsordo{#2}%
1572 {%
```

Check if default has been set

```
1573 \ifundef\glsdefaultttype
1574 {%
1575 \gdef\glsdefaultttype{#2}%
1576 }{}}%
```

Add this to the list of glossary types:

```
1577 \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.

```
1578 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1579 \expandafter\edef\csname @glo@type@#2@log\endcsname{#1}%
1580 \expandafter\edef\csname @glo@type@#2@in\endcsname{#3}%
1581 \expandafter\edef\csname @glo@type@#2@out\endcsname{#4}%
1582 \expandafter\@gls@forbidtexext\csname @glo@type@#2@log\endcsname
1583 \expandafter\@gls@forbidtexext\csname @glo@type@#2@in\endcsname
1584 \expandafter\@gls@forbidtexext\csname @glo@type@#2@out\endcsname
```

Store the title:

```
1585 \expandafter\def\csname @glo@type@#2@title\endcsname{#5}%
```

```
1586 \@gls@provide@newglossary
```

```
1587 \protected@write\@auxout{}{\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsentry` by default). This can be redefined by the user later if required (see `\defglsentry`). This may already have been defined if this has been specified as a list of acronyms.

```

1588 \ifcsundef{gls@#2@entryfmt}%
1589 {%
1590   \defglsentryfmt [#2]{\glsentryfmt}%
1591 }%
1592 {}%

```

Define sort counter if required:

```
1593 \@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.)

```

1594 \@ifnextchar[{\@gls@setcounter{#2}}%
1595   {\@gls@setcounter{#2}[\glscounter]}%
1596 }%
1597 {%
1598   \gls@gobbleopt
1599 }%
1600 }

```

`\altnewglossary`

```

1601 \newcommand*{\altnewglossary}[3]{%
1602   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1603 }

```

Only define new glossaries in the preamble:

```
1604 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1605 \@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 \LaTeX , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```
1606 \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`

```

1607 \def\@gls@setcounter#1[#2]{%
1608   \expandafter\def\csname @gls@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1609   \ifglsxindy
1610     \GlsAddXdyCounters{#2}%
1611   \fi
1612 }

```

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

@gls@getcounter

```
1613 \newcommand*{\@gls@getcounter}[1]{%
1614 \csname @gls@#1@counter\endcsname
1615 }
```

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

```
1616 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1617 \@gls@do@acronymsdef
```

Define the “symbols”, “numbers” and “index” glossaries if required.

```
1618 \@gls@do@symbolsdef
```

```
1619 \@gls@do@numbersdef
```

```
1620 \@gls@do@indexdef
```

`ignoredglossary` Creates a new glossary that doesn't have associated files. This glossary is ignored by and commands that iterate over glossaries, such as `\printglossaries`, and won't work with commands like `\printglossary`. It's intended for entries that are so commonly-known they don't require a glossary.

```
1621 \newcommand*{\newignoredglossary}[1]{%
1622 \ifdefempty\@ignored@glossaries
1623 {%
1624 \edef\@ignored@glossaries{#1}%
1625 }%
1626 {%
1627 \eappto\@ignored@glossaries{,#1}%
1628 }%
1629 \csgdef{glslist@#1}{,}%
1630 \ifcsundef{gls@#1@entryfmt}%
1631 {%
1632 \defglsentryfmt[#1]{\glsentryfmt}%
1633 }%
1634 }%
1635 \ifdefempty\@gls@nohyperlist
1636 {%
1637 \renewcommand*{\@gls@nohyperlist}{#1}%
1638 }%
1639 {%
1640 \eappto\@gls@nohyperlist{,#1}%
1641 }%
1642 }
```

`ignored@glossaries` List of ignored glossaries.

```
1643 \newcommand*{\@ignored@glossaries}{}
```

`ignoredglossary` Tests if the given glossary is an ignored glossary. Expansion is used in case the first argument is a control sequence.

```

1644 \newcommand*{\ifignoredglossary}[3]{%
1645   \edef\@gls@igtype{#1}%
1646   \expandafter\DTLifinlist\expandafter
1647     {\@gls@igtype}\@ignored@glossaries}{#2}{#3}%
1648 }

```

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.

```

1649 \define@key{glossentry}{name}{%
1650 \def\@glo@name{#1}%
1651 }

```

description The description key is usually only used in the glossary, but can be made to appear in the text by redefining `\glsentryfmt` or using `\defglsentryfmt`. The description key is required when defining a new glossary entry. If a long description is required, use `\longnewglossaryentry` instead of `\newglossaryentry`.

```

1652 \define@key{glossentry}{description}{%
1653 \def\@glo@desc{#1}%
1654 }

```

descriptionplural

```

1655 \define@key{glossentry}{descriptionplural}{%
1656 \def\@glo@descplural{#1}%
1657 }

```

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 *<name>* *<description>*.

```

1658 \define@key{glossentry}{sort}{%
1659 \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.

```

1660 \define@key{glossentry}{text}{%
1661 \def\@glo@text{#1}%
1662 }

```

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.

```
1663 \define@key{glossentry}{plural}{%
1664 \def\@glo@plural{#1}%
1665 }
```

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.

```
1666 \define@key{glossentry}{first}{%
1667 \def\@glo@first{#1}%
1668 }
```

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.

```
1669 \define@key{glossentry}{firstplural}{%
1670 \def\@glo@firstplural{#1}%
1671 }
```

s@default@value

```
1672 \newcommand*{\@gls@default@value}{\relax}
```

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 `\glossentry`. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsentryfmt` (or use for `\defglsentryfmt` individual glossaries).

```
1673 \define@key{glossentry}{symbol}{%
1674 \def\@glo@symbol{#1}%
1675 }
```

symbolplural

```
1676 \define@key{glossentry}{symbolplural}{%
1677 \def\@glo@symbolplural{#1}%
1678 }
```

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

```
1679 \define@key{glossentry}{type}{%
1680 \def\@glo@type{#1}}
```

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

```
1681 \define@key{glossentry}{counter}{%
1682 \ifcsundef{c@#1}%
```

```

1683  {%
1684    \PackageError{glossaries}%
1685    {There is no counter called '#1'}%
1686    {%
1687      The counter key should have the name of a valid counter
1688      as its value%
1689    }%
1690  }%
1691  {%
1692    \def\@glo@counter{#1}%
1693  }%
1694 }

```

see The see key specifies a list of cross-references

```

1695 \define@key{glossentry}{see}{%
1696   \gls@checkseeallowed
1697   \def\@glo@see{#1}%
1698   \@glo@seeautonumberlist
1699 }

```

checkseeallowed

```

1700 \newcommand*{\gls@checkseeallowed}{%
1701   \@gls@see@noindex
1702 }

```

ed@preambleonly

```

1703 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1704   \GlossariesWarning{glossaries}%
1705   {'see' key doesn't have any effect when used in the document
1706   environment. Move the definition to the preamble
1707   after \string\makeglossaries\space
1708   or \string\makenoidxglossaries}%
1709 }

```

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

```

1710 \define@key{glossentry}{parent}{%
1711 \def\@glo@parent{#1}}

```

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

```

1712 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1713   \ifcase\nr\relax
1714     \def\@glo@prefix{\glsnonextpages}%
1715     \@gls@savenonumberlist{true}%
1716   \else
1717     \def\@glo@prefix{\glsnextpages}%
1718     \@gls@savenonumberlist{false}%
1719   \fi
1720 }

```

savenonumberlist The nonumberlist option isn't saved by default (as it just sets the prefix) which isn't a problem when the entries are defined in the preamble, but causes a problem when entries are defined in the document. In this case, the value needs to be saved so that it can be written to the .glsdefs file.

```
1721 \newcommand*{\@gls@savenonumberlist}[1]{}
```

initnonumberlist

```
1722 \newcommand*{\@gls@initnonumberlist}{}%
```

storenonumberlist

```
1723 \newcommand*{\@gls@storenonumberlist}[1]{}
```

savenonumberlist Allow the nonumberlist value to be saved.

```
1724 \newcommand*{\@gls@enablesavenonumberlist}{%
1725   \renewcommand*{\@gls@initnonumberlist}{%
1726     \undef\@glo@nonumberlist
1727   }%
1728   \renewcommand*{\@gls@savenonumberlist}[1]{%
1729     \def\@glo@nonumberlist{##1}%
1730   }%
1731   \renewcommand*{\@gls@storenonumberlist}[1]{%
1732     \ifdef\@glo@nonumberlist
1733       {%
1734         \cslet{glo@glsdetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1735       }%
1736     }%
1737   }%
1738   \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1739 }
```

Define some generic user keys. (Additional keys can be added by the user.)

user1

```
1740 \define@key{glossentry}{user1}{%
1741   \def\@glo@useri{#1}%
1742 }
```

user2

```
1743 \define@key{glossentry}{user2}{%
1744   \def\@glo@userii{#1}%
1745 }
```

user3

```
1746 \define@key{glossentry}{user3}{%
1747   \def\@glo@useriii{#1}%
1748 }
```

user4

```
1749 \define@key{glossentry}{user4}{%
1750   \def\@glo@useriv{#1}%
1751 }
```

user5

```
1752 \define@key{glossentry}{user5}{%
1753   \def\@glo@userv{#1}%
1754 }
```

user6

```
1755 \define@key{glossentry}{user6}{%
1756   \def\@glo@uservi{#1}%
1757 }
```

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.

```
1758 \define@key{glossentry}{short}{%
1759   \def\@glo@short{#1}%
1760 }
```

shortplural This key is provided for use by `\newacronym`.

```
1761 \define@key{glossentry}{shortplural}{%
1762   \def\@glo@shortpl{#1}%
1763 }
```

long This key is provided for use by `\newacronym`.

```
1764 \define@key{glossentry}{long}{%
1765   \def\@glo@long{#1}%
1766 }
```

longplural This key is provided for use by `\newacronym`.

```
1767 \define@key{glossentry}{longplural}{%
1768   \def\@glo@longpl{#1}%
1769 }
```

`\@glsnname` Define command to generate error if name key is missing.

```
1770 \newcommand*\@glsnname{%
1771   \PackageError{glossaries}{name key required in
1772   \string\newglossaryentry\space for entry '@glo@label'}{You
1773   haven't specified the entry name}}
```

`\@glsnodesc` Define command to generate error if description key is missing.

```
1774 \newcommand*\@glsnodesc{%
1775   \PackageError{glossaries}
1776   {%
1777     description key required in \string\newglossaryentry\space
1778     for entry '@glo@label'%
1779   }}
```



```

1779 }%
1780 {%
1781     You haven't specified the entry description%
1782 }%
1783 }%

```

`lsdefaultplural` Now obsolete. Don't use.

```
1784 \newcommand*{\@glsdefaultplural}{}
```

`missingnumberlist` Define a command to generate warning when numberlist not set.

```

1785 \newcommand*{\@gls@missingnumberlist}[1]{%
1786   ??%
1787   \ifglssavenumberlist
1788     \GlossariesWarning{Missing number list for entry '#1'.
1789       Maybe makeglossaries + rerun required}%
1790   \else
1791     \PackageError{glossaries}%
1792       {Package option 'savenumberlist=true' required}%
1793     {%
1794       You must use the 'savenumberlist' package option
1795       to reference location lists.%
1796     }%
1797   \fi
1798 }

```

`@glsdefaultsort` Define command to set default sort.

```
1799 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

`\gls@level` Register to increment entry levels.

```
1800 \newcount\gls@level
```

`@noexpand@field`

```

1801 \newcommand{\@@gls@noexpand@field}[3]{%
1802   \expandafter\global\expandafter
1803     \let\csname glo@#1@#2\endcsname#3%
1804 }

```

`noexpand@fields`

```

1805 \newcommand{\@gls@noexpand@fields}[4]{%
1806   \ifcsdef{gls@assign@#3@field}
1807   {%
1808     \ifdefequal{#4}{\@gls@default@value}%
1809     {%
1810       \edef\@gls@value{\expandonce{#1}}%
1811       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1812     }%
1813   }%
1814     \csuse{gls@assign@#3@field}{#2}{#4}%

```

```

1815     }%
1816 }%
1817 {%
1818   \ifdefequal{#4}{\@gls@default@value}%
1819   {%
1820     \edef\@gls@value{\expandonce{#1}}%
1821     \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
1822   }%
1823   {%
1824     \@@gls@noexpand@field{#2}{#3}{#4}%
1825   }%
1826 }%
1827 }

```

ls@expand@field

```

1828 \newcommand{\@@gls@expand@field}[3]{%
1829   \expandafter
1830   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1831 }

```

s@expand@fields

```

1832 \newcommand{\@gls@expand@fields}[4]{%
1833   \ifcsdef{gls@assign@#3@field}
1834   {%
1835     \ifdefequal{#4}{\@gls@default@value}%
1836     {%
1837       \edef\@gls@value{\expandonce{#1}}%
1838       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1839     }%
1840     {%
1841       \expandafter\@gls@startswithexpandonce#4\relax\relax@gls@endcheck
1842       {%
1843         \@@gls@expand@field{#2}{#3}{#4}%
1844       }%
1845       {%
1846         \csuse{gls@assign@#3@field}{#2}{#4}%
1847       }%
1848     }%
1849   }%
1850   {%
1851     \ifdefequal{#4}{\@gls@default@value}%
1852     {%
1853       \@@gls@expand@field{#2}{#3}{#1}%
1854     }%
1855     {%
1856       \@@gls@expand@field{#2}{#3}{#4}%
1857     }%
1858   }%
1859 }

```

switexpandonce

```
1860 \def\@gls@expandonce{\expandonce}
1861 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1862   \def\@gls@tmp{#1}%
1863   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1864 }
```

gls@assign@field

```
\gls@assign@field{<def value>}{<label>}{<field>}{<tmp cs>}
```

Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If *<tmp cs>* is *<@gls@default@value>*, *<def value>* is used instead.

```
1865 \let\gls@assign@field\@gls@expand@fields
```

gls@expand@fields

Fully expand values when assigning fields (except for specific fields that are overridden by `\glssetnoexpandfield`).

```
1866 \newcommand*{\gls@expand@fields}{%
1867   \let\gls@assign@field\@gls@expand@fields
1868 }
```

gls@noexpand@fields

Don't expand values when assigning fields (except for specific fields that are overridden by `\glssetexpandfield`).

```
1869 \newcommand*{\gls@noexpand@fields}{%
1870   \let\gls@assign@field\@gls@noexpand@fields
1871 }
```

newglossaryentry

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

```
1872 \newrobustcmd{\newglossaryentry}[2]{%
```

Check to see if this glossary entry has already been defined:

```
1873   \glsdoifnoexists{#1}%
1874   {%
1875     \gls@defglossaryentry{#1}{#2}%
1876   }%
1877 }
```

newglossaryentry

The definition of `\newglossaryentry` is changed at the start of the document environment. The see key doesn't work for entries that have been defined in the document environment.

```
1878 \newcommand*{\gls@defdocnewglossaryentry}{%
1879   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1880   \let\newglossaryentry\new@glossaryentry
1881 }
```

provideglossaryentry

Like `\newglossaryentry` but does nothing if the entry has already been defined.

```
1882 \newrobustcmd{\provideglossaryentry}[2]{%
```

```

1883 \ifglsentryexists{#1}%
1884 {}%
1885 {%
1886   \gls@defglossaryentry{#1}{#2}%
1887 }%
1888 }
1889 \@onlypreamble{\provideglossaryentry}

```

`w@glossaryentry` For use in document environment.

```

1890 \newrobustcmd{\new@glossaryentry}[2]{%
1891   \ifundef\@gls@deffile
1892   {%
1893     \global\newwrite\@gls@deffile
1894     \immediate\openout\@gls@deffile=\jobname.glsdefs
1895   }%
1896   {}%
1897   \ifglsentryexists{#1}{}%
1898   {%
1899     \gls@defglossaryentry{#1}{#2}%
1900   }%
1901   \@gls@writedef{#1}%
1902 }
1903 \AtBeginDocument
1904 {
1905   \@gls@enablesavenonumberlist
1906   \makeatletter
1907   \InputIfFileExists{\jobname.glsdefs}{}{}%
1908   \makeatother
1909   \gls@defdocnewglossaryentry
1910 }
1911 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

```

`\@gls@writedef` Writes glossary entry definition to `\@gls@deffile`.

```

1912 \newcommand*{\@gls@writedef}[1]{%
1913   \immediate\write\@gls@deffile
1914   {%
1915     \string\ifglsentryexists{#1}{}\glspercentchar^^J%
1916     \expandafter\@gobble\string\{\glspercentchar^^J%
1917     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1918     \expandafter\@gobble\string\{\glspercentchar%
1919   }%

```

Write key value information:

```

1920 \@for\@gls@map:=\@gls@keymap\do
1921 {%
1922   \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
1923   \ifdef\glo@value
1924   {%
1925     \@onelevel@sanitize\glo@value
1926     \immediate\write\@gls@deffile

```

```

1927   {%
1928     \expandafter \@firstoftwo \@gls@map
1929     =\expandafter \@gobble\string\{\@glo@value\expandafter \@gobble\string\},%
1930     \glspercentchar
1931   }%
1932 }%
1933 {}%
1934 }%

```

Provide hook:

```

1935 \glswritedefhook
1936 \immediate\write \@gls@deffile
1937 {%
1938     \glspercentchar^^J%
1939     \expandafter \@gobble\string\}\glspercentchar^^J%
1940     \expandafter \@gobble\string\}\glspercentchar%
1941 }%
1942 }

```

`\@gls@keymap` List of entry definition key names and corresponding tag in control sequence used to store the value.

```

1943 \newcommand*{\@gls@keymap}{%
1944   {name}{name},%
1945   {sort}{sortvalue},% unescaped sort value
1946   {type}{type},%
1947   {first}{first},%
1948   {firstplural}{firstpl},%
1949   {text}{text},%
1950   {plural}{plural},%
1951   {description}{desc},%
1952   {descriptionplural}{descplural},%
1953   {symbol}{symbol},%
1954   {symbolplural}{symbolplural},%
1955   {user1}{useri},%
1956   {user2}{userii},%
1957   {user3}{useriii},%
1958   {user4}{useriv},%
1959   {user5}{userv},%
1960   {user6}{uservi},%
1961   {long}{long},%
1962   {longplural}{longpl},%
1963   {short}{short},%
1964   {shortplural}{shortpl},%
1965   {counter}{counter},%
1966   {parent}{parent}%
1967 }

```

`\@gls@fetchfield` `\@gls@fetchfield{<cs>}{<field>}`

Fetches the internal field label from the given user *<field>* and stores in *<cs>*.

```
1968 \newcommand*{\@gls@fetchfield}[2]{%
  Ensure user field name is fully expanded
1969   \edef\@gls@thisval{#2}%
  Iterate through known mappings until we find the one for this field.
1970   \@for\@gls@map:=\@gls@keymap\do{%
1971     \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
1972     \ifdefequal{\@this@key}{\@gls@thisval}%
1973       {%
  Found it.
1974         \edef#1{\expandafter\@secondoftwo\@gls@map}%
  Break out of loop.
1975         \@endfortrue
1976       }%
1977     {}%
1978   }%
1979 }
```

`\glsaddstoragekey`

```
\glsaddstoragekey{<key>}{<default value>}{<no link cs>}
```

Similar to `\glsaddkey` but intended for keys whose values aren't explicitly used in the document, but might be required behind the scenes by other commands.

```
1980 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
  Starred version switches on expansion for this key.
1981 \newcommand*{\@sglsaddstoragekey}[1]{%
1982   \key@ifundefined{glossentry}{#1}%
1983   {%
1984     \expandafter\newcommand\expandafter*\expandafter
1985     {\csname gls@assign@#1@field\endcsname}[2]{%
1986       \@gls@expand@field{##1}{#1}{##2}%
1987     }%
1988   }%
1989   {}%
1990   \@glsaddstoragekey{#1}%
1991 }
```

Unstarred version doesn't override default expansion.

```
1992 \newcommand*{\@glsaddstoragekey}[3]{%
  Check the specified key doesn't already exist.
1993   \key@ifundefined{glossentry}{#1}%
1994   {%
  Set up the key.
1995     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1996     \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
1997 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
1998 \appto\@newglossaryentryposthook{%  
1999 \letcs{\@glo@tmp}{@glo@#1}%  
2000 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%  
2001 }%
```

Define the no-link commands.

```
2002 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%  
2003 }%  
2004 {%  
2005 \PackageError{glossaries}{Key ‘#1’ already exists}{}%  
2006 }%  
2007 }
```

`\glsaddkey`

```
\glsaddkey{<key>}{<default value>}{<no link cs>}{<no link ucfirst  
cs>}{<link cs>}{<link ucfirst cs>}{<link allcaps cs>}
```

Allow user to add their own custom keys.

```
2008 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2009 \newcommand*{\@sglsaddkey}[1]{%  
2010 \key@ifundefined{glossentry}{#1}%  
2011 {%  
2012 \expandafter\newcommand\expandafter*\expandafter  
2013 {\csname gls@assign@#1@field\endcsname}[2]{%  
2014 \@gls@expand@field{##1}{#1}{##2}%  
2015 }%  
2016 }%  
2017 }%  
2018 \@glsaddkey{#1}%  
2019 }
```

Unstarred version doesn't override default expansion.

```
2020 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2021 \key@ifundefined{glossentry}{#1}%  
2022 {%
```

Set up the key.

```
2023 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%  
2024 \appto\@gls@keymap{,}{#1}{#1}}%
```

Set the default value.

```
2025 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2026 \appto\@newglossaryentryposthook{%
2027   \letcs{\@glo@tmp}{\@glo@#1}%
2028   \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2029   }%
```

Define the no-link commands.

```
2030 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2031 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

Now for the commands with links. First the version with no case change:

```
2032 \ifcsdef{@gls@user@#1@}%
2033 {%
2034   \PackageError{glossaries}%
2035   {Can't define '\string#5' as helper command
2036   '\expandafter\string\csname @gls@user@#1@endcsname' already exists}%
2037   }%
2038 }%
2039 {%
2040   \expandafter\newcommand\expandafter*\expandafter
2041   {\csname @gls@user@#1@endcsname}[2][ ]{%
2042     \new@ifnextchar[%
2043       {\csuse{@gls@user@#1@}{##1}{##2}}%
2044       {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
2045   \csdef{@gls@user@#1@}##1##2[##3]{%
2046     \@gls@field@link{##1}{##2}{#3{##2}##3}%
2047   }%
2048   \newrobustcmd*{#5}{%
2049     \expandafter\@gls@hyp@opt\csname @gls@user@#1@endcsname}%
2050   }%
```

Next the version with the first letter converted to upper case:

```
2051 \ifcsdef{@Gls@user@#1@}%
2052 {%
2053   \PackageError{glossaries}%
2054   {Can't define '\string#6' as helper command
2055   '\expandafter\string\csname @Gls@user@#1@endcsname' already exists}%
2056   }%
2057 }%
2058 {%
2059   \expandafter\newcommand\expandafter*\expandafter
2060   {\csname @Gls@user@#1@endcsname}[2][ ]{%
2061     \new@ifnextchar[%
2062       {\csuse{@Gls@user@#1@}{##1}{##2}}%
2063       {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
2064   \csdef{@Gls@user@#1@}##1##2[##3]{%
2065     \@gls@field@link{##1}{##2}{#4{##2}##3}%
2066   }%
2067   \newrobustcmd*{#6}{%
```



```

2068     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2069 }%

    Finally the all caps version:
2070     \ifcsdef{@GLS@user@#1@}%
2071     {%
2072         \PackageError{glossaries}%
2073         {Can't define '\string#7' as helper command
2074         '\expandafter\string\csname @GLS@user@#1\endcsname' already exists}%
2075         }%
2076     }%
2077     {%

2078     \expandafter\newcommand\expandafter*\expandafter
2079     {\csname @GLS@user@#1\endcsname}[2][ ]{%
2080         \new@ifnextchar[%
2081             {\csuse{@GLS@user@#1@}{##1}{##2}}%
2082             {\csuse{@GLS@user@#1@}{##1}{##2}[ ]}}%
2083     \csdef{@GLS@user@#1@}##1##2[##3]{%
2084         \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2085     }%
2086     \newrobustcmd*{#7}{%
2087         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2088     }%
2089 }%
2090 {%
2091     \PackageError{glossaries}{Key '#1' already exists}{}%
2092 }%
2093 }

```

```
\glsfieldxdef \glsfieldxdef{<label>}{<field>}{<definition>}
```

```

2094 \newcommand{\glsfieldxdef}[3]{%
2095     \glsdoifexists{#1}%
2096     {%
2097         \edef\@glo@label{\glsdetoklabel{#1}}%
2098         \ifcsdef{glo@\@glo@label @#2}%
2099         {%
2100             \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2101         }%
2102         {%
2103             \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2104         }%
2105     }%
2106 }

```

```
\glsfielddedef \glsfielddedef{<label>}{<field>}{<definition>}
```

```

2107 \newcommand{\glsfieldedef}[3]{%
2108   \glsdoifexists{#1}%
2109   {%
2110     \edef\@glo@label{\glsdetoklabel{#1}}%
2111     \ifcsdef{glo@\@glo@label @#2}%
2112     {%
2113       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2114     }%
2115     {%
2116       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2117     }%
2118   }%
2119 }

```

`\glsfieldgdef` `\glsfieldgdef{<label>}{<field>}{<definition>}`

```

2120 \newcommand{\glsfieldgdef}[3]{%
2121   \glsdoifexists{#1}%
2122   {%
2123     \edef\@glo@label{\glsdetoklabel{#1}}%
2124     \ifcsdef{glo@\@glo@label @#2}%
2125     {%
2126       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2127     }%
2128     {%
2129       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2130     }%
2131   }%
2132 }

```

`\glsfieldddef` `\glsfieldddef{<label>}{<field>}{<definition>}`

```

2133 \newcommand{\glsfieldddef}[3]{%
2134   \glsdoifexists{#1}%
2135   {%
2136     \edef\@glo@label{\glsdetoklabel{#1}}%
2137     \ifcsdef{glo@\@glo@label @#2}%
2138     {%
2139       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2140     }%
2141     {%
2142       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2143     }%
2144   }%

```

2145 }

`\glsfieldfetch` `\glsfieldfetch{<label>}{<field>}{<cs>}`

Fetches the value of the given field and stores in the given control sequence.

```
2146 \newcommand{\glsfieldfetch}[3]{%
2147   \glsdoifexists{#1}%
2148   {%
2149     \edef\@glo@label{\glsdetoklabel{#1}}%
2150     \ifcsdef{glo@\@glo@label @#2}%
2151     {%
2152       \letcs#3{glo@\@glo@label @#2}%
2153     }%
2154   }%
2155   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2156 }%
2157 }%
2158 }
```

`\ifglsfieldeq` `\ifglsfieldeq{<label>}{<field>}{<string>}{<true>}{<false>}`

Tests if the value of the given field is equal to the given string.

```
2159 \newcommand{\ifglsfieldeq}[5]{%
2160   \glsdoifexists{#1}%
2161   {%
2162     \edef\@glo@label{\glsdetoklabel{#1}}%
2163     \ifcsdef{glo@\@glo@label @#2}%
2164     {%
2165       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2166     }%
2167   }%
2168   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2169 }%
2170 }%
2171 }
```

`\ifglsfielddefeq` `\ifglsfielddefeq{<label>}{<field>}{<command>}{<true>}{<false>}`

Tests if the value of the given field is equal to the replacement text of the given command.

```
2172 \newcommand{\ifglsfielddefeq}[5]{%
2173   \glsdoifexists{#1}%
2174   {%
2175     \edef\@glo@label{\glsdetoklabel{#1}}%
2176     \ifcsdef{glo@\@glo@label @#2}%
2177     {%
```

```

2178     \expandafter\ifdefstrequal
2179     \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2180 }%
2181 {%
2182     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2183 }%
2184 }%
2185 }

```

```
\ifglsfieldcseq <label> <field> {<cs name>} {<true>} {<false>}
```

As above but uses \ifcsstrequal instead of \ifdefstrequal

```

2186 \newcommand{\ifglsfieldcseq}[5]{%
2187   \glsdoifexists{#1}%
2188   {%
2189     \edef\@glo@label{\glsdetoklabel{#1}}%
2190     \ifcsdef{glo@\@glo@label @#2}%
2191     {%
2192       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2193     }%
2194     {%
2195       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2196     }%
2197   }%
2198 }

```

gls.writedefhook

```
2199 \newcommand*{\gls.writedefhook}{}

```

gls.assign@desc

```

2200 \newcommand*{\gls.assign@desc}[1]{%
2201   \gls.assign@field{#1}{desc}{\@glo@desc}%
2202   \gls.assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2203 }

```

ew.glossaryentry

```

2204 \newcommand{\longnewglossaryentry}[3]{%
2205   \glsdoifnoexists{#1}%
2206   {%
2207     \bgroup
2208     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2209     \long\def\@newglossaryentryprehook{%
2210       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2211       \@org@newglossaryentryprehook
2212     }%
2213     \renewcommand*{\gls.assign@desc}[1]{%
2214       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

```

2215     \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2216     }
2217     \gls@defglossaryentry{#1}{#2}%
2218   \egroup
2219 }
2220 }

```

Only allowed in the preamble. (Otherwise a long description could cause problems when writing the entry definition to the temporary file.)

```
2221 \@onlypreamble{\longnewglossaryentry}
```

`deglossaryentry` As the above but only defines the entry if it doesn't already exist.

```

2222 \newcommand{\longprovideglossaryentry}[3]{%
2223   \ifglentryexists{#1}{}%
2224   {\longnewglossaryentry{#1}{#2}{#3}}%
2225 }
2226 \@onlypreamble{\longprovideglossaryentry}

```

`defglossaryentry` `\gls@defglossaryentry{<label>}{<key-val list>}`

Defines a new entry without checking if it already exists.

```
2227 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of `\GlsSetQuote`:

```
2228   \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2229   \edef\@glo@label{\glsdetoklabel{#1}}%
```

Provide a means for user defined keys to reference the label:

```
2230   \let\glslabel\@glo@label
```

Set up defaults. If the name or description keys are omitted, an error will be generated.

```
2231   \let\@glo@name\@glsnname
```

```
2232   \let\@glo@desc\@glsnodesc
```

```
2233   \let\@glo@descplural\@gls@default@value
```

```
2234   \let\@glo@type\@gls@default@value
```

```
2235   \let\@glo@symbol\@gls@default@value
```

```
2236   \let\@glo@symbolplural\@gls@default@value
```

```
2237   \let\@glo@text\@gls@default@value
```

```
2238   \let\@glo@plural\@gls@default@value
```

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

```
2239   \let\@glo@first\@gls@default@value
```

```
2240   \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2241 \let@glo@sort\@gls@default@value
```

Set the default counter:

```
2242 \let@glo@counter\@gls@default@value
```

```
2243 \def@glo@see{}%
```

```
2244 \def@glo@parent{}%
```

```
2245 \def@glo@prefix{}%
```

Initialise nonnumberlist setting if we're in the document environment.

```
2246 \@gls@initnonnumberlist
```

```
2247 \def@glo@useri{}%
```

```
2248 \def@glo@userii{}%
```

```
2249 \def@glo@useriii{}%
```

```
2250 \def@glo@useriv{}%
```

```
2251 \def@glo@userv{}%
```

```
2252 \def@glo@servi{}%
```

```
2253 \def@glo@short{}%
```

```
2254 \def@glo@shortpl{}%
```

```
2255 \def@glo@long{}%
```

```
2256 \def@glo@longpl{}%
```

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

```
2257 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2258 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2259 \ifundef@glsdefaultttype
```

```
2260 {%
```

```
2261 \PackageError{glossaries}%
```

```
2262 {No default glossary type (have you used 'nomain' by mistake?)}%
```

```
2263 {If you use package option 'nomain' you must define
```

```
2264 a new glossary before you can define entries}%
```

```
2265 }%
```

```
2266 {}%
```

Assign type. This must be fully expandable

```
2267 \gls@assign@field{\glsdefaultttype}{\@glo@label}{type}{\@glo@type}%
```

```
2268 \edef@glo@type{\glsentrytype{\@glo@label}}%
```

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

```
2269 \ifcsundef{glolist@\@glo@type}%
```

```
2270 {%
```

```
2271 \PackageError{glossaries}%
```

```

2272     {Glossary type ‘\@glo@type’ has not been defined}%
2273     {You need to define a new glossary type, before making entries
2274     in it}%
2275     }%
2276     {%

    Check if it's an ignored glossary
2277     \ifignoredglossary\@glo@type
2278     {%

    The description may be omitted for an entry in an ignored glossary.
2279     \ifx\@glo@desc\@glsnodesc
2280     \let\@glo@desc\@empty
2281     \fi
2282     }%
2283     {%
2284     }%
2285     \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2286     \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2287     \@glo@list@{\@glo@label},}%
2288     }%

    Initialise level to 0.
2289     \gls@level=0\relax

    Has this entry been assigned a parent?
2290     \ifx\@glo@parent\@empty

    Doesn't have a parent. Set \glo@<label>@parent to empty.
2291     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2292     \else

    Has a parent. Check to ensure this entry isn't its own parent.
2293     \ifdefequal\@glo@label\@glo@parent%
2294     {%
2295     \PackageError{glossaries}{Entry ‘\@glo@label’ can't be its own parent}{}%
2296     \def\@glo@parent{}%
2297     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2298     }%
2299     {%

    Check the parent exists:
2300     \ifglsentryexists{\@glo@parent}%
2301     {%

    Parent exists. Set \glo@<label>@parent.
2302     \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2303     \@glo@parent}%

    Determine level.
2304     \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2305     \advance\gls@level by 1\relax

```

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

```
2306     \ifx\@glo@name\@gls@name
2307     \expandafter\let\expandafter\@glo@name
2308     \csname glo@\@glo@parent @name\endcsname
```

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

```
2309     \ifx\@glo@plural\@gls@default@value
2310     \expandafter\let\expandafter\@glo@plural
2311     \csname glo@\@glo@parent @plural\endcsname
2312     \fi
2313     \fi
2314     }%
2315     {%
```

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

```
2316     \PackageError{glossaries}%
2317     {%
2318     Invalid parent '@glo@parent'
2319     for entry '@glo@label' - parent doesn't exist%
2320     }%
2321     {%
2322     Parent entries must be defined before their children%
2323     }%
2324     \def\@glo@parent{}%
2325     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2326     }%
2327     }%
2328     \fi
```

Set the level for this entry

```
2329     \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%
```

Define commands associated with this entry:

```
2330     \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2331     \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2332     \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2333     \expandafter\gls@assign@field\expandafter
2334     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2335     {\@glo@label}{plural}{\@glo@plural}%
2336     \expandafter\gls@assign@field\expandafter
2337     {\csname glo@\@glo@label @text\endcsname}%
2338     {\@glo@label}{first}{\@glo@first}%
```

If first has been specified, make the default by appending \glspluralsuffix, otherwise make the default the value of the plural key.

```
2339     \ifx\@glo@first\@gls@default@value
2340     \expandafter\gls@assign@field\expandafter
2341     {\csname glo@\@glo@label @plural\endcsname}%
2342     {\@glo@label}{firstpl}{\@glo@firstplural}%
2343     \else
2344     \expandafter\gls@assign@field\expandafter
```



```

2345     {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2346     {\@glo@label}{firstpl}{\@glo@firstplural}%
2347 \fi

2348 \ifcsundef{@glo@type@\@glo@type @counter}%
2349 {%
2350     \def\@glo@defaultcounter{\glscounter}%
2351 }%
2352 {%
2353     \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2354 }%
2355 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2356 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2357 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2358 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2359 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2360 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2361 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2362 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2363 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2364 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2365 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2366 \ifx\@glo@name\@glsnoname
2367     \@glsnoname
2368     \let\@glo@name\@gls@default@value
2369 \fi
2370 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2371 \ifcsundef{glo@\@glo@label @numberlist}%
2372 {%
2373     \csxdef{glo@\@glo@label @numberlist}{%
2374         \noexpand\@gls@missingnumberlist{\@glo@label}}%
2375 }%
2376 {}%

```

Store nonumberlist setting if we're in the document environment.

```

2377 \@gls@storenonumberlist{\@glo@label}%

```

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.

```

2378 \def\@glo@@desc{\@glo@first}%
2379 \ifx\@glo@desc\@glo@@desc
2380     \let\@glo@desc\@glo@first
2381 \fi
2382 \ifx\@glo@desc\@glsnodesc
2383     \@glsnodesc
2384     \let\@glo@desc\@gls@default@value
2385 \fi
2386 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```
2387 \gls@defs@sort{\@glo@type}{\@glo@label}%
2388 \def\@glo@@symbol{\@glo@text}%
2389 \ifx\@glo@symbol\@glo@@symbol
2390 \let\@glo@symbol\@glo@text
2391 \fi
2392 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2393 \expandafter
2394 \gls@assign@field\expandafter
2395 {\csname glo@\@glo@label @symbol\endcsname}
2396 {\@glo@label}{symbolplural}{\@glo@symbolplural}%
```

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

```
2397 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2398 \noexpand\global
2399 \noexpand\let\expandafter\noexpand
2400 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2401 }%
2402 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2403 \noexpand\global
2404 \noexpand\let\expandafter\noexpand
2405 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2406 }%
2407 \csname glo@\@glo@label @flagfalse\endcsname
```

Sort out any cross-referencing if required.

```
2408 \ifdefvoid\@glo@see
2409 {}%
2410 {%
2411 \protected@edef\@do@glsee{%
2412 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
2413 \noexpand\@nil
2414 \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
2415 \@do@glsee
2416 }%
```

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

```
2417 \ifignoredglossary\@glo@type
2418 {%
2419 \csdef{glo@\@glo@label @index}{}%
2420 }
2421 {%
2422 \do@glo@storeentry{\@glo@label}%
2423 }%
```

Define entry counters if enabled:

```
2424 \@newglossaryentry@defcounters
```

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

```

2425 \@newglossaryentryposthook
2426 }

```

aryentryprehook Allow extra information to be added to glossary entries:

```

2427 \newcommand*{\@newglossaryentryprehook}{}

```

ryentryposthook Allow extra information to be added to glossary entries:

```

2428 \newcommand*{\@newglossaryentryposthook}{}

```

try@defcounters

```

2429 \newcommand*{\@newglossaryentry@defcounters}{}

```

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

```

2430 \newcommand*\glsmoveentry [2]{%
2431   \edef\glo@thislabel{\glsdetoklabel{#1}}%
2432   \edef\glo@type{\csname glo@\glo@thislabel @type\endcsname}%
2433   \def\glo@list{,%}
2434   \forglsentries[\glo@type]{\glo@label}%
2435   {%
2436     \ifdefequal\glo@thislabel\glo@label
2437       {\\eappto\glo@list{\glo@label,}}%
2438     }%
2439     \cslet{glo@list@\glo@type}{\glo@list}%
2440     \csdef{glo@\glo@thislabel @type}{#2}%
2441 }

```

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

```

2442 \ifglxindy
2443   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2444 \else
2445   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2446 \fi

```

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

```

2447 \ifglxindy
2448   \newcommand*{\@glossarysubentryfield}{%
2449     \string\subglossentry}
2450 \else
2451   \newcommand*{\@glossarysubentryfield}{%
2452     \string\subglossentry}
2453 \fi

```

\glo@storeentry \@glo@storeentry{<label>}

Determine the format to write the entry in the glossary output (.glo) file. The argument is the entry's label (should already have been de-tok'ed if required). The result is stored in `\glo@<label>@index`, where *<label>* is the entry's label. (This doesn't include any formatting or location information.)

```
2454 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2455 \edef\@glo@esclabel{#1}%
```

```
2456 \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2457 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
```

```
2458 \@gls@checkmkidxchars\@glo@sort
```

Same again for the name string. Escape any special characters in the prefix

```
2459 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2460 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2461 \ifglxindy
```

Store using xindy syntax.

```
2462 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2463 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2464 (\string"\@glo@sort\string" %
```

```
2465 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
```

```
2466 }%
```

```
2467 \else
```

Entry has a parent

```
2468 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2469 \csname glo@\@glo@parent @index\endcsname
```

```
2470 (\string"\@glo@sort\string" %
```

```
2471 \string"\@glo@prefix\@glossarysubentryfield
```

```
2472 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
```

```
2473 }%
```

```
2474 \fi
```

```
2475 \else
```

Store using makeindex syntax.

```
2476 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2477 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2478 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2479 \@glo@sort\@gls@actualchar\@glo@prefix
```

```
2480 \@glossaryentryfield{\@glo@esclabel}%
```

```

2481     }%
2482   \else
      Entry has a parent
2483     \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2484       \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2485       \@glo@sort\@gls@actualchar\@glo@prefix
2486       \@glossarysubentryfield
2487       {\csname glo@#1@level\endcsname}\@glo@esclabel}%
2488     }%
2489   \fi
2490 \fi
2491 }

```

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, but first we need to know if we're in `amsmath`'s align environment's measuring pass.

`@ifnotmeasuring`

```

2492 \AtBeginDocument{%
2493   \@ifpackageloaded{amsmath}%
2494   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2495   }%
2496 }
2497 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2498   \ifmeasuring@
2499   \else
2500     #1%
2501   \fi
2502 }
2503 \newcommand*\gls@ifnotmeasuring[1]{#1}

```

`\glsreset` 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.

```

2504 \newcommand*{\glsreset}[1]{%
2505   \gls@ifnotmeasuring
2506   {%
2507     \glsdoifexists{#1}%
2508     {%
2509       \@glsreset{#1}%
2510     }%
2511   }%
2512 }

```

`\glslocalreset` As above, but with only a local effect:

```

2513 \newcommand*\glslocalreset}[1]{%
2514   \gls@ifnotmeasuring
2515   {%
2516     \glsdoifexists{#1}%
2517     {%
2518       \@glslocalreset{#1}%
2519     }%
2520   }%
2521 }

```

`\glsunset` 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.

```

2522 \newcommand*\glsunset}[1]{%
2523   \gls@ifnotmeasuring
2524   {%
2525     \glsdoifexists{#1}%
2526     {%
2527       \@glsunset{#1}%
2528     }%
2529   }%
2530 }

```

`\glslocalunset` As above, but with only a local effect:

```

2531 \newcommand*\glslocalunset}[1]{%
2532   \gls@ifnotmeasuring
2533   {%
2534     \glsdoifexists{#1}%
2535     {%
2536       \@glslocalunset{#1}%
2537     }%
2538   }%
2539 }

```

`\@glslocalunset` Local unset. This defaults to just `\@glslocalunset` but is changed by `\glsenableentrycount`.

```

2540 \newcommand*\@glslocalunset{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2541 \newcommand*\@@glslocalunset}[1]{%
2542   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2543 }

```

`\@glsunset` Global unset. This defaults to just `\@glsunset` but is changed by `\glsenableentrycount`.

```

2544 \newcommand*\@glsunset{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2545 \newcommand*\@@glsunset}[1]{%
2546   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2547 }

```

`\@glslocalreset` Local reset. This defaults to just `\@@glslocalreset` but is changed by `\glsenableentrycount`.

```
2548 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`@@glslocalreset` Local reset without checks.

```
2549 \newcommand*{\@@glslocalreset}[1]{%
2550   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2551 }
```

`\@glsreset` Global reset. This defaults to just `\@@glsreset` but is changed by `\glsenableentrycount`.

```
2552 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2553 \newcommand*{\@@glsreset}[1]{%
2554   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2555 }
```

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

`\glsresetall`

```
2556 \newcommand*{\glsresetall}[1][\@glo@types]{%
2557   \forallglsentries[#1]{\@glsentry}%
2558   {%
2559     \glsreset{\@glsentry}%
2560   }%
2561 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2562 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2563   \forallglsentries[#1]{\@glsentry}%
2564   {%
2565     \glslocalreset{\@glsentry}%
2566   }%
2567 }
```

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

`\glsunsetall`

```
2568 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2569   \forallglsentries[#1]{\@glsentry}%
2570   {%
2571     \glsunset{\@glsentry}%
2572   }%
2573 }
```

As above, but with only a local effect:

lslocalunsetall

```
2574 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2575   \forallglsentries[#1]{\@glsentry}%
2576   {%
2577     \glslocalunset{\@glsentry}%
2578   }%
2579 }
```

1.9 Keeping Track of How Many Times an Entry Has Been Unset

Version 4.14 introduced `\glsenableentrycount` that keeps track of how many times an entry is marked as used. The counter is reset back to zero when the first use flag is reset. Note that although the word “counter” is used here, it’s not an actual \LaTeX counter or even an explicit \TeX count register but is just a macro. Any of the commands that use `\glsunset` or `\glslocalunset`, such as `\gls`, will automatically increment this value. Commands that don’t modify the first use flag (such as `\glstext` or `\glsentrytext`) don’t modify this value.

`entry@defcounters` Define entry fields to keep track of how many times that entry has been marked as used.

```
2580 \newcommand*{\@newglossaryentry@defcounters}{%
2581   \csdef{glo@\@glo@label @currcount}{0}%
2582   \csdef{glo@\@glo@label @prevcount}{0}%
2583 }
```

`enableentrycount` Enables tracking of how many times an entry has been marked as used.

```
2584 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2585 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2586 \renewcommand*{\gls@defdocnewglossaryentry}{%
2587   \renewcommand*\newglossaryentry[2]{%
2588     \PackageError{glossaries}{\string\newglossaryentry\space
2589     may only be used in the preamble when entry counting has
2590     been activated}{If you use \string\glsenableentrycount\space
2591     you must place all entry definitions in the preamble not in
2592     the document environment}%
2593   }%
2594 }
```

Define commands `\glsentrycurrcount` and `\glsentryprevcount` to access these new fields. Default to zero if undefined.

```
2595 \newcommand*{\glsentrycurrcount}[1]{%
2596   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2597   {0}{\@gls@entry@field{##1}{currcount}}%
2598 }%
2599 \newcommand*{\glsentryprevcount}[1]{%
```



```

2600 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2601 {0}{\@gls@entry@field{##1}{prevcount}}%
2602 }%

```

Make the unset and reset functions also increment or reset the entry counter.

```

2603 \renewcommand*\@glsunset}[1]{%
2604   \@glsunset{##1}%
2605   \@gls@increment@currcount{##1}%
2606 }%
2607 \renewcommand*\@glslocalunset}[1]{%
2608   \@glslocalunset{##1}%
2609   \@gls@local@increment@currcount{##1}%
2610 }%
2611 \renewcommand*\@glsreset}[1]{%
2612   \@glsreset{##1}%
2613   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2614 }%
2615 \renewcommand*\@glslocalreset}[1]{%
2616   \@glslocalreset{##1}%
2617   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2618 }%

```

Alter behaviour of \cgl's. (Only global unset is used if previous count was one as it doesn't make sense to have a local unset here given that the previous count was global.)

```

2619 \def\@cgl's@##1##2[##3]{%
2620   \ifnum\glsentryprevcount{##2}=1\relax
2621     \cgl'sformat{##2}{##3}%
2622     \glsunset{##2}%
2623   \else
2624     \@gls@{##1}{##2}[##3]%
2625   \fi
2626 }%

```

Similarly for the analogous commands. No case change plural:

```

2627 \def\@cgl'spl@##1##2[##3]{%
2628   \ifnum\glsentryprevcount{##2}=1\relax
2629     \cgl'splformat{##2}{##3}%
2630     \glsunset{##2}%
2631   \else
2632     \@glspl@{##1}{##2}[##3]%
2633   \fi
2634 }%

```

First letter uppercase singular:

```

2635 \def\@cGls@##1##2[##3]{%
2636   \ifnum\glsentryprevcount{##2}=1\relax
2637     \cGlsformat{##2}{##3}%
2638     \glsunset{##2}%
2639   \else
2640     \@Gls@{##1}{##2}[##3]%
2641   \fi

```

2642 }%

First letter uppercase plural:

```
2643 \def\cGlsp1@##1##2[##3]{%
2644 \ifnum\glentryprevcount{##2}=1\relax
2645 \cGlsp1format{##2}{##3}%
2646 \glunset{##2}%
2647 \else
2648 \cGlsp1@{##1}{##2}[##3]%
2649 \fi
2650 }%
```

Write information to aux file at the end of the document

```
2651 \AtEndDocument{\@gls@write@entrycounts}%
```

Fetch previous count information from aux file. (No check here to determine if the entry is still defined.)

```
2652 \renewcommand*{\@gls@entry@count}[2]{%
2653 \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
2654 }%
```

\glsenableentrycount may only be used once and only in the preamble.

```
2655 \let\glsenableentrycount\relax
2656 }
2657 \@onlypreamble\glsenableentrycount
```

ement@currcount

```
2658 \newcommand*{\@gls@increment@currcount}[1]{%
2659 \csxdef{glo@glsdetoklabel{##1}@currcount}{%
2660 \number\numexpr\glentrycurrcount{##1}+1}%
2661 }
```

ement@currcount

```
2662 \newcommand*{\@gls@local@increment@currcount}[1]{%
2663 \csedef{glo@glsdetoklabel{##1}@currcount}{%
2664 \number\numexpr\glentrycurrcount{##1}+1}%
2665 }
```

ite@entrycounts

Write the entry counts to the aux file. Use \immediate since this occurs right at the end of the document. Only write information for entries that have been used. (Some users have a file containing vast numbers of entries, many of which may not be used. There's no point writing information about the entries that haven't been used and it will only slow things down.)

```
2666 \newcommand*{\@gls@write@entrycounts}{%
2667 \immediate\write\@auxout
2668 {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2669 \forallglsentries{\@glsentry}{%
2670 \ifglsused{\@glsentry}%
2671 {\immediate\write\@auxout
2672 {\string\@gls@entry@count{\@glsentry}{\glentrycurrcount{\@glsentry}}}}%
2673 }
```

```
2674 }%
2675 }
```

`\gls@entry@count` Default behaviour is to ignore arguments. Activated by `\glsenableentrycount`.

```
2676 \newcommand*{\@gls@entry@count}[2]{}

\cgl
```

`\cgl` Define command that works like `\gls` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\gls` but issues a warning.)

```
2677 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

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

```
2678 \newcommand*{\@cgl}[2] [] {%
2679   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2} []}%
2680 }
```

`\@cgl@` Read in the final optional argument. This defaults to same behaviour as `\gls` but issues a warning.

```
2681 \def\@cgl@#1#2[#3] {%
2682   \GlossariesWarning{\string\cgl\space is defaulting to
2683     \string\gls\space since you haven't enabled entry counting}%
2684   \@gls@{#1}{#2}[#3]%
2685 }
```

`\cglformat` Format used by `\cgl` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2686 \newcommand*{\cglformat}[2] {%
2687   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2688 }
```

`\cGl` Define command that works like `\Gls` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\Gls` but issues a warning.)

```
2689 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}
```

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

```
2690 \newcommand*{\@cGl}[2] [] {%
2691   \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2} []}%
2692 }
```

`\@cGl@` Read in the final optional argument. This defaults to same behaviour as `\Gls` but issues a warning.

```
2693 \def\@cGl@#1#2[#3] {%
2694   \GlossariesWarning{\string\cGl\space is defaulting to
2695     \string\Gls\space since you haven't enabled entry counting}%
2696   \@Gls@{#1}{#2}[#3]%
2697 }
```

`\cGlsformat` Format used by `\cGls` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2698 \newcommand*{\cGlsformat}[2]{%
2699   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2700 }
```

`\cglsp1` Define command that works like `\glsp1` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\glsp1` but issues a warning.)

```
2701 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}
```

`\@cglsp1` Defined the un-starred form. Need to determine if there is a final optional argument

```
2702 \newcommand*{\@cglsp1}[2][ ]{%
2703   \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[ ]}]%
2704 }
```

`\@cglsp1@` Read in the final optional argument. This defaults to same behaviour as `\glsp1` but issues a warning.

```
2705 \def\@cglsp1@#1#2[#3]{%
2706   \GlossariesWarning{\string\cglsp1\space is defaulting to
2707     \string\glsp1\space since you haven't enabled entry counting}%
2708   \@glsp1@{#1}{#2}[#3]%
2709 }
```

`\cglsp1format` Format used by `\cglsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2710 \newcommand*{\cglsp1format}[2]{%
2711   \ifglshaslong{#1}{\glsp1longpl{#1}}{\glsp1firstplural{#1}}#2%
2712 }
```

`\cGlspl` Define command that works like `\Glspl` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\Glspl` but issues a warning.)

```
2713 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

`\@cglsp1` Defined the un-starred form. Need to determine if there is a final optional argument

```
2714 \newcommand*{\@cGlspl}[2][ ]{%
2715   \new@ifnextchar[{\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[ ]}]%
2716 }
```

`\@cGlspl@` Read in the final optional argument. This defaults to same behaviour as `\Glspl` but issues a warning.

```
2717 \def\@cGlspl@#1#2[#3]{%
2718   \GlossariesWarning{\string\cGlspl\space is defaulting to
2719     \string\Glspl\space since you haven't enabled entry counting}%
2720   \@Glspl@{#1}{#2}[#3]%
2721 }
```

`\cGlsplformat` Format used by `\cGlspl` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2722 \newcommand*{\cGlsplformat}[2]{%
2723   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2724 }
```

1.10 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`

```
2725 \newcommand*{\loadglsentries}[2] [\@gls@default]{%
2726   \let\@gls@default\glsdefaulttype
2727   \def\glsdefaulttype{#1}\input{#2}%
2728   \let\glsdefaulttype\@gls@default
2729 }
```

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

```
2730 \@onlypreamble{\loadglsentries}
```

1.11 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`

```
2731 \newcommand*{\glstextformat}[1]{#1}
```

`\glsentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glsentryfmt`. This doesn't take any arguments. The required information is set by commands like `\gls`. To

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

ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
2732 \newcommand*{\glsentryfmt}{%
2733   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2734 }
```

Format that provides backwards compatibility:

```
2735 \newcommand*{\@gls@default@entryfmt}[2]{%
2736   \ifdefempty\glscustomtext
2737   {%
2738     \glsifplural
2739     {%
```

Plural form

```
2740     \gls caps case
2741     {%
```

Don't adjust case

```
2742     \ifglsused\glslabel
2743     {%
```

Subsequent use

```
2744         #2{\glsentryplural{\glslabel}}%
2745         {\glsentrydescplural{\glslabel}}%
2746         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2747     }%
2748     {%
```

First use

```
2749         #1{\glsentryfirstplural{\glslabel}}%
2750         {\glsentrydescplural{\glslabel}}%
2751         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2752     }%
2753 }%
2754 {%
```

Make first letter upper case

```
2755     \ifglsused\glslabel
2756     {%
```

Subsequent use. (Expansion was used in version 3.07 and below in case the name wasn't the first thing to be displayed, but now the user can sort out the upper casing in `\defglsentryfmt`, which avoids the issues caused by fragile commands.)

```
2757     \ifbool{glscompatible-3.07}%
2758     {%
2759     \protected@edef\@gls@etext{%
2760       #2{\glsentryplural{\glslabel}}%
2761       {\glsentrydescplural{\glslabel}}%
2762       {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2763     \xmakefirstuc\@gls@etext
2764     }%
```

```

2765     {%
2766         #2{\Glsentryplural{\glslabel}}%
2767         {\Glsentrydescplural{\glslabel}}%
2768         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2769     }%
2770 }%
2771 {%

```

First use

```

2772     \ifbool{glscompatible-3.07}%
2773     {%
2774         \protected@edef\@glo@etext{%
2775             #1{\Glsentryfirstplural{\glslabel}}%
2776             {\Glsentrydescplural{\glslabel}}%
2777             {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2778         \xmakefirstuc\@glo@etext
2779     }%
2780     {%
2781         #1{\Glsentryfirstplural{\glslabel}}%
2782         {\Glsentrydescplural{\glslabel}}%
2783         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2784     }%
2785 }%
2786 }%
2787 {%

```

Make all upper case

```

2788     \ifglsused\glslabel
2789     {%

```

Subsequent use

```

2790         \mfirstucMakeUppercase{#2{\Glsentryplural{\glslabel}}%
2791         {\Glsentrydescplural{\glslabel}}%
2792         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2793     }%
2794     {%

```

First use

```

2795         \mfirstucMakeUppercase{#1{\Glsentryfirstplural{\glslabel}}%
2796         {\Glsentrydescplural{\glslabel}}%
2797         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2798     }%
2799 }%
2800 }%
2801 {%

```

Singular form

```

2802     \glscaps case
2803     {%

```

Don't adjust case

```

2804     \ifglsused\glslabel

```

```

2805      {%
Subsequent use
2806      #2{\glsentrytext{\glslabel}}%
2807      {\glsentrydesc{\glslabel}}%
2808      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2809      }%
2810      {%

First use
2811      #1{\glsentryfirst{\glslabel}}%
2812      {\glsentrydesc{\glslabel}}%
2813      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2814      }%
2815      }%
2816      {%

Make first letter upper case
2817      \ifglsused\glslabel
2818      {%

Subsequent use
2819      \ifbool{glscompatible-3.07}%
2820      {%
2821      \protected@edef\@glo@etext{%
2822      #2{\Glsentrytext{\glslabel}}%
2823      {\glsentrydesc{\glslabel}}%
2824      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2825      \xmakefirstuc\@glo@etext
2826      }%
2827      {%
2828      #2{\Glsentrytext{\glslabel}}%
2829      {\glsentrydesc{\glslabel}}%
2830      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2831      }%
2832      }%
2833      {%

First use
2834      \ifbool{glscompatible-3.07}%
2835      {%
2836      \protected@edef\@glo@etext{%
2837      #1{\glsentryfirst{\glslabel}}%
2838      {\glsentrydesc{\glslabel}}%
2839      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2840      \xmakefirstuc\@glo@etext
2841      }%
2842      {%
2843      #1{\Glsentryfirst{\glslabel}}%
2844      {\glsentrydesc{\glslabel}}%
2845      {\glsentrysymbol{\glslabel}}{\glsinsert}%

```



```

2846     }%
2847     }%
2848     }%
2849     {%

    Make all upper case
2850     \ifglsused\glslabel
2851     {%

    Subsequent use
2852     \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2853     {\glsentrydesc{\glslabel}}%
2854     {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2855     }%
2856     {%

    First use
2857     \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2858     {\glsentrydesc{\glslabel}}%
2859     {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2860     }%
2861     }%
2862     }%
2863     }%
2864     {%

    Custom text provided in \glsdisp
2865     \ifglsused{\glslabel}%
2866     {%

    Subsequent use
2867     #2{\glscustomtext}%
2868     {\glsentrydesc{\glslabel}}%
2869     {\glsentrysymbol{\glslabel}}{}%
2870     }%
2871     {%

    First use
2872     #1{\glscustomtext}%
2873     {\glsentrydesc{\glslabel}}%
2874     {\glsentrysymbol{\glslabel}}{}%
2875     }%
2876     }%
2877 }

```

`\glsentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

2878 \newcommand*{\glsentryfmt}{%
2879 \ifdefempty\glscustomtext
2880 {%
2881 \glsifplural
2882 {%

```

Plural form

2883 \glscapscase
2884 {%

Don't adjust case

2885 \ifglused\glslabel
2886 {%

Subsequent use

2887 \glstentryplural{\glslabel}\glsinsert
2888 }%
2889 {%

First use

2890 \glstentryfirstplural{\glslabel}\glsinsert
2891 }%
2892 }%
2893 {%

Make first letter upper case

2894 \ifglused\glslabel
2895 {%

Subsequent use.

2896 \Glstentryplural{\glslabel}\glsinsert
2897 }%
2898 {%

First use

2899 \Glstentryfirstplural{\glslabel}\glsinsert
2900 }%
2901 }%
2902 {%

Make all upper case

2903 \ifglused\glslabel
2904 {%

Subsequent use

2905 \mfirstucMakeUppercase
2906 {\glstentryplural{\glslabel}\glsinsert}%
2907 }%
2908 {%

First use

2909 \mfirstucMakeUppercase
2910 {\glstentryfirstplural{\glslabel}\glsinsert}%
2911 }%
2912 }%
2913 }%
2914 {%

Singular form

2915 `\glscapscase`
2916 `{%`

Don't adjust case

2917 `\ifglused\glslabel`
2918 `{%`

Subsequent use

2919 `\glentrytext{\glslabel}\glsinsert`
2920 `}%`
2921 `{%`

First use

2922 `\glentryfirst{\glslabel}\glsinsert`
2923 `}%`
2924 `}%`
2925 `{%`

Make first letter upper case

2926 `\ifglused\glslabel`
2927 `{%`

Subsequent use

2928 `\Glsentrytext{\glslabel}\glsinsert`
2929 `}%`
2930 `{%`

First use

2931 `\Glsentryfirst{\glslabel}\glsinsert`
2932 `}%`
2933 `}%`
2934 `{%`

Make all upper case

2935 `\ifglused\glslabel`
2936 `{%`

Subsequent use

2937 `\mfirstucMakeUppercase{\glentrytext{\glslabel}\glsinsert}%`
2938 `}%`
2939 `{%`

First use

2940 `\mfirstucMakeUppercase{\glentryfirst{\glslabel}\glsinsert}%`
2941 `}%`
2942 `}%`
2943 `}%`
2944 `}%`
2945 `{%`

Custom text provided in `\glsdisp`. (The insert is most likely to be empty at this point.)

2946 `\glscustomtext\glsinsert`

```
2947 }%
2948 }
```

`\glsgenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```
2949 \newcommand*{\glsgenacfmt}{%
2950   \ifdefempty\glscustomtext
2951   {%
2952     \ifglused\glslabel
2953     {%
```

Subsequent use:

```
2954     \glsifplural
2955     {%
```

Subsequent plural form:

```
2956     \glscapscase
2957     {%
```

Subsequent plural form, don't adjust case:

```
2958     \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2959     }%
2960     {%
```

Subsequent plural form, make first letter upper case:

```
2961     \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2962     }%
2963     {%
```

Subsequent plural form, all caps:

```
2964     \mfirstucMakeUppercase
2965     {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2966     }%
2967     }%
2968     {%
```

Subsequent singular form

```
2969     \glscapscase
2970     {%
```

Subsequent singular form, don't adjust case:

```
2971     \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2972     }%
2973     {%
```

Subsequent singular form, make first letter upper case:

```
2974     \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2975     }%
2976     {%
```

Subsequent singular form, all caps:

```
2977     \mfirstucMakeUppercase
2978     {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
```

2979 }%
2980 }%
2981 }%
2982 {%

First use:

2983 \glsifplural
2984 {%

First use plural form:

2985 \glscapscase
2986 {%

First use plural form, don't adjust case:

2987 \genplacrformat{\glslabel}{\glsinsert}%
2988 }%
2989 {%

First use plural form, make first letter upper case:

2990 \Genplacrformat{\glslabel}{\glsinsert}%
2991 }%
2992 {%

First use plural form, all caps:

2993 \mfirstucMakeUppercase
2994 {\genplacrformat{\glslabel}{\glsinsert}}%
2995 }%
2996 }%
2997 {%

First use singular form

2998 \glscapscase
2999 {%

First use singular form, don't adjust case:

3000 \genacrformat{\glslabel}{\glsinsert}%
3001 }%
3002 {%

First use singular form, make first letter upper case:

3003 \Genacrformat{\glslabel}{\glsinsert}%
3004 }%
3005 {%

First use singular form, all caps:

3006 \mfirstucMakeUppercase
3007 {\genacrformat{\glslabel}{\glsinsert}}%
3008 }%
3009 }%
3010 }%
3011 }%
3012 {%

User supplied text.

```
3013 \glscustomtext
3014 }%
3015 }
```

genacrfullformat `\genacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (singular).

```
3016 \newcommand*{\genacrfullformat}[2]{%
3017 \glentrylong{#1}#2\space
3018 (\protect\firstacronymfont{\glentryshort{#1}})%
3019 }
```

Genacrfullformat `\Genacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3020 \newcommand*{\Genacrfullformat}[2]{%
3021 \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3022 \xmakefirstuc\gls@text
3023 }
```

nplacrfullformat `\genplacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (plural).

```
3024 \newcommand*{\genplacrfullformat}[2]{%
3025 \glentrylongpl{#1}#2\space
3026 (\protect\firstacronymfont{\glentryshortpl{#1}})%
3027 }
```

Genplacrfullformat `\Genplacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3028 \newcommand*{\Genplacrfullformat}[2]{%
3029 \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3030 \xmakefirstuc\gls@text
3031 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
3032 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3033 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3034 \newcommand*{\defglsdisplay}[2] [\glsdefaulttype]{%
3035 \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3036 Use \string\defglsentryfmt\space instead}%
3037 \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3038 \edef\@gls@doentrydef{%
3039 \noexpand\defglsentryfmt [#1]{%
3040 \noexpand\ifcsdef{gls@#1@displayfirst}%
3041 {%
3042 \noexpand\@@gls@default@entryfmt
3043 {\noexpand\csuse{gls@#1@displayfirst}}}%
3044 {\noexpand\csuse{gls@#1@display}}}%
3045 }%
3046 {%
3047 \noexpand\@@gls@default@entryfmt
3048 {\noexpand\glsdisplayfirst}%
3049 {\noexpand\csuse{gls@#1@display}}}%
3050 }%
3051 }%
3052 }%
3053 \@gls@doentrydef
3054 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3055 \newcommand*{\defglsdisplayfirst}[2] [\glsdefaulttype]{%
3056 \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3057 Use \string\defglsentryfmt\space instead}%
3058 \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3059 \edef\@gls@doentrydef{%
3060 \noexpand\defglsentryfmt [#1]{%
3061 \noexpand\ifcsdef{gls@#1@display}%
3062 {%
3063 \noexpand\@@gls@default@entryfmt
3064 {\noexpand\csuse{gls@#1@displayfirst}}}%
3065 {\noexpand\csuse{gls@#1@display}}}%
3066 }%
3067 {%
3068 \noexpand\@@gls@default@entryfmt
3069 {\noexpand\csuse{gls@#1@displayfirst}}}%
3070 {\noexpand\glsdisplay}%
3071 }%
3072 }%
3073 }%
3074 \@gls@doentrydef
3075 }
```

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` and `\glsdisp`, 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 `\defentryfmt`). It goes against the \LaTeX 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.

```
3076 \define@key{glslink}{counter}{%
3077   \ifcsundef{c@#1}%
3078   {%
3079     \PackageError{glossaries}%
3080     {There is no counter called ‘#1’}%
3081     {%
3082       The counter key should have the name of a valid counter
3083       as its value%
3084     }%
3085   }%
3086   {%
3087     \def\@gls@counter{#1}%
3088   }%
3089 }
```

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.

```
3090 \define@key{glslink}{format}{%
3091   \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.

```
3092 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3093 \ifdef{\hyperlink}{\KV@glslink@hypertrue}{\KV@glslink@hyperfalse}
```

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.

```
3094 \define@boolkey{glslink}{local}[true]{}
```

The original `\glsifhyper` command isn't particularly useful as it makes more sense to check the actual hyperlink setting rather than testing whether the starred or unstarred version has been used. Therefore, as from version 4.08, `\glsifhyper` is deprecated in favour of

`\glsifhyperon`. In case there is a particular need to know whether the starred or unstarred version was used, provide a new command that determines whether the *-version, +-version or unmodified version was used.

```
\glslinkvar{<unmodified case>}{<star case>}{<plus case>}
```

`\glslinkvar` Initialise to unmodified case.

```
3095 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3096 \newcommand*{\glsifhyper}[2]{%
3097 \glslinkvar{#1}{#2}{#1}%
3098 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3099 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3100 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
3101 \newcommand*{\@gls@hyp@opt}[1]{%
3102 \let\glslinkvar\@firstofthree
3103 \let\@gls@hyp@opt@cs#1\relax
3104 \@ifstar{\s@gls@hyp@opt}%
3105 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3106 }
```

`\s@gls@hyp@opt` Starred version

```
3107 \newcommand*{\s@gls@hyp@opt}[1] []{%
3108 \let\glslinkvar\@secondofthree
3109 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
3110 \newcommand*{\p@gls@hyp@opt}[1] []{%
3111 \let\glslinkvar\@thirdofthree
3112 \@gls@hyp@opt@cs[hyper=true,#1]}
```

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 which version is being used:

`\glslink`

```
3113 \newrobustcmd*{\glslink}{%
3114 \@gls@hyp@opt\@gls@@link
3115 }
```

`\@gls@@link` 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.

```
3116 \newcommand*{\@gls@@link}[3] []{%
3117 \glsdoifexistsordo{#2}%
3118 {%
3119 \let\do@gls@link@checkfirsthyper\relax
3120 \@gls@link[#1]{#2}{#3}%
3121 }%
```

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

```
3122 \glstextformat{#3}%
3123 }%

3124 \glspostlinkhook
3125 }
```

`glspostlinkhook`

```
3126 \newcommand*{\glspostlinkhook}{}
3127 % \end{macrocode}
3128 %\end{macro}
3129 %
3130 %
3131 %\begin{macro}{\@gls@link@checkfirsthyper}
3132 % Check for first use and switch off \gloskey[glslink]{hyper} key
3133 % if hyperlink not wanted. (Should be off if first use and
3134 % hyper=false is on or if first use and both the entry is in an acronym
3135 % list and the acrfootnote setting is on.)
3136 % This assumes the glossary type is stored in \cs{glstype} and the
3137 % label is stored in \cs{glslabel}.
3138 %\changes{4.08}{2014-07-30}{new}
3139 % \begin{macrocode}
3140 \newcommand*{\@gls@link@checkfirsthyper}{%
3141 \ifglsused{glslabel}%
3142 {%
3143 }%
3144 {%
3145 \gls@checkisacronymlist\glstype
3146 \ifglshyperfirst
3147 \if@glsisacronymlist
3148 \ifglsacrfootnote
3149 \KV@glslink@hyperfalse
3150 \fi
```

```

3151     \fi
3152     \else
3153         \KV@glslink@hyperfalse
3154     \fi
3155 }%

```

Allow user to hook into this

```

3156 \glslinkcheckfirsthyperhook
3157 }

```

linkfirsthyperhook Allow used to hook into the \@gls@link@checkfirsthyper macro

```

3158 \newcommand*{\glslinkcheckfirsthyperhook}{}

```

linkpostsetkeys

```

3159 \newcommand*{\glslinkpostsetkeys}{}

```

\glsifhyperon Check the value of the hyper key:

```

3160 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

```

ablehyperinlist Disable hyperlink if in the “nohyper” list.

```

3161 \newcommand*\do@glsglisablehyperinlist{}%
3162 \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3163     {\KV@glslink@hyperfalse}{}%
3164 }

```

lt@glslink@opts Hook to set default options for \@gls@link.

```

3165 \newcommand*{\@gls@setdefault@glslink@opts}{}

```

\@gls@link

```

3166 \def\@gls@link[#1]#2#3{%

```

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

```

3167     \leavevmode
3168     \edef\glslabel{\glsdetoklabel{#2}}%

```

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

```

3169     \def\@gls@link@opts{#1}%
3170     \let\@gls@link@label\glslabel

```

```

3171     \def\@glsnumberformat{glsnumberformat}%
3172     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%

```

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

```

3173     \edef\glstype{\csname glo@\glslabel @type\endcsname}%

```

Save original setting

```

3174     \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

```

Set defaults:

```

3175     \@gls@setdefault@glslink@opts

```

Switch off hyper setting if the glossary type has been identified in nohyperlist.

```
3176 \do@gl:disablehyperinlist
```

Macros must set this before calling `\@gls@link`. The commands that check the first use flag should set this to `\@gls@link@checkfirsthyper` otherwise it should be set to `\relax`.

```
3177 \do@gls@link@checkfirsthyper
```

```
3178 \setkeys{glslink}{#1}%
```

Add a hook for the user to customise things after the keys have been set.

```
3179 \glslinkpostsetkeys
```

Store the entry's counter in `\theglsentrycounter`

```
3180 \@gls@saveentrycounter
```

Define sort key if necessary:

```
3181 \@gls@setsort{\glslabel}%
```

(De-tok'ing done by `\@do@wrglossary`)

```
3182 \@do@wrglossary{#2}%
```

```
3183 \ifKV@glslink@hyper
```

```
3184 \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3185 \else
```

```
3186 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3187 \fi
```

Restore original setting

```
3188 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
```

```
3189 }
```

`\glolinkprefix`

```
3190 \newcommand*{\glolinkprefix}{glo:}
```

`glsentrycounter` Set default value of entry counter

```
3191 \def\glsentrycounter{\glscounter}%
```

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

```
3192 \newcommand*{\@gls@saveentrycounter}{%
```

```
3193 \def\@gls@Hcounter}{}%
```

Are we using equation counter?

```
3194 \ifthenelse{\equal{\@gls@counter}{equation}}{%
```

```
3195 {
```

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

```
3196 \ifcsundef{xatlevel@}%
```

```
3197 {%
```

```
3198 \edef\theglsentrycounter{\expandafter\noexpand
```

```
3199 \csname the\@gls@counter\endcsname}%
```

```
3200 }%
```

```

3201   {%
3202     \ifx\xatlevel@\@empty
3203       \edef\theglentrycounter{\expandafter\noexpand
3204         \csname the\@gls@counter\endcsname}%
3205     \else
3206       \savecounters@
3207       \advance\c@equation by 1\relax
3208       \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3209     \ifcsundef{theH\@gls@counter}%
3210     {%
3211       \def\@gls@Hcounter{\theglentrycounter}%
3212     }%
3213     {%
3214       \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3215     }%
3216     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3217     \restorecounters@
3218   \fi
3219 }%
3220 }%
3221 {%

```

Not using equation counter so no special measures:

```

3222   \edef\theglentrycounter{\expandafter\noexpand
3223     \csname the\@gls@counter\endcsname}%
3224 }%

```

Check if hyperref version of this counter

```

3225 \ifx\@gls@Hcounter\@empty
3226   \ifcsundef{theH\@gls@counter}%
3227   {%
3228     \def\theHglentrycounter{\theglentrycounter}%
3229   }%
3230   {%
3231     \protected@edef\theHglentrycounter{\expandafter\noexpand
3232       \csname theH\@gls@counter\endcsname}%
3233   }%
3234 \fi
3235 }

```

t@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.

```

3236 \def\@set@glo@numformat#1#2#3#4{%
3237   \expandafter\@glo@check@mkidxrangear#3\@nil
3238   \protected@edef#1{%

```

```

3239 \glo@prefix setentrycounter[#4]{#2}%
3240 \expandafter\string\curname\glo@suffix\endcsname
3241 }%
3242 \gls@checkmkidxchars#1%
3243 }

```

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.

```

3244 \def\glo@check@mkidxrangechar#1#2\@nil{%
3245 \if#1(\relax
3246 \def\glo@prefix{(%}
3247 \if\relax#2\relax
3248 \def\glo@suffix{glsnumberformat}%
3249 \else
3250 \def\glo@suffix{#2}%
3251 \fi
3252 \else
3253 \if#1)\relax
3254 \def\glo@prefix{)%}
3255 \if\relax#2\relax
3256 \def\glo@suffix{glsnumberformat}%
3257 \else
3258 \def\glo@suffix{#2}%
3259 \fi
3260 \else
3261 \def\glo@prefix{}\def\glo@suffix{#1#2}%
3262 \fi
3263 \fi}

```

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

```

3264 \newcommand*\@gls@escbsdq[1]{%
3265 \def\gls@checkedmkidx{}%
3266 \let\gls@xdystring=#1\relax
3267 \@onelevel@sanitize\gls@xdystring
3268 \edef\do@gls@xdycheckbackslash{%
3269 \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3270 \@backslashchar\@backslashchar\noexpand\null}%
3271 \do@gls@xdycheckbackslash
3272 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3273 \def\gls@checkedmkidx{}%
3274 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3275 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3276 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3277 {%
3278 \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\ \expandonce\@gls@tmp}%

```

```

3279 \@onelevel@sanitize\@gls@sanitized@tmp
3280 \edef\gls@dostsubst{%
3281 \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3282 {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3283 }%
3284 \gls@dostsubst
3285 }%

```

Assign to required control sequence

```

3286 \let#1=\gls@xdystring
3287 }

```

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

checkmkidxchars

```

3288 \newcommand{\@gls@checkmkidxchars}[1]{%
3289 \ifglsexindy
3290 \@gls@escbsdq{#1}%
3291 \else
3292 \def\@gls@checkedmkidx{%
3293 \expandafter\@gls@checkquote#1\@nil""\null
3294 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3295 \def\@gls@checkedmkidx{%
3296 \expandafter\@gls@checkescquote#1\@nil""\null
3297 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3298 \def\@gls@checkedmkidx{%
3299 \expandafter\@gls@checkescactual#1\@nil\?\?\null
3300 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3301 \def\@gls@checkedmkidx{%
3302 \expandafter\@gls@checkactual#1\@nil??\null
3303 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3304 \def\@gls@checkedmkidx{%
3305 \expandafter\@gls@checkbar#1\@nil||\null
3306 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3307 \def\@gls@checkedmkidx{%
3308 \expandafter\@gls@checkescbar#1\@nil\\|\null
3309 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3310 \def\@gls@checkedmkidx{%
3311 \expandafter\@gls@checklevel#1\@nil!!\null
3312 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3313 \fi
3314 }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```

3315 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3316 \newtoks\@gls@tmpb

```

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

```
3317 \def@gls@checkquote#1"#2"#3\null{%
3318   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3319   \toks@={#1}%
3320   \ifx\null#2\null
3321   \ifx\null#3\null
3322   \edef@gls@checkedmkidx{\the@gls@tmpb\the\toks@}%
3323   \def\@gls@checkquote{\relax}%
3324   \else
3325   \edef@gls@checkedmkidx{\the@gls@tmpb\the\toks@
3326     \@gls@quotechar@gls@quotechar@gls@quotechar@gls@quotechar}%
3327   \def\@gls@checkquote{\@gls@checkquote#3\null}%
3328   \fi
3329   \else
3330   \edef@gls@checkedmkidx{\the@gls@tmpb\the\toks@
3331     \@gls@quotechar@gls@quotechar}%
3332   \ifx\null#3\null
3333     \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3334   \else
3335     \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3336   \fi
3337   \fi
3338   \@gls@checkquote
3339 }
```

s@checkescquote Do the same for \":

```
3340 \def@gls@checkescquote#1\"#2\"#3\null{%
3341   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3342   \toks@={#1}%
3343   \ifx\null#2\null
3344   \ifx\null#3\null
3345   \edef@gls@checkedmkidx{\the@gls@tmpb\the\toks@}%
3346   \def\@gls@checkescquote{\relax}%
3347   \else
3348   \edef@gls@checkedmkidx{\the@gls@tmpb\the\toks@
3349     \@gls@quotechar\string\"@gls@quotechar
3350     \@gls@quotechar\string\"@gls@quotechar}%
3351   \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
3352   \fi
3353   \else
3354   \edef@gls@checkedmkidx{\the@gls@tmpb\the\toks@
3355     \@gls@quotechar\string\"@gls@quotechar}%
3356   \ifx\null#3\null
3357     \def\@gls@checkescquote{\@gls@checkescquote#2\""\null}%
3358   \else
3359     \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3360   \fi
3361   \fi
3362   \@gls@checkescquote
```


3363 }

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

```
3364 \def\@gls@checkescactual#1\?#2\?#3\null{%
3365 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3366 \toks@={#1}%
3367 \ifx\null#2\null
3368 \ifx\null#3\null
3369 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3370 \def\@gls@checkescactual{\relax}%
3371 \else
3372 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3373 \@gls@quotechar\string\\"\@gls@actualchar
3374 \@gls@quotechar\string\\"\@gls@actualchar}%
3375 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3376 \fi
3377 \else
3378 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3379 \@gls@quotechar\string\\"\@gls@actualchar}%
3380 \ifx\null#3\null
3381 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3382 \else
3383 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3384 \fi
3385 \fi
3386 \@gls@checkescactual
3387 }
```

gls@checkeschar Similarly for \|:

```
3388 \def\@gls@checkeschar#1\|#2\|#3\null{%
3389 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3390 \toks@={#1}%
3391 \ifx\null#2\null
3392 \ifx\null#3\null
3393 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3394 \def\@gls@checkeschar{\relax}%
3395 \else
3396 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3397 \@gls@quotechar\string\\"\@gls@encapchar
3398 \@gls@quotechar\string\\"\@gls@encapchar}%
3399 \def\@gls@checkeschar{\@gls@checkeschar#3\null}%
3400 \fi
3401 \else
3402 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3403 \@gls@quotechar\string\\"\@gls@encapchar}%
3404 \ifx\null#3\null
3405 \def\@gls@checkeschar{\@gls@checkeschar#2\|\|\null}%
3406 \else
3407 \def\@gls@checkeschar{\@gls@checkeschar#2\|#3\null}%

```

```

3408 \fi
3409 \fi
3410 \@gls@checkeschar
3411 }

```

s@checkesclevel Similarly for \!:

```

3412 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3413 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3414 \toks@={#1}%
3415 \ifx\null#2\null
3416 \ifx\null#3\null
3417 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3418 \def\@gls@checkesclevel{\relax}%
3419 \else
3420 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3421 \@gls@quotechar\string"\@gls@levelchar
3422 \@gls@quotechar\string"\@gls@levelchar}%
3423 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3424 \fi
3425 \else
3426 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3427 \@gls@quotechar\string"\@gls@levelchar}%
3428 \ifx\null#3\null
3429 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3430 \else
3431 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3432 \fi
3433 \fi
3434 \@gls@checkesclevel
3435 }

```

\@gls@checkbar and for |:

```

3436 \def\@gls@checkbar#1|#2|#3\null{%
3437 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3438 \toks@={#1}%
3439 \ifx\null#2\null
3440 \ifx\null#3\null
3441 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3442 \def\@gls@checkbar{\relax}%
3443 \else
3444 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3445 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3446 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3447 \fi
3448 \else
3449 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3450 \@gls@quotechar\@gls@encapchar}%
3451 \ifx\null#3\null
3452 \def\@gls@checkbar{\@gls@checkbar#2||\null}%

```

```

3453 \else
3454 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3455 \fi
3456 \fi
3457 \@gls@checkbar
3458 }

```

`@gls@checklevel` and for !:

```

3459 \def\@gls@checklevel#1!#2!#3\null{%
3460 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3461 \toks@={#1}%
3462 \ifx\null#2\null
3463 \ifx\null#3\null
3464 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3465 \def\@gls@checklevel{\relax}%
3466 \else
3467 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3468 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3469 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3470 \fi
3471 \else
3472 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3473 \@gls@quotechar\@gls@levelchar}%
3474 \ifx\null#3\null
3475 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3476 \else
3477 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3478 \fi
3479 \fi
3480 \@gls@checklevel
3481 }

```

`gls@checkactual` and for ?:

```

3482 \def\@gls@checkactual#1?#2?#3\null{%
3483 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3484 \toks@={#1}%
3485 \ifx\null#2\null
3486 \ifx\null#3\null
3487 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3488 \def\@gls@checkactual{\relax}%
3489 \else
3490 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3491 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3492 \def\@gls@checkactual{\@gls@checkactual#3\null}%
3493 \fi
3494 \else
3495 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3496 \@gls@quotechar\@gls@actualchar}%
3497 \ifx\null#3\null

```

```

3498     \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3499     \else
3500     \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3501     \fi
3502     \fi
3503     \@gls@checkactual
3504 }

```

s@xdycheckquote As before but for use with xindy

```

3505 \def\@gls@xdycheckquote#1"#2"#3\null{%
3506   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3507   \toks@={#1}%
3508   \ifx\null#2\null
3509     \ifx\null#3\null
3510       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3511       \def\@gls@xdycheckquote{\relax}%
3512     \else
3513       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3514         \string"\string"}%
3515       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3516     \fi
3517   \else
3518     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3519       \string"}%
3520     \ifx\null#3\null
3521       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3522     \else
3523       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3524     \fi
3525     \fi
3526     \@gls@xdycheckquote
3527 }

```

ycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

3528 \edef\def@gls@xdycheckbackslash{%
3529   \noexpand\def\noexpand@gls@xdycheckbackslash##1\@backslashchar
3530   ##2\@backslashchar##3\noexpand\null{%
3531     \noexpand\@gls@tmpb=\noexpand\expandafter
3532     {\noexpand@gls@checkedmkidx}%
3533     \noexpand\toks@={##1}%
3534     \noexpand\ifx\noexpand\null##2\noexpand\null
3535     \noexpand\ifx\noexpand\null##3\noexpand\null
3536     \noexpand\edef\noexpand@gls@checkedmkidx{%
3537       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3538     \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3539     \noexpand\else
3540     \noexpand\edef\noexpand@gls@checkedmkidx{%
3541       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3542       \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%

```

```

3543 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3544   \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3545 \noexpand\fi
3546 \noexpand\else
3547 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3548   \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3549   \@backslashchar\@backslashchar}%
3550 \noexpand\ifx\noexpand\null##3\noexpand\null
3551 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3552   \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3553   \@backslashchar\noexpand\null}%
3554 \noexpand\else
3555 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3556   \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3557   ##3\noexpand\null}%
3558 \noexpand\fi
3559 \noexpand\fi
3560 \noexpand\@gls@xdycheckbackslash
3561 }%
3562 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3563 \def@gls@xdycheckbackslash

```

lsdohypertarget

```

3564 \newlength@gls@tmplen
3565 \newcommand*{\glsdohypertarget}[2]{%
3566   \settoheight{\gls@tmplen}{#2}%
3567   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3568 }

```

\glsdohyperlink

```

3569 \newcommand*{\glsdohyperlink}[2]{\hyperlink{#1}{#2}}

```

lsdonohyperlink

```

3570 \newcommand*{\glsdonohyperlink}[2]{#2}

```

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

```

3571 \ifcsundef{hyperlink}%
3572 {%
3573   \let@glslink@glsdonohyperlink
3574 }%
3575 {%
3576   \let@glslink@glsdohyperlink
3577 }

```

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

```
3578 \ifcsundef{hypertarget}%
3579 {%
3580   \let\@glstarget\@secondoftwo
3581 }%
3582 {%
3583   \let\@glstarget\glsdohypertarget
3584 }
```

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

`\glsdisablehyper`

```
3585 \newcommand{\glsdisablehyper}{%
3586   \KV@glslink@hyperfalse
3587   \let\@glslink\glsdonohyperlink
3588   \let\@glstarget\@secondoftwo
3589 }
```

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

`\glsenablehyper`

```
3590 \newcommand{\glsenablehyper}{%
3591   \KV@glslink@hypertrue
3592   \let\@glslink\glsdohyperlink
3593   \let\@glstarget\glsdohypertarget
3594 }
```

Provide some convenience commands if not already defined:

```
3595 \providecommand{\@firstofthree}[3]{#1}
3596 \providecommand{\@secondofthree}[3]{#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 which version is being used:

`\gls`

```
3597 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```
3598 \newcommand*{\@gls}[2] [] {%
3599   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} []}]%
3600 }
```

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

```
3601 \def\@gls@#1#2[#3]{%
3602   \glsdoifexists{#2}%
3603   {%
3604     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3605     \let\glsifplural\@secondoftwo
3606     \let\glsapscase\@firstofthree
3607     \let\glscustomtext\@empty
3608     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3609   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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.

```
3610   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3611   \ifKV@glslink@local
3612     \glslocalunset{#2}%
3613   \else
3614     \glsunset{#2}%
3615   \fi
3616   }%
```

```
3617   \glspostlinkhook
3618 }
```

`\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`

```
3619 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3620 \newcommand*{\@Gls}[2] [] {%
3621   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}]%
3622 }
```

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

```
3623 \def\@Gls@#1#2[#3]{%
3624   \glsdoifexists{#2}%
3625   {%
3626     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3627     \let\glsifplural\@secondoftwo
3628     \let\glsapscase\@secondofthree
3629     \let\glscustomtext\@empty
3630     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3631   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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.

```
3632   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3633   \ifKV@glslink@local
3634     \glslocalunset{#2}%
3635   \else
3636     \glsunset{#2}%
3637   \fi
3638 }%
```

```
3639 \glspostlinkhook
3640 }
```

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

`\GLS`

```
3641 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
3642 \newcommand*{\@GLS}[2][ ]{%
3643   \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} [ ]}%
3644 }
```

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

```
3645 \def\@GLS@#1#2[#3]{%
3646   \glsdoifexists{#2}%
3647   {%
3648     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3649     \let\glsifplural\@secondoftwo
3650     \let\glsapscase\@thirdofthree
3651     \let\glscustomtext\@empty
3652     \def\glsinsert{#3}%
```


Determine what the link text should be (this is stored in \@glo@text). Note that \@gls@link sets \gls@type.

```
3653 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

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

```
3654 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3655 \ifKV@gls@link@local
```

```
3656 \glslocalunset{#2}%
```

```
3657 \else
```

```
3658 \glsunset{#2}%
```

```
3659 \fi
```

```
3660 }%
```

```
3661 \gls@postlinkhook
```

```
3662 }
```

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

\glspl

```
3663 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3664 \newcommand*{\@glspl}[2] [] {%
```

```
3665 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2} []}] {%
```

```
3666 }
```

\@glspl@ Read in the final optional argument:

```
3667 \def\@glspl@#1#2[#3] {%
```

```
3668 \glsdoifexists{#2}%
```

```
3669 {%
```

```
3670 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3671 \let\glsifplural\@firstoftwo
```

```
3672 \let\gls@scapscase\@firstofthree
```

```
3673 \let\gls@customtext\@empty
```

```
3674 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \gls@type.

```
3675 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

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

```
3676 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3677 \ifKV@glslink@local
3678 \glslocalunset{#2}%
3679 \else
3680 \glsunset{#2}%
3681 \fi
3682 }%

3683 \glspostlinkhook
3684 }
```

`\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`

```
3685 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3686 \newcommand*{\@Glspl}[2][ ]{%
3687 \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[ ]}%
3688 }
```

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

```
3689 \def\@Glspl@#1#2[#3]{%
3690 \glsdoifexists{#2}%
3691 {%
3692 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3693 \let\glsifplural\@firstoftwo
3694 \let\glsapscase\@secondofthree
3695 \let\glscustomtext\@empty
3696 \def\glsinsert{#3}%
```

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`. Note that `\@gls@link` sets `\glstype`.

```
3697 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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.

```
3698 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3699 \ifKV@glslink@local
3700 \glslocalunset{#2}%
3701 \else
3702 \glsunset{#2}%
3703 \fi
3704 }%
```

```
3705 \glspostlinkhook
3706 }
```

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

`\GLSp1`

```
3707 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3708 \newcommand*{\@GLSp1}[2] [] {%
3709   \new@ifnextchar [{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}]%
3710 }
```

`\@GLSp1` Read in the final optional argument:

```
3711 \def\@GLSp1@#1#2[#3] {%
3712   \glsdoifexists{#2}%
3713   {%
3714     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3715     \let\glsifplural\@firstoftwo
3716     \let\glsapscase\@thirdofthree
3717     \let\glscustomtext\@empty
3718     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3719   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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.

```
3720   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3721   \ifKV@glslink@local
3722     \glslocalunset{#2}%
3723   \else
3724     \glsunset{#2}%
3725   \fi
3726   }%
```

```
3727 \glspostlinkhook
3728 }
```

`\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:

```
3729 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}
```

Defined the un-starred form.

`\@glsdisp`

```
3730 \newcommand*{\@glsdisp}[3] [] {%
3731   \glsdoifexists{#2}{%

3732     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3733     \let\glsifplural\@secondoftwo
3734     \let\glscapscase\@firstofthree
3735     \def\glscustomtext{#3}%
3736     \def\glsinsert{}}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3737   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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.

```
3738   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3739   \ifKV@gls@link@local
3740     \glslocalunset{#2}%
3741   \else
3742     \glsunset{#2}%
3743   \fi
3744 }%
```

```
3745 \glspostlinkhook
3746 }
```

`checkfirsthyper` Instead of just setting `\do@gls@link@checkfirsthyper` to `\relax` in `\@gls@field@link`, set it to `\@gls@link@nocheckfirsthyper` in case some other action needs to take place.

```
3747 \newcommand*{\@gls@link@nocheckfirsthyper}{}
```

`@gls@field@link`

```
3748 \newcommand{\@gls@field@link}[3] {%
3749   \glsdoifexists{#2}%
3750   {%
3751     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3752     \@gls@link[#1]{#2}{#3}%
3753   }%

3754   \glspostlinkhook
3755 }
```

`\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`

```
3756 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

```
3757 \newcommand*{\@glstext}[2] [] {%
```

```
3758   \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3759 \def\@glstext@#1#2[#3] {%
```

```
3760   \@gls@field@link{#1}{#2}{\glstentrytext{#2}#3}%
```

```
3761 }
```

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

`\GLStext`

```
3762 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3763 \newcommand*{\@GLStext}[2] [] {%
```

```
3764   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3765 \def\@GLStext@#1#2[#3] {%
```

```
3766   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
```

```
3767 }
```

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

`\Glstext`

```
3768 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3769 \newcommand*{\@Glstext}[2] [] {%
```

```
3770   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3771 \def\@Glstext@#1#2[#3] {%
```

```
3772   \@gls@field@link{#1}{#2}{\Glstentrytext{#2}#3}%
```

```
3773 }
```

`\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`

```
3774 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3775 \newcommand*{\@glsfirst}[2] [] {%
```

```
3776   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3777 \def\@glsfirst@#1#2[#3]{%
3778   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3779 }
```

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

`\Glsfirst`

```
3780 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3781 \newcommand*{\@Glsfirst}[2] [] {%
3782   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3783 \def\@Glsfirst@#1#2[#3]{%
3784   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3785 }
```

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

`\GLSfirst`

```
3786 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3787 \newcommand*{\@GLSfirst}[2] [] {%
3788   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3789 \def\@GLSfirst@#1#2[#3]{%
3790   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3791 }
```

`\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`

```
3792 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3793 \newcommand*{\@glsplural}[2] [] {%
3794   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3795 \def\@glsplural@#1#2[#3]{%
3796   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
3797 }
```

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

`\Glsplural`

```
3798 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3799 \newcommand*{\@Glsplural}[2] [] {%
3800   \new@ifnextchar [{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3801 \def\@Glsplural@#1#2[#3] {%
3802   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3803 }
```

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

\Glsplural

```
3804 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3805 \newcommand*{\@Glsplural}[2] [] {%
3806   \new@ifnextchar [{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3807 \def\@Glsplural@#1#2[#3] {%
3808   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
3809 }
```

\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

```
3810 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3811 \newcommand*{\@glsfirstplural}[2] [] {%
3812   \new@ifnextchar [{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3813 \def\@glsfirstplural@#1#2[#3] {%
3814   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
3815 }
```

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

\Glsfirstplural

```
3816 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3817 \newcommand*{\@Glsfirstplural}[2] [] {%
3818   \new@ifnextchar [{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3819 \def\@Glsfirstplural@#1#2[#3] {%
3820   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3821 }
```

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

`\GLSfirstplural`

```
3822 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3823 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3824 \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3825 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3826 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
3827 }
```

`\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`

```
3828 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3829 \newcommand*{\@glsname}[2] [] {%
```

```
3830 \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3831 \def\@glsname@#1#2[#3] {%
```

```
3832 \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}}%
```

```
3833 }
```

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

`\Glsname`

```
3834 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3835 \newcommand*{\@Glsname}[2] [] {%
```

```
3836 \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3837 \def\@Glsname@#1#2[#3] {%
```

```
3838 \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}}%
```

```
3839 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
3840 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3841 \newcommand*{\@GLSname}[2] [] {%
```

```
3842 \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}]}
```


Read in the final optional argument:

```
3843 \def\@GLSname@#1#2[#3]{%
3844 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3845 }
```

`\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`

```
3846 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3847 \newcommand*{\@glsdesc}[2] [] {%
3848 \new@ifnextchar [{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
3849 \def\@glsdesc@#1#2[#3]{%
3850 \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3851 }
```

`\Glsdesc` behaves like `\glsdesc` except that the first letter is converted to uppercase.

`\Glsdesc`

```
3852 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3853 \newcommand*{\@Glsdesc}[2] [] {%
3854 \new@ifnextchar [{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
3855 \def\@Glsdesc@#1#2[#3]{%
3856 \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3857 }
```

`\GLSdesc` behaves like `\glsdesc` except that the link text is converted to uppercase.

`\GLSdesc`

```
3858 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3859 \newcommand*{\@GLSdesc}[2] [] {%
3860 \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
3861 \def\@GLSdesc@#1#2[#3]{%
3862 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3863 }
```

`\glsdescplural` behaves like `\gls` except it always uses the value given by the description-plural key and it doesn't mark the entry as used.

`\glsdescplural`

```
3864 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3865 \newcommand*{\@glsdescplural}[2] [] {%
3866   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3867 \def\@glsdescplural@#1#2[#3] {%
3868   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3869 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
3870 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3871 \newcommand*{\@GLSdescplural}[2] [] {%
3872   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3873 \def\@GLSdescplural@#1#2[#3] {%
3874   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3875 }
```

\GLSdescplural behaves like \glsdescplural except that the link text is converted to uppercase.

\GLSdescplural

```
3876 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3877 \newcommand*{\@GLSdescplural}[2] [] {%
3878   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3879 \def\@GLSdescplural@#1#2[#3] {%
3880   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3881 }
```

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

\glsymbol

```
3882 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
3883 \newcommand*{\@glsymbol}[2] [] {%
3884   \new@ifnextchar [{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3885 \def\@glsymbol@#1#2[#3] {%
3886   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
3887 }
```

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
3888 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
3889 \newcommand*{\@Glssymbol}[2] [] {%
```

```
3890 \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3891 \def\@Glssymbol@#1#2[#3] {%
```

```
3892 \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3893 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3894 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3895 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3896 \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3897 \def\@GLSsymbol@#1#2[#3] {%
```

```
3898 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}%
```

```
3899 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the symbol-plural key and it doesn't mark the entry as used.

`glssymbolplural`

```
3900 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

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

```
3901 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
3902 \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3903 \def\@glssymbolplural@#1#2[#3] {%
```

```
3904 \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}%
```

```
3905 }
```

`\Glssymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`Glssymbolplural`

```
3906 \newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
3907 \newcommand*{\@Glssymbolplural}[2] [] {%
```

```
3908 \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3909 \def\@GLssymbolplural@#1#2[#3]{%
3910 \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3911 }
```

`\GLSsymbolplural` behaves like `\glssymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
3912 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3913 \newcommand*{\@GLSsymbolplural}[2] [] {%
3914 \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3915 \def\@GLSsymbolplural@#1#2[#3]{%
3916 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentrysymbolplural{#2}#3}}%
3917 }
```

`\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`

```
3918 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3919 \newcommand*{\@glsuseri}[2] [] {%
3920 \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3921 \def\@glsuseri@#1#2[#3]{%
3922 \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3923 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
3924 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3925 \newcommand*{\@Glsuseri}[2] [] {%
3926 \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3927 \def\@Glsuseri@#1#2[#3]{%
3928 \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3929 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
3930 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3931 \newcommand*{\@GLSuseri}[2] [] {%
3932   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3933 \def\@GLSuseri@#1#2[#3] {%
3934   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
3935 }
```

`\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`

```
3936 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3937 \newcommand*{\@glsuserii}[2] [] {%
3938   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3939 \def\@glsuserii@#1#2[#3] {%
3940   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
3941 }
```

`\Glsuserii` behaves like `\glsuserii` except that the first letter is converted to uppercase.

`\Glsuserii`

```
3942 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3943 \newcommand*{\@Glsuserii}[2] [] {%
3944   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3945 \def\@Glsuserii@#1#2[#3] {%
3946   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
3947 }
```

`\GLSuserii` behaves like `\glsuserii` except that the link text is converted to uppercase.

`\GLSuserii`

```
3948 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3949 \newcommand*{\@GLSuserii}[2] [] {%
3950   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3951 \def\@GLSuserii@#1#2[#3] {%
3952   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
3953 }
```

`\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`

```
3954 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3955 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3956 \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
3957 \def\@glsuseriii@#1#2[#3] {%
```

```
3958 \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%
```

```
3959 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

`\Glsuseriii`

```
3960 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3961 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
3962 \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
3963 \def\@Glsuseriii@#1#2[#3] {%
```

```
3964 \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
3965 }
```

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
3966 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3967 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
3968 \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
3969 \def\@GLSuseriii@#1#2[#3] {%
```

```
3970 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
```

```
3971 }
```

`\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`

```
3972 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3973 \newcommand*{\@glsuseriv}[2] [] {%
```

```
3974 \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
3975 \def\@glsuseriv@#1#2[#3]{%
3976 \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3977 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3978 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3979 \newcommand*{\@Glsuseriv}[2][ ]{%
3980 \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3981 \def\@Glsuseriv@#1#2[#3]{%
3982 \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3983 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
3984 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
3985 \newcommand*{\@GLSuseriv}[2][ ]{%
3986 \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3987 \def\@GLSuseriv@#1#2[#3]{%
3988 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
3989 }
```

\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

```
3990 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3991 \newcommand*{\@glsuserv}[2][ ]{%
3992 \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3993 \def\@glsuserv@#1#2[#3]{%
3994 \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
3995 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3996 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3997 \newcommand*{\@Glsuserv}[2] [] {%
3998 \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
3999 \def\@Glsuserv@#1#2[#3] {%
4000 \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
4001 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4002 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
4003 \newcommand*{\@GLSuserv}[2] [] {%
4004 \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
4005 \def\@GLSuserv@#1#2[#3] {%
4006 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4007 }
```

\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

```
4008 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4009 \newcommand*{\@glsuservi}[2] [] {%
4010 \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
4011 \def\@glsuservi@#1#2[#3] {%
4012 \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4013 }
```

\GLsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\GLsuservi

```
4014 \newrobustcmd*{\GLsuservi}{\@gls@hyp@opt\@GLsuservi}
```

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

```
4015 \newcommand*{\@GLsuservi}[2] [] {%
4016 \new@ifnextchar [{\@GLsuservi@{#1}{#2}}{\@GLsuservi@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
4017 \def\@GLsuservi@#1#2[#3] {%
4018 \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
4019 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

`\GLSuservi`

```
4020 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4021 \newcommand*{\@GLSuservi}[2][\@GLSuservi]
```

```
4022 \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4023 \def\@GLSuservi@#1#2[#3]{%
```

```
4024 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
4025 }
```

Now deal with acronym related keys. First the short form:

`\acrshort`

```
4026 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4027 \newcommand*{\@ns@acrshort}[2][\@ns@acrshort]
```

```
4028 \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2}[]}]
```

```
4029 }
```

Read in the final optional argument:

```
4030 \def\@acrshort#1#2[#3]{%
```

```
4031 \glsdoifexists{#2}}%
```

```
4032 {%
```

```
4033 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4034 \let\glsifplural\@secondoftwo
```

```
4035 \let\gls@caps@case\@firstofthree
```

```
4036 \let\glsinsert\@empty
```

```
4037 \def\gls@customtext{%
```

```
4038 \acronymfont{\glsentryshort{#2}}#3%
```

```
4039 }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
4040 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}}%
```

```
4041 }%
```

```
4042 \gls@postlinkhook
```

```
4043 }
```

`\Acrshort`

```
4044 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}
```

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

```
4045 \newcommand*{\@ns@Acrshort}[2][\@ns@Acrshort]
```

```
4046 \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2}[]}]
```

```
4047 }
```

Read in the final optional argument:

```
4048 \def\@Acrshort#1#2[#3]{%
4049   \glsdoifexists{#2}%
4050   {%
4051     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4052     \def\glslabel{#2}%
4053     \let\glsifplural\@secondoftwo
4054     \let\gls caps case\@secondofthree
4055     \let\glsinsert\@empty
4056     \def\gls custom text{%
4057       \acronymfont{\Glsentryshort{#2}}#3%
4058     }%
```

Call \@gls@link Note that \@gls@link sets \gls type.

```
4059   \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4060   }%
4061   \gls post link hook
4062 }
```

\ACRshort

```
4063 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\ns@ACRshort}
```

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

```
4064 \newcommand*{\ns@ACRshort}[2][ ]{%
4065   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2}[]}%
4066 }
```

Read in the final optional argument:

```
4067 \def\@ACRshort#1#2[#3]{%
4068   \glsdoifexists{#2}%
4069   {%
4070     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4071     \def\glslabel{#2}%
4072     \let\glsifplural\@secondoftwo
4073     \let\gls caps case\@thirdofthree
4074     \let\glsinsert\@empty
4075     \def\gls custom text{%
4076       \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4077     }%
```

Call \@gls@link Note that \@gls@link sets \gls type.

```
4078   \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4079   }%
4080   \gls post link hook
4081 }
```

Short plural:

`\acrshortpl`

```
4082 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}
```

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

```
4083 \newcommand*{\ns@acrshortpl}[2] [] {%
```

```
4084   \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
```

```
4085 }
```

Read in the final optional argument:

```
4086 \def\@acrshortpl#1#2[#3] {%
```

```
4087   \glsdoifexists{#2}%
```

```
4088   {%
```

```
4089     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4090     \def\glslabel{#2}%
```

```
4091     \let\glsifplural\@firstoftwo
```

```
4092     \let\glsapscase\@firstofthree
```

```
4093     \let\glsinsert\@empty
```

```
4094     \def\glscustomtext{%
```

```
4095       \acronymfont{\glsentryshortpl{#2}}#3%
```

```
4096     }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glsstyle`.

```
4097   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
```

```
4098   }%
```

```
4099   \glspostlinkhook
```

```
4100 }
```

`\Acrshortpl`

```
4101 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

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

```
4102 \newcommand*{\ns@Acrshortpl}[2] [] {%
```

```
4103   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
```

```
4104 }
```

Read in the final optional argument:

```
4105 \def\@Acrshortpl#1#2[#3] {%
```

```
4106   \glsdoifexists{#2}%
```

```
4107   {%
```

```
4108     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4109     \def\glslabel{#2}%
```

```
4110     \let\glsifplural\@firstoftwo
```

```
4111     \let\glsapscase\@secondofthree
```

```
4112     \let\glsinsert\@empty
```

```

4113 \def\glscustomtext{%
4114 \acronymfont{\Glsentryshortpl{#2}}#3%
4115 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4116 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4117 }%

4118 \glspostlinkhook
4119 }

```

\ACRshortpl

```
4120 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\@ns@ACRshortpl}
```

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

```

4121 \newcommand*{\@ns@ACRshortpl}[2][ ]{%
4122 \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2}[ ]}%
4123 }

```

Read in the final optional argument:

```

4124 \def\@ACRshortpl#1#2[#3]{%
4125 \glsdoifexists{#2}%
4126 {%

4127 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4128 \def\glslabel{#2}%
4129 \let\glsifplural\@firstoftwo
4130 \let\glscapscase\@thirdofthree
4131 \let\glsinsert\@empty
4132 \def\glscustomtext{%
4133 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
4134 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4135 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4136 }%

4137 \glspostlinkhook
4138 }

```

\acrlong

```
4139 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}
```

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

```

4140 \newcommand*{\@ns@acrlong}[2][ ]{%
4141 \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%
4142 }

```

Read in the final optional argument:

```
4143 \def\@acrlong#1#2[#3]{%
4144   \glsdoifexists{#2}%
4145   {%
4146     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4147     \def\glslabel{#2}%
4148     \let\glsifplural\@secondoftwo
4149     \let\gls caps case\@firstofthree
4150     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```
4151   \def\gls customtext{%
4152     \glsentrylong{#2}#3%
4153   }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\gls type`.

```
4154   \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4155   }%
4156   \gls post link hook
4157 }
```

`\Acrlong`

```
4158 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}
```

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

```
4159 \newcommand*{\@ns@Acrlong}[2] [] {%
4160   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}%
4161 }
```

Read in the final optional argument:

```
4162 \def\@Acrlong#1#2[#3]{%
4163   \glsdoifexists{#2}%
4164   {%
4165     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4166     \def\glslabel{#2}%
4167     \let\glsifplural\@secondoftwo
4168     \let\gls caps case\@secondofthree
4169     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```
4170   \def\gls customtext{%
4171     \Glsentrylong{#2}#3%
4172   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4173 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4174 }%  
  
4175 \glspostlinkhook  
4176 }
```

`\ACRlong`

```
4177 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\ns@ACRlong}
```

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

```
4178 \newcommand*{\ns@ACRlong}[2][]{%  
4179 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[ ]}%  
4180 }
```

Read in the final optional argument:

```
4181 \def\@ACRlong#1#2[#3]{%  
4182 \glsdoifexists{#2}%  
4183 {%  
  
4184 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4185 \def\glslabel{#2}%  
4186 \let\glsifplural\@secondoftwo  
4187 \let\glscapscase\@thirdofthree  
4188 \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4189 \def\glscustomtext{%  
4190 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%  
4191 }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4192 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4193 }%  
  
4194 \glspostlinkhook  
4195 }
```

Short plural:

`\acrlongpl`

```
4196 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\ns@acrlongpl}
```

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

```
4197 \newcommand*{\ns@acrlongpl}[2][]{%  
4198 \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[ ]}%  
4199 }
```

Read in the final optional argument:

```
4200 \def\@acrlongpl#1#2[#3]{%
4201   \glsdoifexists{#2}%
4202   {%
4203     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4204     \def\glslabel{#2}%
4205     \let\glsifplural\@firstoftwo
4206     \let\gls caps case\@firstofthree
4207     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```
4208   \def\gls customtext{%
4209     \glsentrylongpl{#2}#3%
4210   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\gls type`.

```
4211   \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4212   }%
4213   \gls post link hook
4214 }
```

`\Acrlongpl`

```
4215 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

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

```
4216 \newcommand*{\@ns@Acrlongpl}[2][ ]{%
4217   \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2}[]}%
4218 }
```

Read in the final optional argument:

```
4219 \def\@Acrlongpl#1#2[#3]{%
4220   \glsdoifexists{#2}%
4221   {%
4222     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4223     \def\glslabel{#2}%
4224     \let\glsifplural\@firstoftwo
4225     \let\gls caps case\@secondofthree
4226     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```
4227   \def\gls customtext{%
4228     \Glsentrylongpl{#2}#3%
4229   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glsstyle`.

```
4230 \gls@link[#1]{#2}{\csname gls@glstyle @entryfmt\endcsname}%
4231 }%

4232 \glspostlinkhook
4233 }
```

`\ACRlongpl`

```
4234 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}
```

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

```
4235 \newcommand*{\ns@ACRlongpl}[2][{}]{%
4236 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
4237 }
```

Read in the final optional argument:

```
4238 \def\@ACRlongpl#1#2[#3]{%
4239 \glsdoifexists{#2}%
4240 {%
```

```
4241 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4242 \def\glslabel{#2}%
4243 \let\glsifplural\@firstoftwo
4244 \let\gls caps case\@thirdofthree
4245 \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```
4246 \def\gls customtext{%
4247 \mfirstucMakeUppercase{\glsentrylongpl{#2}{#3}}%
4248 }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glsstyle`.

```
4249 \gls@link[#1]{#2}{\csname gls@glstyle @entryfmt\endcsname}%
4250 }%

4251 \glspostlinkhook
4252 }
```

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.

`\gls@entry@field` Generic version.

```
\@gls@entry@field{<label>}{<field>}
```

```
4253 \newcommand*{\@gls@entry@field}[2]{%
```



```
4254 \csname glo@glstetoklabel{#1}@#2\endcsname
4255 }
```

glstetentryfield `\glstetentryfield{<cs>}{<label>}{<field>}`

```
4256 \newcommand*{\glstetentryfield}[3]{%
4257 \letcs{#1}{glo@glstetoklabel{#2}@#3}%
4258 }
```

Gls@entry@field Generic first letter uppercase version.

`\@Gls@entry@field{<label>}{<field>}`

```
4259 \newcommand*{\@Gls@entry@field}[2]{%
4260 \glstoifexistsordo{#1}%
4261 {%
4262 \letcs\@glo@text{glo@glstetoklabel{#1}@#2}%
4263 \ifdef\@glo@text
4264 {%
4265 \xmakefirstuc{\@glo@text}%
4266 }%
4267 {%
4268 ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4269 entry ‘\glstetoklabel{#1}’}{Check you have correctly spelt the entry
4270 label and the field name}%
4271 }%
4272 }%
4273 {%
4274 ??%
4275 }%
4276 }
```

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.

`\glstentryname`

```
4277 \newcommand*{\glstentryname}[1]{\@Gls@entry@field{#1}{name}}
```

`\Glsentryname`

```
4278 \newrobustcmd*{\Glsentryname}[1]{%
4279 \@Gls@entryname{#1}%
4280 }
```

`\@Gls@entryname` This is a workaround in the event that the user defies the warning in the manual about not using `\Glsname` or `\Glsentryname` with acronyms. First the default behaviour:

```

4281 \newcommand*{\@Gls@entryname}[1]{%
4282   \@Gls@entry@field{#1}{name}%
4283 }

```

`\s@acentryname` Now the behaviour when `\setacronymstyle` is used:

```

4284 \newcommand*{\@Gls@acentryname}[1]{%
4285   \ifglshaslong{#1}%
4286   {%
4287     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4288     \expandafter\@gls@getbody\@glo@text}\@nil
4289     \expandafter\ifx\@gls@body\glsentrylong\relax
4290     \expandafter\Glsentrylong\@gls@rest
4291   \else
4292     \expandafter\ifx\@gls@body\glsentryshort\relax
4293     \expandafter\Glsentryshort\@gls@rest
4294   \else
4295     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make `\glsentryshort` behave like `\Glsentryshort`. (This is on the assumption that the argument of `\acronymfont` is `\glsentryshort{<label>}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4296     {%
4297       \let\glsentryshort\Glsentryshort
4298       \@glo@text
4299     }%
4300   \else
4301     \xmakefirstuc{\@glo@text}%
4302   \fi
4303 \fi
4304 \fi
4305 }%
4306 {%

```

Not an acronym

```

4307   \@Gls@entry@field{#1}{name}%
4308 }%
4309 }

```

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`

```

4310 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4311 \newrobustcmd*{\Glsentrydesc}[1]{%
4312   \@Gls@entry@field{#1}{desc}%
4313 }

```

Plural form:

`\entrydescplural`

```
4314 \newcommand*{\glsentrydescplural}[1]{%
4315   \@gls@entry@field{#1}{descplural}%
4316 }
```

`\Glsentrydescplural`

```
4317 \newrobustcmd*{\Glsentrydescplural}[1]{%
4318   \@Gls@entry@field{#1}{descplural}%
4319 }
```

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`

```
4320 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4321 \newrobustcmd*{\Glsentrytext}[1]{%
4322   \@Gls@entry@field{#1}{text}%
4323 }
```

Get the plural form:

`\glsentryplural`

```
4324 \newcommand*{\glsentryplural}[1]{%
4325   \@gls@entry@field{#1}{plural}%
4326 }
```

`\Glsentryplural`

```
4327 \newrobustcmd*{\Glsentryplural}[1]{%
4328   \@Gls@entry@field{#1}{plural}%
4329 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4330 \newcommand*{\glsentrysymbol}[1]{%
4331   \@gls@entry@field{#1}{symbol}%
4332 }
```

`\Glsentrysymbol`

```
4333 \newrobustcmd*{\Glsentrysymbol}[1]{%
4334   \@Gls@entry@field{#1}{symbol}%
4335 }
```

Plural form:

trysymbolplural

```
4336 \newcommand*\glsentrysymbolplural}[1]{%
4337   \@gls@entry@field{#1}{symbolplural}%
4338 }
```

trysymbolplural

```
4339 \newrobustcmd*\Glsentrysymbolplural}[1]{%
4340   \@Gls@entry@field{#1}{symbolplural}%
4341 }
```

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

```
4342 \newcommand*\glsentryfirst}[1]{%
4343   \@gls@entry@field{#1}{first}%
4344 }
```

\Glsentryfirst

```
4345 \newrobustcmd*\Glsentryfirst}[1]{%
4346   \@Gls@entry@field{#1}{first}%
4347 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

ntryfirstplural

```
4348 \newcommand*\glsentryfirstplural}[1]{%
4349   \@gls@entry@field{#1}{firstpl}%
4350 }
```

ntryfirstplural

```
4351 \newrobustcmd*\Glsentryfirstplural}[1]{%
4352   \@Gls@entry@field{#1}{firstpl}%
4353 }
```

sentrytitlecase

```
4354 \newrobustcmd*\@glsentrytitlecase}[2]{%
4355   \glsfieldfetch{#1}{#2}{\@gls@value}%
4356   \xcapitalisewords{\@gls@value}%
4357 }
4358 \ifdef\texorpdfstring
4359 {
4360   \newcommand*\glsentrytitlecase}[2]{%
4361     \texorpdfstring
4362       {\@glsentrytitlecase{#1}{#2}}%
4363       {\@gls@entry@field{#1}{#2}}%
4364   }
4365 }
4366 {
```

```
4367 \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4368 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

`\glsentrytype`

```
4369 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitized, so unexpected results may occur if the sort key contained commands.

`\glsentrysort`

```
4370 \newcommand*{\glsentrysort}[1]{%
4371 \@gls@entry@field{#1}{sort}}%
4372 }
```

`\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.

```
4373 \newcommand*{\glsentryuseri}[1]{%
4374 \@gls@entry@field{#1}{useri}}%
4375 }
```

`\Glsentryuseri`

```
4376 \newrobustcmd*{\Glsentryuseri}[1]{%
4377 \@Gls@entry@field{#1}{useri}}%
4378 }
```

`\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.

```
4379 \newcommand*{\glsentryuserii}[1]{%
4380 \@gls@entry@field{#1}{userii}}%
4381 }
```

`\Glsentryuserii`

```
4382 \newrobustcmd*{\Glsentryuserii}[1]{%
4383 \@Gls@entry@field{#1}{userii}}%
4384 }
```

`\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.

```
4385 \newcommand*{\glsentryuseriii}[1]{%
4386 \@gls@entry@field{#1}{useriii}}%
4387 }
```

`\Glsentryuseriii`

```
4388 \newrobustcmd*{\Glsentryuseriii}[1]{%
4389 \@Gls@entry@field{#1}{useriii}}%
4390 }
```

`\glentryuseriv` Get the fourth user key (as specified by the user4 when the entry was defined). The argument is the label associated with the entry.

```
4391 \newcommand*{\glentryuseriv}[1]{%
4392   \@gls@entry@field{#1}{useriv}%
4393 }
```

`\Glsentryuseriv`

```
4394 \newrobustcmd*{\Glsentryuseriv}[1]{%
4395   \@Gls@entry@field{#1}{useriv}%
4396 }
```

`\glentryuseriv` Get the fifth user key (as specified by the user5 when the entry was defined). The argument is the label associated with the entry.

```
4397 \newcommand*{\glentryuseriv}[1]{%
4398   \@gls@entry@field{#1}{useriv}%
4399 }
```

`\Glsentryuseriv`

```
4400 \newrobustcmd*{\Glsentryuseriv}[1]{%
4401   \@Gls@entry@field{#1}{useriv}%
4402 }
```

`\glentryuseriv` Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.

```
4403 \newcommand*{\glentryuseriv}[1]{%
4404   \@gls@entry@field{#1}{useriv}%
4405 }
```

`\Glsentryuseriv`

```
4406 \newrobustcmd*{\Glsentryuseriv}[1]{%
4407   \@Gls@entry@field{#1}{useriv}%
4408 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4409 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4410 \newrobustcmd*{\Glsentryshort}[1]{%
4411   \@Gls@entry@field{#1}{short}%
4412 }
```

`glsentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4413 \newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4414 \newrobustcmd*{\Glsentryshortpl}[1]{%
4415   \@Gls@entry@field{#1}{shortpl}}%
4416 }
```

`\glsentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4417 \newcommand*{\glsentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4418 \newrobustcmd*{\Glsentrylong}[1]{%
4419   \@Gls@entry@field{#1}{long}}%
4420 }
```

`\glsentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4421 \newcommand*{\glsentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4422 \newrobustcmd*{\Glsentrylongpl}[1]{%
4423   \@Gls@entry@field{#1}{longpl}}%
4424 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4425 \newcommand*{\glsentryfull}[1]{%
4426   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4427 }
```

`\Glsentryfull`

```
4428 \newrobustcmd*{\Glsentryfull}[1]{%
4429   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4430 }
```

`\glsentryfullpl`

```
4431 \newcommand*{\glsentryfullpl}[1]{%
4432   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4433 }
```

`\Glsentryfullpl`

```
4434 \newrobustcmd*{\Glsentryfullpl}[1]{%
4435   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4436 }
```

entrynumberlist Displays the number list as is.

```
4437 \newcommand*\glsentrynumberlist[1]{%
4438   \glsdoifexists{#1}%
4439   {%
4440     \gls@entry@field{#1}{numberlist}%
4441   }%
4442 }
```

splaynumberlist Formats the number list for the given entry label. Doesn't work with hyperref.

```
4443 \@ifpackageloaded{hyperref} {%
4444   \newcommand*\glsdisplaynumberlist[1]{%
4445     \GlossariesWarning
4446     {%
4447       \string\glsdisplaynumberlist\space
4448       doesn't work with hyperref.^^JUsing
4449       \string\glsentrynumberlist\space instead%
4450     }%
4451     \glsentrynumberlist{#1}%
4452   }%
4453 }%
4454 {%
4455   \newcommand*\glsdisplaynumberlist[1]{%
4456     \glsdoifexists{#1}%
4457     {%
4458       \bgroup
4459
4460       \edef\@glo@label{\glsdetoklabel{#1}}%
4461       \let\@org@glsnumberformat\glsnumberformat
4462       \def\glsnumberformat##1{##1}%
4463       \protected@edef\the@numberlist{%
4464         \csname glo@\@glo@label @numberlist\endcsname}%
4465       \def\@gls@numlist@sep{}%
4466       \def\@gls@numlist@nextsep{}%
4467       \def\@gls@numlist@lastsep{}%
4468       \def\@gls@thislist{}%
4469       \def\@gls@donext@def{}%
4470       \renewcommand\do[1]{%
4471         \protected@edef\@gls@thislist{%
4472           \@gls@thislist
4473           \noexpand\@gls@numlist@sep
4474           ##1%
4475         }%
4476         \let\@gls@numlist@sep\@gls@numlist@nextsep
4477         \def\@gls@numlist@nextsep{\glsnumlistsep}%
4478         \@gls@donext@def
4479         \def\@gls@donext@def{%
4480           \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4481         }%
4482       }%
4483     }%
4484   }
```



```

4482     \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4483     \let\@gls@numlist@sep\@gls@numlist@lastsep
4484     \@gls@thislist
4485     \egroup
4486 }%
4487 }
4488 }

```

`\glsnumlistsep`

```
4489 \newcommand*{\glsnumlistsep}{, }
```

`\glsnumlistlastsep`

```
4490 \newcommand*{\glsnumlistlastsep}{ \& }
```

`\gls hyperlink`

Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\gls link` or `\gls add` 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.

```

4491 \newcommand*{\gls hyperlink}[2][\gls entrytext{\@glo@label}]{%
4492 \def\@glo@label{#2}%
4493 \@gls link{\glo link prefix\gls detok label{#2}}{#1}}

```

1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\gls add` and `\gls add all`:

```

4494 \define@key{gloss add}{counter}{\def\@gls@counter{#1}}
4495 \define@key{gloss add}{format}{\def\@gls number format{#1}}

```

This key is only used by `\gls add all`:

```
4496 \define@key{gloss add}{types}{\def\@glo@type{#1}}
```

```
\gls add[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).

`\gls add`

```
4497 \newrobustcmd*{\gls add}[2][ ]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4498 \@gls@adjustmode
4499 \gls do if exists{#2}%
4500 {%
4501 \def\@gls number format{gls number format}%

```

```

4502 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
4503 \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4504 \@gls@saveentrycounter

```

This should use `\@do@wrglossary` rather than `\do@wrglossary` since the whole point of `\glsadd` is to add a line to the glossary.

```

4505 \@do@wrglossary{#2}%
4506 }%
4507 }

```

`@gls@adjustmode`

```

4508 \newcommand*{\@gls@adjustmode}{}
4509 \AtBeginDocument{\renewcommand*{\@gls@adjustmode}{\ifvmode\mbox{}\fi}}

```

```
\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`

```

4510 \newrobustcmd*{\glsaddall}[1] [] {%
4511 \edef\@glo@type{\@glo@types}%
4512 \setkeys{glossadd}{#1}%
4513 \forallglsentries[\@glo@type]{\@glo@entry}{%
4514 \glsadd[#1]{\@glo@entry}%
4515 }%
4516 }

```

`\glsaddallunused`

```
\glsaddallunused[<glossary type>]
```

Add all used terms defined for the listed glossaries (without displaying any text). If optional argument is omitted, apply to all glossary types. This should typically go at the end of the document.

```

4517 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
4518 \forallglsentries[#1]{\@glo@entry}%
4519 {%
4520 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4521 }%
4522 }

```

`\glsignore`

```

4523 \newcommand*{\glsignore}[1] {}

```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

```
\glsopenbrace  Define \glsopenbrace to make it easier to write an opening brace to a file.
4524 \edef\glsopenbrace{\expandafter\@gobble\string\{}}

\glsclosebrace  Define \glsclosebrace to make it easier to write an opening brace to a file.
4525 \edef\glsclosebrace{\expandafter\@gobble\string\}}

\glsbackslash  Define \glsbackslash to make it easier to write a backslash to a file.
4526 \edef\glsbackslash{\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.
4527 \edef\glsquote#1{\string"#1\string"}

\glspercentchar  Define \glspercentchar to make it easier to write a percent character to a file.
4528 \edef\glspercentchar{\expandafter\@gobble\string\%}

\glstildechar  Define \glstildechar to make it easier to write a tilde character to a file.
4529 \edef\glstildechar{\string~}

\@glsfirstletter  Define the first letter to come after the digits 0,...,9. Only required for xindy.
4530 \ifglsxindy
4531   \newcommand*{\@glsfirstletter}{A}
4532 \fi

\GlsSetXdyFirstLetterAfterDigits  Sets the first letter to come after the digits 0,...,9.
4533 \ifglsxindy
4534   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4535     \renewcommand*{\@glsfirstletter}{#1}}
4536 \else
```

```

4537 \newcommand*\GlsSetXdyFirstLetterAfterDigits}[1]{%
4538   \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
4539 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```

4540 \newcommand*\@glsminrange}{2}

```

`yMinRangeLength` 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.

```

4541 \ifglsxindy
4542   \newcommand*\GlsSetXdyMinRangeLength}[1]{%
4543     \renewcommand*\@glsminrange}{#1}}
4544 \else
4545   \newcommand*\GlsSetXdyMinRangeLength}[1]{%
4546     \glsnoxindywarning\GlsSetXdyMinRangeLength}
4547 \fi

```

`\writeist`

```

4548 \ifglsxindy

```

Code to use if xindy is required.

```

4549 \def\writeist{%

```

Define write register if not already defined

```

4550   \ifundef{\glswrite}{\newwrite\glswrite}{}%

```

Update attributes list

```

4551   \@gls@addpredefinedattributes

```

Open the file.

```

4552   \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

4553   \write\glswrite{;; xindy style file created by the glossaries

```

```

4554     package}%

```

```

4555   \write\glswrite{;; for document '\jobname' on

```

```

4556     \the\year-\the\month-\the\day}%

```

Specify the required styles

```

4557   \write\glswrite{^^J; required styles^^J}

```

```

4558   \@for\@xdystyle:=\@xdyrequiredstyles\do{%

```

```

4559     \ifx\@xdystyle\@empty

```

```

4560       \else

```

```

4561         \protected@write\glswrite}{(require

```

```

4562           \string"\@xdystyle.xdy\string")}%

```

```

4563       \fi

```

```

4564     }%

```

List the allowed attributes (possible values used by the format key)

```

4565   \write\glswrite{^^J%

```

```

4566     ; list of allowed attributes (number formats)^^J}%

```

```

4567   \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```
4568 \write\glswrite{^^J; user defined alphabets^^J}%
4569 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4570 \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.

```
4571 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4572 \protected@write\glswrite{}{(define-location-class
4573 \string"\@gls@classI\string"^^J\space\space\space
4574 (
4575 :sep "{}"
4576 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4577 :sep "}"
4578 )
4579 ^^J\space\space\space
4580 :min-range-length \@glsminrange^^J%
4581 )
4582 }%
```

Nested iteration over all classes:

```
4583 {%
4584 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4585 \protected@write\glswrite{}{(define-location-class
4586 \string"\@gls@classII-\@gls@classI\string"
4587 ^^J\space\space\space
4588 (
4589 :sep "{"
4590 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4591 :sep "}"
4592 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4593 :sep "}"
4594 )
4595 ^^J\space\space\space
4596 :min-range-length \@glsminrange^^J%
4597 )
4598 }%
4599 }%
4600 }%
4601 }%
```

User defined location classes (needs checking for new location format).

```
4602 \write\glswrite{^^J; user defined location classes}%
4603 \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 `\glsseeformat` which xindy won't recognise.)

```

4604 \write\glswrite{^^J; define cross-reference class^^J}%
4605 \write\glswrite{(define-crossref-class \string"see\string"
4606 :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.)

```

4607 \write\glswrite{(markup-crossref-list
4608 :class \string"see\string"^^J\space\space\space
4609 :open \string"\string\glsseeformat\string"
4610 :close \string"{}\string")}%

```

List the order to sort the classes.

```

4611 \write\glswrite{^^J; define the order of the location classes}%
4612 \write\glswrite{(define-location-class-order
4613 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4614 \write\glswrite{^^J; define the glossary markup^^J}%
4615 \write\glswrite{(markup-index^^J\space\space\space
4616 :open \string"\string
4617 \glossarysection[\string\glossarytoctitle]{\string
4618 \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`)

```

4619 \@for\@this@ctr:=\@xdycounters\do{%
4620   {%
4621     \@for\@this@attr:=\@xdyattributelist\do{%
4622       \protected\write\glswrite{}{\string\providecommand*%
4623         \expandafter\string
4624         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4625         {%
4626           \string\setentrycounter
4627             [\expandafter\@gobble\string\#1]{\@this@ctr}%
4628           \expandafter\string
4629           \csname\@this@attr\endcsname
4630           {\expandafter\@gobble\string\#2}%
4631         }%
4632       }%
4633     }%
4634   }%
4635 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4636 \write\glswrite{%
4637   \string\begin
4638   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4639   \space\space:close \string"\glspersentchar\glstildechar n\string

```

```

4640     \end{theglossary}\string\glossarypostamble
4641     \glstildechar n\string" ^^J\space\space\space
4642     :tree)}}%

```

Specify what to put between letter groups

```

4643     \write\glswrite{(markup-letter-group-list
4644     :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4645     \write\glswrite{(markup-indexentry
4646     :open \string"\string\relax \string\glsresetentrylist
4647     \glstildechar n\string")}%

```

Specify how to format entries

```

4648     \write\glswrite{(markup-locclass-list :open
4649     \string"\glsopenbrace\string\glossaryentrynumbers
4650     \glsopenbrace\string\relax\space \string"^^J\space\space\space
4651     :sep \string", \string"
4652     :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

4653     \write\glswrite{(markup-locref-list
4654     :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

4655     \write\glswrite{(markup-range
4656     :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.

```

4657     \@onelevel@sanitize\gls@suffixF
4658     \@onelevel@sanitize\gls@suffixFF
4659     \ifx\gls@suffixF\@empty
4660     \else
4661     \write\glswrite{(markup-range
4662     :close "\gls@suffixF" :length 1 :ignore-end)}%
4663     \fi
4664     \ifx\gls@suffixFF\@empty
4665     \else
4666     \write\glswrite{(markup-range
4667     :close "\gls@suffixFF" :length 2 :ignore-end)}%
4668     \fi

```

Specify how to format locations.

```

4669     \write\glswrite{^^J; define format to use for locations^^J}%
4670     \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4671     \write\glswrite{^^J; define letter group list format^^J}%
4672     \write\glswrite{(markup-letter-group-list
4673     :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```
4674 \write\glswrite{^^J; letter group headings^^J}%
4675 \write\glswrite{(markup-letter-group
4676 :open-head \string"\string\glsgroupheading
4677 \glsopenbrace\string"^^J\space\space\space
4678 :close-head \string"\glsclosebrace\string")}%
```

Define additional letter groups.

```
4679 \write\glswrite{^^J; additional letter groups^^J}%
4680 \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
4681 \write\glswrite{^^J; additional sort rules^^J}
4682 \write\glswrite{\@xdysortrules}%
```

Hook for any additional information:

```
4683 \@gls@writeisthook
```

Close the style file

```
4684 \closeout\glswrite
```

Suppress any further calls.

```
4685 \let\writeist\relax
4686 }
4687 \else
```

Code to use if makeindex is required.

```
4688 \edef\@gls@actualchar{\string?}
4689 \edef\@gls@encapchar{\string|}
4690 \edef\@gls@levelchar{\string!}
4691 \edef\@gls@quotechar{\string"}%
4692 \let\GlsSetQuote\gls@nosetquote
4693 \def\writeist{\relax
4694 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4695 \openout\glswrite=\istfilename
4696 \write\glswrite{\glspercentchar\space makeindex style file
4697 created by the glossaries package}
4698 \write\glswrite{\glspercentchar\space for document
4699 'jobname' on \the\year-\the\month-\the\day}
4700 \write\glswrite{actual '@gls@actualchar'}
4701 \write\glswrite{encap '@gls@encapchar'}
4702 \write\glswrite{level '@gls@levelchar'}
4703 \write\glswrite{quote '@gls@quotechar'}
4704 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4705 \write\glswrite{preamble \string"\string\glossarysection[\string
4706 \glossarytoctitle]{\string\glossarytitle}\string
4707 \glossarypreamble\string\n\string\begin{theglossary}\string
4708 \glossaryheader\string\n\string"}
4709 \write\glswrite{postamble \string"\string%\string\n\string
4710 \end{theglossary}\string\glossarypostamble\string\n
4711 \string"}
4712 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
```



```

4713     \string"}
4714 \write\glswrite{item_0 \string"\string%\string\n\string"}
4715 \write\glswrite{item_1 \string"\string%\string\n\string"}
4716 \write\glswrite{item_2 \string"\string%\string\n\string"}
4717 \write\glswrite{item_01 \string"\string%\string\n\string"}
4718 \write\glswrite{item_x1
4719     \string"\string\relax \string\glsresetentrylist\string\n
4720     \string"}
4721 \write\glswrite{item_12 \string"\string%\string\n\string"}
4722 \write\glswrite{item_x2
4723     \string"\string\relax \string\glsresetentrylist\string\n
4724     \string"}

4725 \write\glswrite{delim_0 \string"\string\{\string
4726     \glossaryentrynumbers\string\{\string\relax \string"}
4727 \write\glswrite{delim_1 \string"\string\{\string
4728     \glossaryentrynumbers\string\{\string\relax \string"}
4729 \write\glswrite{delim_2 \string"\string\{\string
4730     \glossaryentrynumbers\string\{\string\relax \string"}
4731 \write\glswrite{delim_t \string"\string\}\string\}\string"}
4732 \write\glswrite{delim_n \string"\string\delimN \string"}
4733 \write\glswrite{delim_r \string"\string\delimR \string"}
4734 \write\glswrite{headings_flag 1}
4735 \write\glswrite{heading_prefix
4736     \string"\string\glsgroupheading\string\{\string"}
4737 \write\glswrite{heading_suffix
4738     \string"\string\}\string\relax
4739     \string\glsresetentrylist \string"}
4740 \write\glswrite{symhead_positive \string"glssymbols\string"}
4741 \write\glswrite{numhead_positive \string"glnumbers\string"}
4742 \write\glswrite{page_compositor \string"glscpositor\string"}
4743 \@gls@escbsdq\gls@suffixF
4744 \@gls@escbsdq\gls@suffixFF
4745 \ifx\gls@suffixF\@empty
4746 \else
4747     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4748 \fi
4749 \ifx\gls@suffixFF\@empty
4750 \else
4751     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4752 \fi

```

Hook for any additional information:

```
4753 \@gls@writeisthook
```

Close the file and disable \writeist.

```

4754 \closeout\glswrite
4755 \let\writeist\relax
4756 }
4757 \fi

```

```

SetWriteIstHook Allow user to append information to the style file.
4758 \newcommand*\GlsSetWriteIstHook}[1]{\renewcommand*\@gls@writeisthook}{#1}}
4759 \@onlypremakeg\GlsSetWriteIstHook

ls@writeisthook
4760 \newcommand*\@gls@writeisthook}{

\GlsSetQuote Allow user to set the makeindex quote character. This is primarily for ngerman users who
want to use makeindex's -g option.
4761 \ifglxsindy
4762 \newcommand*\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4763 \newcommand*\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4764 \else
4765 \newcommand*\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}}

If German is in use, set the extra makeindex option so makeglossaries can pick it up.
4766 \@ifpackageloaded{tracklang}%
4767 {%
4768 \IfTrackedLanguage{german}%
4769 {%
4770 \def\@gls@extramakeindexopts{-g}%
4771 }%
4772 }%
4773 }%
4774 }%

Need to redefine \@gls@checkquote
4775 \edef\@gls@docheckquotedef{%
4776 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
4777 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4778 \noexpand\toks@={####1}%
4779 \noexpand\ifx\noexpand\null####2\noexpand\null
4780 \noexpand\ifx\noexpand\null####3\noexpand\null
4781 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4782 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4783 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
4784 \noexpand\else
4785 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4786 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4787 \noexpand\@gls@quotechar\noexpand\@gls@quotechar
4788 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4789 \noexpand\def\noexpand\@gls@checkquote{%
4790 \noexpand\@gls@checkquote####3\noexpand\null}%
4791 \noexpand\fi
4792 \noexpand\else
4793 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4794 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4795 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4796 \noexpand\ifx\noexpand\null####3\noexpand\null
4797 \noexpand\def\noexpand\@gls@checkquote{%

```

```

4798         \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4799     \noexpand\else
4800         \noexpand\def\noexpand\@gls@checkquote{%
4801             \noexpand\@gls@checkquote####2#1###3\noexpand\null}%
4802     \noexpand\fi
4803 \noexpand\fi
4804 \noexpand\@gls@checkquote
4805 }%
4806 }%
4807 \@gls@docheckquotedef
4808 \edef\@gls@docheckquotedef{%
4809     \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4810         \noexpand\def\noexpand\@gls@checkedmkidx{%
4811             \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4812             #1#1\noexpand\null
4813             \noexpand\expandafter\noexpand\@gls@updatechecked
4814             \noexpand\@gls@checkedmkidx{####1}%
4815             \noexpand\def\noexpand\@gls@checkedmkidx{%
4816                 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4817                 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4818                 \noexpand\null
4819                 \noexpand\expandafter\noexpand\@gls@updatechecked
4820                 \noexpand\@gls@checkedmkidx{####1}%
4821                 \noexpand\def\noexpand\@gls@checkedmkidx{%
4822                     \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4823                     \noexpand\?\noexpand\?\noexpand\null
4824                     \noexpand\expandafter\noexpand\@gls@updatechecked
4825                     \noexpand\@gls@checkedmkidx{####1}%
4826                     \noexpand\def\noexpand\@gls@checkedmkidx{%
4827                         \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4828                         \noexpand?\noexpand?\noexpand\null
4829                         \noexpand\expandafter\noexpand\@gls@updatechecked
4830                         \noexpand\@gls@checkedmkidx{####1}%
4831                         \noexpand\def\noexpand\@gls@checkedmkidx{%
4832                             \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
4833                             \noexpand|\noexpand|\noexpand\null
4834                             \noexpand\expandafter\noexpand\@gls@updatechecked
4835                             \noexpand\@gls@checkedmkidx{####1}%
4836                             \noexpand\def\noexpand\@gls@checkedmkidx{%
4837                                 \noexpand\expandafter\noexpand\@gls@checkesbar####1\noexpand\@nil
4838                                 \noexpand\|\noexpand\|\noexpand\null
4839                                 \noexpand\expandafter\noexpand\@gls@updatechecked
4840                                 \noexpand\@gls@checkedmkidx{####1}%
4841                                 \noexpand\def\noexpand\@gls@checkedmkidx{%
4842                                     \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
4843                                     \noexpand!\noexpand!\noexpand\null
4844                                     \noexpand\expandafter\noexpand\@gls@updatechecked
4845                                     \noexpand\@gls@checkedmkidx{####1}%
4846                                 }%

```

```

4847 }%
4848 \@gls@docheckquotedef
4849 \edef\@gls@docheckquotedef{%
4850   \noexpand\def\noexpand\@gls@checkescquote####1%
4851     \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
4852     ####3\noexpand\null{%
4853       \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4854       \noexpand\toks@={####1}%
4855       \noexpand\ifx\noexpand\null####2\noexpand\null
4856         \noexpand\ifx\noexpand\null####3\noexpand\null
4857         \noexpand\edef\noexpand\@gls@checkedmkidx{%
4858           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4859         \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
4860         \noexpand\else
4861         \noexpand\edef\noexpand\@gls@checkedmkidx{%
4862           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4863           \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4864             \csname#1\endcsname}\noexpand\@gls@quotechar
4865           \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4866             \csname#1\endcsname}\noexpand\@gls@quotechar}%
4867         \noexpand\def\noexpand\@gls@checkescquote{%
4868           \noexpand\@gls@checkescquote####3\noexpand\null}%
4869         \noexpand\fi
4870         \noexpand\else
4871         \noexpand\edef\noexpand\@gls@checkedmkidx{%
4872           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4873           \noexpand\@gls@quotechar\noexpand\string
4874           \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
4875         \noexpand\ifx\noexpand\null####3\noexpand\null
4876         \noexpand\def\noexpand\@gls@checkescquote{%
4877           \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4878           \expandonce{\csname#1\endcsname}\noexpand\null}%
4879         \noexpand\else
4880         \noexpand\def\noexpand\@gls@checkescquote{%
4881           \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4882           ####3\noexpand\null}%
4883         \noexpand\fi
4884         \noexpand\fi
4885         \noexpand\@gls@checkescquote
4886       }%
4887     }%
4888   \@gls@docheckquotedef
4889 }
4890 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
4891   {\string\GlsSetQuote\space not permitted here}%
4892   {Move \string\GlsSetQuote\space earlier in the preamble, as
4893     soon as possible after glossaries.sty has been loaded}}
4894 \fi

```

ramakeindexopts

```
4895 \newcommand*{\@gls@extramakeindexopts}[1]{}
```

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`

```
4896 \newcommand{\noist}{%
```

Update attributes list

```
4897 \@gls@addpredefinedattributes
```

```
4898 \let\writeist\relax
```

```
4899 }
```

`\@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`

```
4900 \newcommand*{\@makeglossary}[1]{%
```

```
4901 \ifglossaryexists{#1}%
```

```
4902 {%
```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```
4903 \ifglssavewrites
```

```
4904 \expandafter\newtoks\csname glo@#1@filetok\endcsname
```

```
4905 \else
```

```
4906 \expandafter\newwrite\csname glo@#1@file\endcsname
```

```
4907 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
```

```
4908 \fi
```

```
4909 \@gls@renewglossary
```

```
4910 \writeist
```

```
4911 }%
```

```
4912 {%
```

```
4913 \PackageError{glossaries}%
```

```
4914 {Glossary type ‘#1’ not defined}%
```

```
4915 {New glossaries must be defined before using \string\makeglossary}%
```

```
4916 }%
```

```
4917 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```
4918 \newcommand*{\@glsopenfile}[2]{%
```

```

4919 \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4920 \PackageInfo{glossaries}{Writing glossary file
4921   \jobname.\csname @glotype@#2@out\endcsname}%
4922 }

```

\@closegls

```

4923 \newcommand*{\@closegls}[1]{%
4924   \closeout\csname glo@#1@file\endcsname
4925 }
4926 %   \end{macrocode}
4927 %\end{macro}
4928 %
4929 %\begin{macro}{\@gls@automake}
4930 %\changes{4.08}{2014-07-30}{new}
4931 %   \begin{macrocode}
4932 \ifglxindy
4933   \newcommand*{\@gls@automake}[1]{%
4934     \ifglossaryexists{#1}
4935     {%
4936       \@closegls{#1}%
4937       \ifdefstring{\glsorder}{letter}%
4938         {\def\@gls@order{-M ord/letorder }}%
4939         {\let\@gls@order\@empty}%
4940       \ifcsundef{@xdy@#1@language}%
4941         {\let\@gls@langmod\@xdy@main@language}%
4942         {\letcs\@gls@langmod{@xdy@#1@language}}%
4943       \edef\@gls@dothiswrite{\noexpand\write18{xindy
4944         -I xindy
4945         \@gls@order
4946         -L \@gls@langmod\space
4947         -M \gls@istfilebase\space
4948         -C \gls@codepage\space
4949         -t \jobname.\csuse{@glotype@#1@log}
4950         -o \jobname.\csuse{@glotype@#1@in}
4951         \jobname.\csuse{@glotype@#1@out}}}%
4952     }%
4953     \@gls@dothiswrite
4954   }%
4955   {%
4956     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4957   }%
4958 }
4959 \else
4960 \newcommand*{\@gls@automake}[1]{%
4961   \ifglossaryexists{#1}
4962   {%
4963     \@closegls{#1}%
4964     \ifdefstring{\glsorder}{letter}%
4965     {\def\@gls@order{-l }}%

```

```

4966     {\let\@gls@order\@empty}%
4967     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4968     -s \istfilename\space
4969     -t \jobname.\csuse{@glotype@#1@log}
4970     -o \jobname.\csuse{@glotype@#1@in}
4971     \jobname.\csuse{@glotype@#1@out}}}%
4972     }%
4973     \@gls@dothiswrite
4974 }%
4975 {%
4976     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4977 }%
4978 }
4979 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
4980 \newcommand*{\@warn@nomakeglossaries}{}%
```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```
4981 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}
```

`\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`

```
4982 \newcommand*{\makeglossaries}{}%
```

Define the write used for style file also used for all other output files if `savewrites=true`.

```
4983 \ifundef{glswrite}{\newwrite\glswrite}{}%
```

If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
4984 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}}
```

```
4985 \protected@write\@auxout{}{\string\providecommand\string\istfilename[1]{}}
```

If `\@gls@extramakeindexopts` has been defined, write it:

```
4986 \ifundef\@gls@extramakeindexopts
```

```
4987 {}%
```

```
4988 {%
```

```
4989     \protected@write\@auxout{}{\string\providecommand
```

```
4990         \string\@gls@extramakeindexopts[1]{}}
```

```
4991     \protected@write\@auxout{}{\string\@gls@extramakeindexopts
```

```
4992         {\@gls@extramakeindexopts}}%
```

```
4993 }%
```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```
4994 \protected@write\@auxout{}{\string\istfilename{\istfilename}}%
```

```
4995 \protected@write\@auxout{}{\string\@glsorder{\@glsorder}}
```

Iterate through each glossary type and activate it.

```
4996 \for@glo@type:=@glo@types\do{%
4997   \ifthenelse{equal{@glo@type}{}}{ }{%
4998     \@makeglossary{@glo@type}}%
4999   }%
```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```
5000 \renewcommand*\newglossary[4] []{%
5001   \PackageError{glossaries}{New glossaries
5002   must be created before \string\makeglossaries}{You need
5003   to move \string\makeglossaries\space after all your
5004   \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
5005 \let\@makeglossary\relax
5006 \let\makeglossary\relax
5007 \let\makeglossaries\relax
```

Disable all commands that have no effect after `\makeglossaries`

```
5008 \@disable@onlypremakeg
```

Allow see key:

```
5009 \let\gls@checkseeallowed\relax
```

Suppress warning about no `\makeglossaries`

```
5010 \let\warn@nomakeglossaries\relax
```

Activate warning about missing `\printglossary`

```
5011 \def\warn@noprintglossary{%
5012   \ifdefstring{@glo@types}{,}%
5013   {%
5014     \GlossariesWarningNoLine{No glossaries have been defined}%
5015   }%
5016   {%
5017     \GlossariesWarningNoLine{No \string\printglossary\space
5018     or \string\printglossaries\space
5019     found. ^^J(Remove \string\makeglossaries\space if you
5020     don't want any glossaries.) ^^JThis document will not
5021     have a glossary}%
5022   }%
5023 }%
```

Declare list parser for `\glsdisplaynumberlist`

```
5024 \ifglssavenumberlist
5025   \edef@gls@dodolistparser{\noexpand\DeclareListParser
5026     {\noexpand\glsnumlistparser}{\delimN}}%
5027   \@gls@dodolistparser
5028 \fi
```

Prevent user from also using `\makenoidxglossaries`

```
5029 \let\makenoidxglossaries\@no@makeglossaries
```


Prohibit sort key in printgloss family:

```
5030 \renewcommand*{\@printgloss@setsort}{%
5031   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5032 }%
```

Check the automake setting:

```
5033 \ifglsautomake
5034   \renewcommand*{\@gls@doautomake}{%
5035     \@for\@gls@type:=\@glo@types\do{%
5036       \ifdefempty{\@gls@type}{}%
5037       {\@gls@automake{\@gls@type}}%
5038     }%
5039   }%
5040 \fi
5041 }
```

Must occur in the preamble:

```
5042 \@onlypreamble{\makeglossaries}
```

`\glswrite` The definition of `\glswrite` has now been moved to `\makeglossaries` so that it's only defined if needed.

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`

```
5043 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
5044 \AtEndDocument{%
5045   \warn@nomakeglossaries
5046   \warn@noprintglossary
5047 }
```

`\noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
5048 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5049   \renewcommand{\@gls@noref@warn}[1]{%
5050     \GlossariesWarning{Empty glossary for
5051     \string\printnoidxglossary[type={##1}].
5052     Rerun may be required (or you may have forgotten to use
5053     commands like \string\gls)}}%
5054   }%
```

Don't escape makeindex/xindy characters

```
5055   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
5056 \let\do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5057 \let\gls@getgrouptitle\gls@noidx@getgrouptitle
```

Allow see key:

```
5058 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5059 \renewcommand{\do@seeglossary}[2]{%
5060   \edef\gls@label{\glsdetoklabel{##1}}%
5061   \protected@write\@auxout{}{%
5062     \string\gls@reference
5063     {\csname glo@\gls@label @type\endcsname}%
5064     {\gls@label}%
5065     {%
5066       \string\glsseeformat##2}%
5067     }%
5068   }%
5069 }
```

If user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
5070 \AtBeginDocument
5071 {%
5072   \write\@auxout{\string\providecommand\string\gls@reference[3]{}}%
5073 }
```

Change warning about no glossares

```
5074 \def\warn@noprintglossary{%
5075   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5076     or \string\printnoidxglossaries ^^J
5077     found. (Remove \string\makenoidxglossaries\space if you
5078     don't want any glossaries.)^^JThis document will not have a glossary}%
5079 }
```

Suppress warning about no \makeglossaries

```
5080 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5081 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5082 \renewcommand*{\@printgloss@setsort}{%
5083   \let\glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5084   \def\glo@sorttype{\glo@default@sorttype}%
5085 }
```

All entries must be defined in the preamble:

```
5086 \renewcommand*\new@glossaryentry[2]{%
5087   \PackageError{glossaries}{Glossary entries must be
5088   defined in the preamble^^Jwhen you use
5089   \string\makenoid@glossaries}%
5090   {Either move your definitions to the preamble or use
5091   \string\makeglossaries}%
5092 }%
```

Redefine \glsentrynumberlist

```
5093 \renewcommand*\glsentrynumberlist[1]{%
5094   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5095   \ifdef\@gls@loclist
5096   {%
5097     \glsnoidxloclist{\@gls@loclist}%
5098   }%
5099   {%
5100     ??\glsdoifexists{##1}%
5101     {%
5102       \GlossariesWarning{Missing location list for ‘##1’. Either
5103       a rerun is required or you haven’t referenced the entry}%
5104     }%
5105   }%
5106 }%
```

Redefine \glsdisplaynumberlist

```
5107 \renewcommand*\glsdisplaynumberlist[1]{%
5108   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5109   \ifdef\@gls@loclist
5110   {%
5111     \def\@gls@noidxloclist@sep{%
5112       \def\@gls@noidxloclist@sep{%
5113         \def\@gls@noidxloclist@sep{%
5114           \glsnumlistsep
5115         }%
5116       \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5117     }%
5118   }%
5119   \def\@gls@noidxloclist@finalsep{}%
5120   \def\@gls@noidxloclist@prev{}%
5121   \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5122   \@gls@noidxloclist@finalsep
5123   \@gls@noidxloclist@prev
5124 }%
5125 {%
5126   ??\glsdoifexists{##1}%
5127   {%
5128     \GlossariesWarning{Missing location list for ‘##1’. Either
5129     a rerun is required or you haven’t referenced the entry}%
5130   }%
```

```
5131 }%
5132 }%
```

Provide a generic way of iterating through the number list:

```
5133 \renewcommand*\glsnumberlistloop}[3]{%
5134   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5135   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5136   \let\@gls@org@glsseeformat\glsseeformat
5137   \let\glsnoidxdisplayloc##2\relax
5138   \let\glsseeformat##3\relax
5139   \ifdef\@gls@loclist
5140     {%
5141       \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5142     }%
5143     {%
5144       ??\glsdoifexists{##1}%
5145       {%
5146         \GlossariesWarning{Missing location list for ‘##1’. Either
5147           a rerun is required or you haven’t referenced the entry}%
5148       }%
5149     }%
5150   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5151   \let\glsseeformat\@gls@org@glsseeformat
5152 }%
```

Modify sanitize sort function

```
5153 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5154 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5155 \@gls@noidx@setsanitizesort
5156 }
```

Preamble-only command:

```
5157 \@onlypreamble{\makenoidxglossaries}
```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```
5158 \newcommand*\glsnumberlistloop}[2]{%
5159   \PackageError{glossaries}{\string\glsnumberlistloop\space
5160     only works with \string\makenoidxglossaries}{}%
5161 }
```

`\listloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc{<prefix>}`)

```
5162 \newcommand*\glsnoidxnumberlistloophandler}[1]{%
5163   #1%
5164 }
```

`@makeglossaries` Can’t use both `\makeglossaries` and `\makenoidxglossaries`

```
5165 \newcommand*\@no@makeglossaries}{%
5166   \PackageError{glossaries}{You can’t use both
```

```

5167 \string\makeglossaries\space and \string\makenoidxglossaries}%
5168 {Either use one or other (or none) of those commands but not both
5169 together.}%
5170 }

```

`\@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

5171 \newcommand{\@gls@noref@warn}[1]{%
5172 \GlossariesWarning{\string\makenoidxglossaries\space
5173 is required to make \string\printnoidxglossary[type={#1}] work}%
5174 }

```

`\@noidxglossary` Write the glossary information to the aux file:

```

5175 \newcommand*{\@gls@noidxglossary}{%
5176 \protected@write\@auxout{}{%
5177 \string\@gls@reference
5178 {\csname glo@\@gls@label @type\endcsname}%
5179 {\@gls@label}%
5180 {\string\glsnoidxdisplayloc
5181 {\@glo@counterprefix}%
5182 {\@gls@counter}%
5183 {\@glsnumberformat}%
5184 {\@glslocref}%
5185 }%
5186 }%
5187 }

```

1.14 Writing information to associated files

`\istfile` Deprecated.

```

5188 \def\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

5189 \AtEndDocument{%
5190 \glswritefiles
5191 }

```

`\@glswritefiles` Only write the files if `savewrites=true`

```

5192 \newcommand*{\@glswritefiles}{%

```

Iterate through all the glossaries

```

5193 \forallglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```

5194 \ifcsundef{glo@\@glo@type @filetok}%
5195 {%
5196 \def\gls@tmp{}%
5197 }%
5198 {%

```

```

5199     \edef\gls@tmp{\expandafter\the
5200         \csname glo@\@glo@type @filetok\endcsname}%
5201     }%
5202     \ifx\gls@tmp\@empty
5203         \ifx\@glo@type\glsdefaulttype
5204             \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5205                 entries.^^JRemember to use package option ‘nomain’ if
5206 you
5207                 don’t want to^^Juse the main glossary}%
5208         \else
5209             \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5210                 entries}%
5211         \fi
5212     \else
5213         \@glsopenfile{\glswrite}{\@glo@type}%
5214         \immediate\write\glswrite{%
5215             \expandafter\the
5216                 \csname glo@\@glo@type @filetok\endcsname}%
5217         \immediate\closeout\glswrite
5218     \fi
5219 }%
5220 }

```

As from v4.10, the `\glossary` command is used by the `glossaries` package. Since the user isn’t expected to use this command (as `glossaries` takes care of the particular format required for `makeindex/xindy`) there’s no need for a user level command. Using a custom internal command prevents any conflict with other packages (and with the `\mark` mechanism).

In v4.10, the redefinition of `\glossary` was removed since it wasn’t intended as a user level command, however it seems there are packages that have hacked the internal macros used by `glossaries` and no longer work with this redefinition removed, so it’s been restored in v4.11 but is not used at all by `glossaries`. (This may be removed or moved to a compatibility mode in future.)

`\glossary`

```

5221 \if@gls@docloaded
5222 \else
5223   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5224 \fi

```

The associated number should be stored in `\theglentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

5225 \newcommand*{\gls@glossary}[1]{%
5226   \@gls@glossary{#1}%
5227 }

```

`\@gls@glossary` (In v4.10, `\@glossary` was redefined to `\@gls@glossary` to avoid conflict with other packages.) Define internal `\@gls@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.) The argument #1 is the glossary type.

```
5228 \newcommand*{\@gls@glossary}[2]{%
5229   \if@gls@debug
5230     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5231   \fi
5232   \index{#2}%
5233 }
```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`s@renewglossary`

```
5234 \newcommand{\@gls@renewglossary}{%
5235   \gdef\@gls@glossary##1{\@bsphack\begin@group\gls@wrglossary{##1}}%
5236   \let\@gls@renewglossary\@empty
5237 }
```

The `\gls@wrglossary` command is defined 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`).

`\gls@wrglossary`

```
5238 \newcommand*{\gls@wrglossary}[2]{%
5239   \ifglssavewrites
5240     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5241     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5242       \expandafter{\@gls@tmp^^J}%
5243   \else
5244     \ifcsdef{glo@#1@file}%
5245     {%
5246       \expandafter\protected@write\csname glo@#1@file\endcsname{%
5247         \gls@disablepagerefexpansion}{#2}%
5248     }%
5249     {%
5250       \ifignoredglossary{#1}{}%
5251       {%
5252         \GlossariesWarning{No file defined for glossary '#1'}%
5253       }%
5254     }%
5255   \fi
5256   \endgroup\@esphack
5257 }
```

`\@do@wrglossary`

```
5258 \newcommand*{\@do@wrglossary}[1]{%
5259   \glswriteentry{#1}{\@do@wrglossary{#1}}%
5260 }
```

`\glswriteentry` Provide a user level command so the user can customize whether or not a line should be added to the glossary. The arguments are the label and the code that writes to the glossary file.

```
5261 \newcommand*{\glswriteentry}[2]{%
5262   \ifglsindexonlyfirst
5263     \ifglsused{#1}{#2}%
5264   \else
5265     #2%
5266   \fi
5267 }
```

`protected@pagefmts` List of page formats to be protected against expansion.

```
5268 \newcommand{\gls@protected@pagefmts}{%
5269   \gls@numberpage, \gls@alphpage, \gls@Alphpage, \gls@romanpage, \gls@Romanpage, \gls@arabicpage%
5270 }
```

`pagerefexpansion`

```
5271 \newcommand*{\gls@disablepagerefexpansion}{%
5272   \@for\@gls@this:=\gls@protected@pagefmts\do
5273   {%
5274     \expandafter\let\@gls@this\relax
5275   }%
5276 }
```

`\gls@alphpage`

```
5277 \newcommand*{\gls@alphpage}{\@alph\c@page}
```

`\gls@Alphpage`

```
5278 \newcommand*{\gls@Alphpage}{\@Alph\c@page}
```

`\gls@numberpage`

```
5279 \newcommand*{\gls@numberpage}{\number\c@page}
```

`\gls@arabicpage`

```
5280 \newcommand*{\gls@arabicpage}{\@arabic\c@page}
```

`\gls@romanpage`

```
5281 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5282 \newcommand*{\gls@Romanpage}{\@Roman\c@page}
```

`protectedpagefmt`

```
\glsaddprotectedpagefmt{<cs name>}
```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument (`\<csname>\c@page` must be valid).


```

5283 \newcommand*\glsaddprotectedpagefmt}[1]{%
5284 \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5285 \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5286 \eappto\@wrglossarynumberhook{%
5287 \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5288 \expandonce{\csname#1\endcsname}%
5289 \noexpand\def\expandonce{\csname#1\endcsname}{%
5290 \noexpand\@wrglossary@pageformat
5291 \expandonce{\csname gls#1page\endcsname}%
5292 \expandonce{\csname org@gls#1\endcsname}%
5293 }%
5294 }%
5295 }

```

`\@do@wrglossary` Hook used by `\@do@wrglossary`

```
5296 \newcommand*\@wrglossarynumberhook{}
```

`\@wrglossary@pageformat`

```

5297 \newcommand{\@wrglossary@pageformat}[3]{%
5298 \ifx#3\c@page #1\else #2#3\fi
5299 }

```

`\@wrglossary@allowprimitivemods` Conditional to determine whether or not `\@do@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```

5300 \newif\ifglswrallowprimitivemods
5301 \glswrallowprimitivemodstrue

```

`\@do@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@glsnumberformat` and `\@gls@counter` prior to use.) The argument is the entry's label.

```

5302 \newcommand*\@do@wrglossary}[1]{%
5303 \begingroup

```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```

5304 \let\orgthe\the
5305 \let\orgnumber\number

5306 \let\orgarabic\@arabic
5307 \let\orgromannumeral\romannumeral
5308 \let\orgalph\@alph
5309 \let\orgAlph\@Alph
5310 \let\orgRoman\@Roman

```

Redefine:

```

5311 \ifglswrallowprimitivemods
5312 \def\the##1{%
5313 \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5314 \def\number##1{%
5315 \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5316 \fi

```

```

5317 \def\@arabic##1{%
5318   \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5319 \def\romannumeral##1{%
5320   \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5321 \def\@Roman##1{%
5322   \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5323 \def\@alph##1{%
5324   \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5325 \def\@Alph##1{%
5326   \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5327 \@wrglossarynumberhook
```

Prevent expansion:

```
5328 \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```
5329 \protected@xdef\@glslocref{\theHglentrycounter}%
```

```
5330 \endgroup
```

Escape any special characters

```
5331 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
5332 \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
```

```
5333 \def\@glo@counterprefix{}%
```

```
5334 \else
```

```
5335 \protected@edef\@glsHlocref{\theHglentrycounter}%
```

```
5336 \@gls@checkmkidxchars\@glsHlocref
```

```
5337 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
```

```
5338   {\@glslocref}{\@glsHlocref}%
```

```
5339   }%
```

```
5340 \do@gls@getcounterprefix
```

```
5341 \fi
```

De-tok label if required

```
5342 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5343 \@do@@wrglossary
```

```
5344 }
```

```
@do@@wrglossary
```

```
5345 \newcommand*{\@do@@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5346 \ifglxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```
5347 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
```

```
5348 \def\@glo@range{}%
```

```
5349 \expandafter\if\@glo@prefix(\relax
```

```

5350     \def\@glo@range{:open-range}%
5351 \else
5352     \expandafter\if\@glo@prefix)\relax
5353     \def\@glo@range{:close-range}%
5354     \fi
5355 \fi

```

Write to the glossary file using xindy syntax.

```

5356     \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5357     (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

5358         :locref \string"\@glo@counterprefix}{\@gls@locref}\string" %
5359         :attr \string"\@gls@counter\@glo@suffix\string"
5360         \@glo@range
5361     )
5362     }%
5363 \else

```

Convert the format information into the format required for makeindex

```

5364     \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@gls@numberformat}%
5365     {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5366     \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5367     \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5368     \@gls@encapchar\@glo@numfmt}{\@gls@locref}}%
5369 \fi
5370 }

```

`etcounterprefix` 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.)

```

5371 \newcommand*\@gls@getcounterprefix[2]{%
5372 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5373 \ifx\@gls@thisloc\@gls@thisHloc
5374 \def\@glo@counterprefix{}%
5375 \else
5376 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5377 \def\@glo@tmp{##2}%
5378 \ifx\@glo@tmp\@empty
5379 \def\@glo@counterprefix{}%
5380 \else
5381 \def\@glo@counterprefix{##1}%
5382 \fi
5383 }%
5384 \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5385 \ifx\@glo@counterprefix\@empty
5386 \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5387 prefixing^^Jlocation ‘#1’. You need to modify the
5388 definition of \string\theH\@gls@counter^^Jotherwise you
5389 will get the warning: "‘name{\@gls@counter.#1}’ has been^^J
5390 referenced but does not exist"}%
5391 \fi
5392 \fi
5393 }

```

1.15 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.

```

5394 \newcommand{\@do@seeglossary}[2]{%
5395 \def\@gls@xref{#2}%
5396 \@onelevel@sanitize\@gls@xref
5397 \@gls@checkmkidxchars\@gls@xref
5398 \ifglsxindy
5399 \gls@glossary{\csname glo@#1@type\endcsname}{%
5400 (indexentry
5401 :tkey (\csname glo@#1@index\endcsname)
5402 :xref (\string"\@gls@xref\string")
5403 :attr \string"see\string"
5404 )
5405 }%
5406 \else
5407 \gls@glossary{\csname glo@#1@type\endcsname}{%
5408 \string@glossaryentry{\csname glo@#1@index\endcsname
5409 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5410 \fi
5411 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5412 \def\@gls@fixbraces#1#2#3\@nil{%
5413 \ifx#2[\relax
5414 \@gls@fixbraces#1#2#3\@end@fixbraces
5415 \else
5416 \def#1{{#2#3}}%
5417 \fi
5418 }

```

`@@gls@fixbraces`

```

5419 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5420 \def#1{[#2]{#3}}%
5421 }

```

```

\glssee \glssee{<label>}{<cross-ref list>}
5422 \DeclareRobustCommand*\glssee}[3][\seename]{%
5423 \do@seeglossary{#2}{#1}{#3}}
5424 \newcommand*\@glssee}[3][\seename]{%
5425 \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.

```

5426 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%
5427 \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5428 \DeclareRobustCommand*\glsseelist}[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5429 \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

5430 \let\@gls@donext\relax

```

Iterate through the labels

```

5431 \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5432 \ifx\@xfor@nextelement\@nnil

```

```

5433 \@gls@dolast

```

```

5434 \else

```

```

5435 \@gls@donext

```

```

5436 \fi

```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```

5437 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```

5438 \let\@gls@dolast\glsseelastsep

```

```

5439 \let\@gls@donext\glsseesep

```

```

5440 }%

```

```

5441 }

```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```

5442 \newcommand*\glsseelastsep{\space\andname\space}

```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```

5443 \newcommand*\glsseesep{, }

```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```

5444 \DeclareRobustCommand*\glsseeitem}[1]{\gls hyperlink[\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.)

```

5445 \newcommand*\glsseeitemformat}[1]{\glsentrytext{#1}}

```

1.16 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.

`save@numberlist` Provide command to store number list.

```
5446 \newcommand*\gls@save@numberlist}[1]{%
5447   \ifglssavenumberlist
5448     \toks@{#1}%
5449     \edef\@do@writeaux@info{%
5450       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5451     }%
5452     \@onelevel@sanitize\@do@writeaux@info
5453     \protected@write\@auxout{}\@do@writeaux@info}%
5454   \fi
5455 }
```

`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.)

```
5456 \newcommand*\warn@noprintglossary}{}%
```

`\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.

```
5457 \ifcsundef{printglossary}{}%
5458 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5459 \@gls@warnonglossdefined
5460 \undef\printglossary
5461 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5462 \newcommand*\printglossary}[1] [type=\glsdefaultttype]{%
5463   \@printglossary{#1}{\@print@glossary}%
5464 }
```

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

```
5465 \newcommand*\printglossaries}{%
5466   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5467 }
```

printnoidxglossary Provide an alternative to `\printglossary` that doesn't require an external indexing application. Entries won't be sorted and the location list will be empty.

```
5468 \newcommand*\printnoidxglossary}[1] [type=\glsdefaulttype]{%
5469   \@printglossary{#1}{\@printnoidx@glossary}%
5470 }
```

printnoidxglossaries Analogous to `\printglossaries`

```
5471 \newcommand*\printnoidxglossaries}{%
5472   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5473 }
```

printgloss@setsort Initialise to do nothing.

```
5474 \newcommand*\@printgloss@setsort}{}
```

preglossaryhook

```
5475 \newcommand*\@gls@preglossaryhook}{}
```

`\@printglossary` Sets up the glossary for either `\printglossary` or `\printnoidxglossary`. The first argument is the options list, the second argument is the handler macro that deals with the actual glossary.

```
5476 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5477   \def\@glo@type{\glsdefaulttype}%
5478   \def\glossarytitle{\csname @glo@type @title\endcsname}%

5479   \def\glossarytoctitle{\glossarytitle}%
5480   \let\org@glossarytitle\glossarytitle

5481   \def\@glossarystyle{%
5482     \ifx\@glossary@default@style\relax
5483       \GlossariesWarning{No default glossary style provided \MessageBreak
5484         for the glossary '@glo@type'. \MessageBreak
5485         Using deprecated fallback. \MessageBreak
5486         To fix this set the style with \MessageBreak
5487         \string\setglossarystyle\space or use the \MessageBreak
5488         style key=value option}%
5489     \fi
5490   }%
5491   \def\gls@dotoc@title{\glssettoctitle{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5492   \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5493 \bgroup
```

Activate or deactivate sort key:

```
5494 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5495 \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)

```
5496 \ifx\glossarytitle\org@glossarytitle
```

```
5497 \else
```

```
5498 \expandafter\let\csname @glo@type @title\endcsname
```

```
5499 \glossarytitle
```

```
5500 \fi
```

Allow a high-level user command to indicate the current glossary

```
5501 \let\currentglossary@glo@type
```

Enable individual number lists to be suppressed.

```
5502 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

```
5503 \let\glsnonextpages@glsnonextpages
```

Enable individual number list to be activated:

```
5504 \let\glsnextpages@glsnextpages
```

Enable suppression of description terminators.

```
5505 \let\nopostdesc@nopostdesc
```

Set up the entry for the TOC

```
5506 \gls@dotoc@title
```

Set the glossary style

```
5507 \@glossarystyle
```

Added a way to fetch the current entry label (v3.08 updated for new \glossentry and \subglossentry, but this is now only needed for backward compatibility):

```
5508 \let\gls@org@glossaryentryfield\glossentry
```

```
5509 \let\gls@org@glossarysubentryfield\subglossentry
```

```
5510 \renewcommand{\glossentry}[1]{%
```

```
5511 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
```

```
5512 \gls@org@glossaryentryfield{##1}%
```

```
5513 }%
```

```
5514 \renewcommand{\subglossentry}[2]{%
```

```
5515 \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
```

```
5516 \gls@org@glossarysubentryfield{##1}{##2}%
```

```
5517 }%
```

```
5518 \@gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
5519 #2%
```


End the current scope

```
5520 \egroup
```

Reset `\glossaryentrynumbers`

```
5521 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5522 \global\let\warn@noprntglossary\relax
```

```
5523 }
```

`@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```
5524 \newcommand{\@print@glossary}{%
```

Some macros may end up being expanded into internals in the glossary, so need to make `@` a letter. (Unlikely to be a problem since v3.08a but kept for backward compatibility.)

```
5525 \makeatletter
```

Input the glossary file, if it exists.

```
5526 \@input@{\jobname.\csname @glo@type@\@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.

```
5527 \IfFileExists{\jobname.\csname @glo@type@\@glo@type @in\endcsname}{%
```

```
5528 {}%
```

```
5529 {\null}{%
```

If `xindy` is being used, need to write the language dependent information to the `.aux` file for `makeglossaries`.

```
5530 \ifglxindy
```

```
5531 \ifcsundef{@xdy@\@glo@type @language}{%
```

```
5532 {%
```

```
5533 \edef\@do@auxoutstuff{%
```

```
5534 \noexpand\AtEndDocument{%
```

If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
5535 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5536 \string\providecommand\string\@xdy@language[2]{}}%
```

```
5537 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5538 \string\@xdy@language{\@glo@type}{\@xdy@main@language}}%
```

```
5539 }%
```

```
5540 }%
```

```
5541 }%
```

```
5542 {%
```

```
5543 \edef\@do@auxoutstuff{%
```

```
5544 \noexpand\AtEndDocument{%
```

```
5545 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5546 \string\providecommand\string\@xdy@language[2]{}}%
```

```
5547 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5548 \string\@xdy@language{\@glo@type}{\csname @xdy@\@glo@type
```

```

5549         @language\endcsname}}}%
5550     }%
5551 }%
5552 }%
5553 \do@auxoutstuff
5554 \edef\do@auxoutstuff{%
5555     \noexpand\AtEndDocument{%

```

If the user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

5556     \noexpand\immediate\noexpand\write\@auxout{%
5557         \string\providecommand\string\@gls@codepage[2]{}}}%
5558     \noexpand\immediate\noexpand\write\@auxout{%
5559         \string\@gls@codepage{\@glo@type}{\@gls@codepage}}}%
5560 }%
5561 }%
5562 \do@auxoutstuff
5563 \fi

```

Activate warning if `\makeglossaries` hasn't been used.

```

5564 \renewcommand*\@warn@nomakeglossaries{%
5565     \GlossariesWarningNoLine{\string\makeglossaries\space
5566     hasn't been used,^^Jthe glossaries will not be updated}%
5567 }%
5568 }

```

The sort macros all have the syntax:

```

\@glo@sortmacro@<order>{<type>}

```

where `<order>` is the sort order as specified by the sort key and `<type>` is the glossary type. (The referenced entry list is stored in `\@glsref@<type>`). The actual sorting is done by `\@glo@sortentries{<handler>}{<type>}`.

`glo@sortentries`

```

5569 \newcommand*\@glo@sortentries}[2]{%
5570     \def\@glo@sortinglist{}%
5571     \def\@glo@sortinghandler{#1}%
5572     \edef\@glo@type{#2}%
5573     \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
5574     \csdef{\@glsref@#2}{}%
5575     \for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5576     \xifinlistcs{\@this@label}{\@glsref@#2}%
5577     {}%
5578     {%
5579         \listcsxadd{\@glsref@#2}{\@this@label}%
5580     }%
5581     \ifcsdef{\@glo@sortingchildren@\@this@label}%

```

```

5582   {%
5583     \@glo@addchildren{#2}{\@this@label}%
5584   }%
5585   {}%
5586 }%
5587 }

```

```
@glo@addchildren \@glo@addchildren{<type>}{<parent>}
```

```
5588 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```

5589   \bgroup
5590     \letcs{\@glo@childlist}{@glo@sortingchildren@#2}%
5591     \@for\@this@childlabel:=\@glo@childlist\do
5592     {%

```

Check this label hasn't already been added.

```

5593       \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5594       {}%
5595       {%
5596         \listcsxadd{@glsref@#1}{\@this@childlabel}%
5597       }%

```

Does this child have children?

```

5598       \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5599       {%
5600         \@glo@addchildren{#1}{\@this@childlabel}%
5601       }%
5602       {%
5603       }%
5604     }%
5605   \egroup
5606 }

```

```
@do@sortentries
```

```

5607 \newcommand*{\@glo@do@sortentries}[1]{%
5608   \ifglshasparent{#1}%
5609   {%

```

This entry has a parent, so add it to the child list

```

5610     \edef\@glo@parent{\csuse{glo@glsdetoklabel{#1}@parent}}%
5611     \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5612     {%
5613       \csdef{@glo@sortingchildren@\@glo@parent}{}%
5614     }%
5615     {}%
5616     \expandafter\@glo@sortedinsert
5617     \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```
5618 \xifinlistcs{\@glo@parent}{\@glsref{\@glo@type}}%
5619 {%
```

Yes, it has so do nothing.

```
5620 }%
5621 {%
```

No, it hasn't so add it now.

```
5622 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5623 }%
5624 }%
5625 {%
5626 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5627 }%
5628 }
```

```
glo@sortedinsert \@glo@sortedinsert{\<list>}{\<entry label>}
```

Insert into list.

```
5629 \newcommand*\@glo@sortedinsert}[2]{%
5630 \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5631 }%
```

The sort handlers need to be in the form required by datatool's `\dtl@sortlist` macro. These must set the count register `\dtl@sortresult` to either -1 ($\#1$ less than $\#2$), 0 ($\#1 = \#2$) or $+1$ ($\#1$ greater than $\#2$).

sorthandler@word

```
5632 \newcommand*\@glo@sorthandler@word}[2]{%
5633 \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
5634 \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
5635 \edef\@glo@do@compare{%
5636 \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5637 {\expandonce\@gls@sort@B}%
5638 {\expandonce\@gls@sort@A}%
5639 }%
5640 \@glo@do@compare
5641 }
```

thandler@letter

```
5642 \newcommand*\@glo@sorthandler@letter}[2]{%
5643 \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
5644 \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
5645 \edef\@glo@do@compare{%
5646 \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5647 {\expandonce\@gls@sort@B}%
5648 {\expandonce\@gls@sort@A}%
```

```

5649 }%
5650 \glo@do@compare
5651 }

```

sorthandler@case Case-sensitive sort.

```

5652 \newcommand*{\@glo@sorthandler@case}[2]{%
5653 \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5654 \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5655 \edef\glo@do@compare{%
5656 \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5657 {\expandonce\@gls@sort@B}%
5658 {\expandonce\@gls@sort@A}%
5659 }%
5660 \glo@do@compare
5661 }

```

sorthandler@nocase Case-insensitive sort.

```

5662 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5663 \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5664 \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5665 \edef\glo@do@compare{%
5666 \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5667 {\expandonce\@gls@sort@B}%
5668 {\expandonce\@gls@sort@A}%
5669 }%
5670 \glo@do@compare
5671 }

```

sortmacro@word Sort macro for ‘word’

```

5672 \newcommand*{\@glo@sortmacro@word}[1]{%
5673 \ifdefstring{\@glo@default@sorttype}{standard}%
5674 {%
5675 \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5676 }%
5677 {%
5678 \PackageError{glossaries}{Conflicting sort options:^^J
5679 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5680 \string\printnoidxglossary[sort=word]}{}%
5681 }%
5682 }

```

sortmacro@letter Sort macro for ‘letter’

```

5683 \newcommand*{\@glo@sortmacro@letter}[1]{%
5684 \ifdefstring{\@glo@default@sorttype}{standard}%
5685 {%
5686 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5687 }%
5688 {%
5689 \PackageError{glossaries}{Conflicting sort options:^^J

```

```

5690     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5691     \string\printnoidxglossary[sort=letter]}{}%
5692 }%
5693 }

```

`\tmacro@standard` Sort macro for 'standard'. (Use either 'word' or 'letter' order.)

```

5694 \newcommand*\@glo@sortmacro@standard}[1]{%
5695   \ifdefstring{\@glo@default@sorttype}{standard}%
5696   {%
5697     \ifcsdef{\@glo@sorthandler@glorder}%
5698     {%
5699       \@glo@sortentries{\csuse{\@glo@sorthandler@glorder}}{#1}%
5700     }%
5701     {%
5702       \PackageError{glossaries}{Unknown sort handler '\glorder'}{}}%
5703     }%
5704   }%
5705   {%
5706     \PackageError{glossaries}{Conflicting sort options:^^J
5707       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5708       \string\printnoidxglossary[sort=standard]}{}%
5709   }%
5710 }

```

`\sortmacro@case` Sort macro for 'case'

```

5711 \newcommand*\@glo@sortmacro@case}[1]{%
5712   \ifdefstring{\@glo@default@sorttype}{standard}%
5713   {%
5714     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5715   }%
5716   {%
5717     \PackageError{glossaries}{Conflicting sort options:^^J
5718       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5719       \string\printnoidxglossary[sort=case]}{}%
5720   }%
5721 }

```

`\ortmacro@nocase` Sort macro for 'nocase'

```

5722 \newcommand*\@glo@sortmacro@nocase}[1]{%
5723   \ifdefstring{\@glo@default@sorttype}{standard}%
5724   {%
5725     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5726   }%
5727   {%
5728     \PackageError{glossaries}{Conflicting sort options:^^J
5729       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5730       \string\printnoidxglossary[sort=nocase]}{}%
5731   }%
5732 }

```

o@sortmacro@def Sort macro for 'def'. The order of definition is given in \glo@list@{type}.

```
5733 \newcommand*{\@glo@sortmacro@def}[1]{%
5734 \def\@glo@sortinglist{}%
5735 \for@gl@sentries[#1]{\@gl@thislabel}%
5736 {%
5737 \xifinlistcs{\@gl@thislabel}{\@gl@ref@#1}%
5738 {%
5739 \list@add{\@glo@sortinglist}{\@gl@thislabel}%
5740 }%
5741 {%
    Hasn't been referenced.
5742 }%
5743 }%
5744 \cslet{\@gl@ref@#1}{\@glo@sortinglist}%
5745 }
```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5746 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5747 \ifinlistcs{#1}{\@gl@ref@\@glo@type}%
5748 {}%
5749 {%
5750 \list@csadd{\@gl@ref@\@glo@type}{#1}%
5751 }%
5752 \ifcsdef{\@glo@sortingchildren@#1}%
5753 {%
5754 \@glo@addchildren{\@glo@type}{#1}%
5755 }%
5756 {}%
5757 }
```

o@sortmacro@use Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5758 \newcommand*{\@glo@sortmacro@use}[1]{}%
```

@noidx@glossary Glossary handler for \printnoidxglossary which doesn't use an indexing application. Since \printnoidxglossary may occur at the start of the document, we can't just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```
5759 \newcommand*{\@print@noidx@glossary}{%
5760 \ifcsdef{\@gl@ref@\@glo@type}%
5761 {%
```

Sort the entries:

```
5762 \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5763 {%
5764 \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5765 }%
```

```

5766   {%
5767     \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
5768   }%

```

Do the glossary heading and preamble

```

5769   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5770   \glossarypreamble
5771   \begin{theglossary}%
5772   \glossaryheader
5773   \glsresetentrylist
5774   \def\@gls@currentlettergroup{}%

```

Iterate through the entries.

```

5775   \forlistcsloop{\@gls@noidx@do}{\@gls@ref@{\@glo@type}%

```

Finally end the glossary and do the postamble:

```

5776     \end{theglossary}%
5777     \glossarypostamble
5778   }%
5779   {%
5780     \@gls@noref@warn{\@glo@type}%
5781   }%
5782 }

```

\glo@grabfirst

```

5783 \def\glo@grabfirst#1#2\@nil{%
5784   \def\@gls@firsttok{#1}%
5785   \ifdefempty\@gls@firsttok
5786   {%
5787     \def\@glo@thislettergrp{0}%
5788   }%
5789   {%

```

Sanitize it:

```

5790     \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5791     \expandafter\@glo@grabfirst\@gls@firsttok{}{\@nil
5792   }%
5793 }

```

\@glo@grabfirst

```

5794 \def\@glo@grabfirst#1#2\@nil{%
5795   \ifdefempty\@glo@thislettergrp
5796   {%
5797     \def\@glo@thislettergrp{glssymbols}%
5798   }%
5799   {%
5800     \count@=\uccode‘#1\relax
5801     \ifnum\count@=0\relax
5802     \def\@glo@thislettergrp{glssymbols}%

```



```

5803 \else
5804 \ifdefstring\@glo@sorttype{case}%
5805 {%
5806 \count@=#1\relax
5807 }%
5808 {%
5809 }%
5810 \edef\@glo@thislettergrp{\the\count@}%
5811 \fi
5812 }%
5813 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label. This only allows one sublevel.

```

5814 \newcommand{\@gls@noidx@do}[1]{%
  Get this entry's location list
5815 \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  Does this entry have a parent?
5816 \ifglshasparent{#1}%
5817 {%
  Has a parent.
5818 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5819 \ifdefvoid{\@gls@loclist}
5820 {%
5821 \subglossentry{\gls@level}{#1}{}%
5822 }%
5823 {%
5824 \subglossentry{\gls@level}{#1}%
5825 }%
5826 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5827 }%
5828 }%
5829 }%
5830 {%
  Doesn't have a parent Get this entry's sort key
5831 \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
  Fetch the first letter:
5832 \expandafter\glo@grabfirst\@gls@sort{}{}@nil
5833 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5834 {}%
5835 {%
  Do the group header:
5836 \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
5837 \gls@groupheading{\@glo@thislettergrp}%
5838 }%
5839 \let\@gls@currentlettergroup\@glo@thislettergrp

```

Do this entry:

```
5840 \ifvoid{\@gls@loclist}
5841 {%
5842 \glossentry{#1}{}%
5843 }%
5844 {%
5845 \glossentry{#1}%
5846 {%
5847 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5848 }%
5849 }%
5850 }%
5851 }
```

```
\glsnoidxloclist \glsnoidxloclist{<list cs>}
```

Display location list.

```
5852 \newcommand*{\glsnoidxloclist}[1]{%
5853 \def\@gls@noidxloclist@sep{}}%
5854 \def\@gls@noidxloclist@prev{}}%
5855 \forlistloop{\glsnoidxloclisthandler}{#1}%
5856 }
```

`xloclisthandler` Handler for location list iterator.

```
5857 \newcommand*{\glsnoidxloclisthandler}[1]{%
5858 \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5859 {%
```

Same as previous location so skip.

```
5860 }%
5861 {%
5862 \@gls@noidxloclist@sep
5863 #1%
5864 \def\@gls@noidxloclist@sep{\delimN}%
5865 \def\@gls@noidxloclist@prev{#1}%
5866 }%
5867 }
```

`yloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```
5868 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
5869 \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5870 {%
```

Same as previous location so skip.

```
5871 }%
5872 {%
5873 \@gls@noidxloclist@sep
5874 \@gls@noidxloclist@prev
```

```

5875 \def\@gls@noidxloclist@prev{#1}%
5876 }%
5877 }

```

```
\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<location>}
```

Display a location in the location list.

```

5878 \newcommand*\glsnoidxdisplayloc[4]{%
5879 \setentrycounter[#1]{#2}%
5880 \csuse{#3}{#4}%
5881 }

```

```
\@gls@reference {<type>}{<label>}{<loc>}
```

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```
5882 \newcommand*\@gls@reference}[3]{%
```

Add to label list

```

5883 \glsdoifexistsorwarn{#2}%
5884 {%
5885 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}}%
5886 \ifinlistcs{#2}{@glsref@#1}%
5887 }%
5888 {\listcsgadd{@glsref@#1}{#2}}%

```

Add to location list

```

5889 \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
5890 {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}%
5891 }%
5892 \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
5893 }%
5894 }

```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
5895 \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`.

```

5896 \define@key{printgloss}{title}{%
5897 \def\glossarytitle{#1}%
5898 \let\gls@dotoc@title\relax
5899 }

```

The toctitle sets the text used for the relevant entry in the table of contents.

```
5900 \define@key{printgloss}{toctitle}{%
```

```

5901 \def\glossarytoctitle{#1}%
5902 \let\gls@dotoc\title\relax
5903 }

```

The `style` key sets the glossary style (but only for the given glossary).

```

5904 \define@key{printgloss}{style}{%
5905   \ifcsundef{@glsstyle@#1}%
5906   {%
5907     \PackageError{glossaries}%
5908     {Glossary style ‘#1’ undefined}{}%
5909   }%
5910   {%
5911     \def@glossarystyle{\setglossentrycompatibility
5912       \csname @glsstyle@#1\endcsname}%
5913   }%
5914 }

```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

5915 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5916 false,nolabel,autolabel,nameref}[nolabel]{%
5917   \ifcase\nr\relax
5918     \renewcommand*{\@glossarysecstar}{*}%
5919     \renewcommand*{\@glossaryseclabel}{}%
5920   \or
5921     \renewcommand*{\@glossarysecstar}{}%
5922     \renewcommand*{\@glossaryseclabel}{}%
5923   \or
5924     \renewcommand*{\@glossarysecstar}{*}%
5925     \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix@glo@type}}%
5926   \or
5927     \renewcommand*{\@glossarysecstar}{*}%
5928     \renewcommand*{\@glossaryseclabel}{%
5929       \protected@edef\@currentlabelname{\glossarytoctitle}%
5930       \label{\glsautoprefix@glo@type}}%
5931   \fi
5932 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

5933 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5934   \csuse{glsnogroupskip#1}%
5935 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

5936 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5937   \csuse{glsnopostdot#1}%
5938 }

```

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

5939 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5940   \csuse{glsentrycounter#1}%

```

```

5941 \ifglentrycounter
5942   \ifx\@gls@counterwithin\@empty
5943     \newcounter{glossaryentry}%
5944   \else
5945     \newcounter{glossaryentry}[\@gls@counterwithin]%
5946   \fi
5947   \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5948   \renewcommand*\glsresetentrycounter{%
5949     \setcounter{glossaryentry}{0}%
5950   }%
5951   \renewcommand*\glsstepentry}[1]{%
5952     \refstepcounter{glossaryentry}%
5953     \label{glsentry-\glsdetoklabel{##1}}%
5954   }%
5955   \renewcommand*\glsentrycounterlabel{\theglossaryentry.\space}%
5956   \renewcommand*\glsentryitem}[1]{%
5957     \glsstepentry{##1}\glsentrycounterlabel
5958   }%
5959 \else
5960   \renewcommand*\glsresetentrycounter{%
5961     \renewcommand*\glsstepentry}[1]{}%
5962     \renewcommand*\glsentrycounterlabel{}%
5963     \renewcommand*\glsentryitem}[1]{\glsresetsubentrycounter}
5964 \fi
5965 }

```

The subentrycounter key is the same as the package option but localised to the current glossary. Note that this doesn't affect the master/slave counter attributes, which occurs if subentrycounter and entrycounter package options are set to true.

```

5966 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5967   \csuse{glssubentrycounter#1}%
5968   \ifglssubentrycounter
5969     \ifundef\c@glossarysubentry
5970     {%
5971       \ifglentrycounter
5972         \newcounter{glossarysubentry}[glossaryentry]%
5973       \else
5974         \newcounter{glossarysubentry}
5975       \fi
5976     }{}%
5977     \renewcommand*\glsstepsubentry}[1]{%
5978       \edef\currentglssubentry{\glsdetoklabel{##1}}%
5979       \refstepcounter{glossarysubentry}%
5980       \label{glsentry-\currentglssubentry}%
5981     }%
5982     \renewcommand*\glsresetsubentrycounter{%
5983       \setcounter{glossarysubentry}{0}%
5984     }%
5985     \renewcommand*\glssubentryitem}[1]{%
5986       \glsstepsubentry{##1}\glssubentrycounterlabel

```

```

5987 }%
5988 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
5989 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5990 \else
5991 \renewcommand*{\glssubentryitem}[1]{}%
5992 \renewcommand*{\glsstepsubentry}[1]{}%
5993 \renewcommand*{\glsresetsubentrycounter}{}%
5994 \renewcommand*{\glssubentrycounterlabel}{}%
5995 \fi
5996 }

```

The nonumberlist key determines if this glossary should have a number list.

```

5997 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
5998 \ifglsnonumberlist
5999 \def\glossaryentrynumbers##1{}%
6000 \else
6001 \def\glossaryentrynumbers##1{##1}%
6002 \fi}

```

The sort key sets the glossary sort handler (`\printnoidxglossary` only).

```

6003 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`@assign@sortkey` Issue error if used with `\printglossary`

```

6004 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6005 \PackageError{glossaries}{‘sort’ key not permitted with
6006 \string\printglossary}%
6007 {The ‘sort’ key may only be used with \string\printnoidxglossary}%
6008 }

```

`@assign@sortkey` For use with `\printnoidxglossary`

```

6009 \newcommand*{\@glo@assign@sortkey}[1]{%
6010 \def\@glo@sorttype{#1}%
6011 }

```

`@glsnonextpages` 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 `\glsnonextpages` 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 re-defined.

```

6012 \newcommand*{\@glsnonextpages}{%
6013 \gdef\glossaryentrynumbers##1{%
6014 \glsresetentrylist
6015 }%
6016 }

```

`\@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 re-defined.

```
6017 \newcommand*{\@glsnextpages}{%
6018   \gdef\glossaryentrynumbers##1{%
6019     ##1\glsresetentrylist}}
```

`glsresetentrylist` Resets `\glossaryentrynumbers`

```
6020 \newcommand*{\glsresetentrylist}{%
6021   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}
```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```
6022 \newcommand*{\glsnonextpages}{}
```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```
6023 \newcommand*{\glsnextpages}{}
```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```
6024 \ifglentrycounter
6025   \ifx\@gls@counterwithin\@empty
6026     \newcounter{glossaryentry}
6027   \else
6028     \newcounter{glossaryentry}[\@gls@counterwithin]
6029   \fi
6030   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
6031 \fi
```

`glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```
6032 \ifglssubentrycounter
6033   \ifglentrycounter
6034     \newcounter{glossarysubentry}[glossaryentry]
6035   \else
6036     \newcounter{glossarysubentry}
6037   \fi
6038   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6039 \fi
```

`glsresetentrylist` Resets the `glossarysubentry` counter.

```
6040 \ifglssubentrycounter
6041   \newcommand*{\glsresetentrylist}{%
6042     \setcounter{glossarysubentry}{0}%
6043   }
6044 \else
6045   \newcommand*{\glsresetentrylist}{}
6046 \fi
```

subentrycounter Resets the glossaryentry counter.

```
6047 \ifglentrycounter
6048 \newcommand*\glsresetentrycounter{%
6049 \setcounter{glossaryentry}{0}%
6050 }
6051 \else
6052 \newcommand*\glsresetentrycounter{}
6053 \fi
```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```
6054 \ifglentrycounter
6055 \newcommand*\glsstepentry}[1]{%
6056 \refstepcounter{glossaryentry}%
6057 \label{glsentry-\glsdetoklabel{#1}}%
6058 }
6059 \else
6060 \newcommand*\glsstepentry}[1]{}
6061 \fi
```

glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
6062 \ifglssubentrycounter
6063 \newcommand*\glsstepsubentry}[1]{%
6064 \edef\currentglssubentry{\glsdetoklabel{#1}}%
6065 \refstepcounter{glossarysubentry}%
6066 \label{glsentry-\currentglssubentry}%
6067 }
6068 \else
6069 \newcommand*\glsstepsubentry}[1]{}
6070 \fi
```

\glsrefentry Reference the entry or sub-entry counter if in use, otherwise just do \gls.

```
6071 \ifglentrycounter
6072 \newcommand*\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
6073 \else
6074 \ifglssubentrycounter
6075 \newcommand*\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
6076 \else
6077 \newcommand*\glsrefentry}[1]{\gls{#1}}
6078 \fi
6079 \fi
```

trycounterlabel Defines how to display the glossaryentry counter.

```
6080 \ifglentrycounter
6081 \newcommand*\glsentrycounterlabel{\theglossaryentry.\space}
6082 \else
6083 \newcommand*\glsentrycounterlabel{}
6084 \fi
```


`trycounterlabel` Defines how to display the `glossarysubentry` counter.

```
6085 \ifglssubentrycounter
6086   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}
6087 \else
6088   \newcommand*{\glssubentrycounterlabel}{}
6089 \fi
```

`\glstentryitem` Step and display `glossaryentry` counter, if appropriate.

```
6090 \ifglstentrycounter
6091   \newcommand*{\glstentryitem}[1]{%
6092     \glststepentry{#1}\glstentrycounterlabel
6093   }
6094 \else
6095   \newcommand*{\glstentryitem}[1]{\glstresetsubentrycounter}
6096 \fi
```

`glssubentryitem` Step and display `glossarysubentry` counter, if appropriate.

```
6097 \ifglssubentrycounter
6098   \newcommand*{\glssubentryitem}[1]{%
6099     \glststepsubentry{#1}\glssubentrycounterlabel
6100   }
6101 \else
6102   \newcommand*{\glssubentryitem}[1]{}
6103 \fi
```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
6104 \ifcsundef{theglossary}%
6105 {%
6106   \newenvironment{theglossary}{}{}%
6107 }%
6108 {%
6109   \@gls@warnontheglossdefined
6110   \renewenvironment{theglossary}{}{}%
6111 }
```

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`

```
6112 \newcommand*{\glossaryheader}{}
```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
6113 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#1}{#2}}
```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

```
\compatibleglossentry \glossentry{<label>}{<page-list>}
```

```
6114 \providecommand*{\compatibleglossentry}[2]{%
6115   \toks@{#2}%
6116   \protected@edef\do@glossentry{\noexpand\glossaryentryfield{#1}%
6117     {\noexpand\glsnamefont
6118       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6119     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6120     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6121     {\the\toks@}}%
6122   }%
6123   \@do@glossentry
6124 }
```

```
\glossentryname
```

```
6125 \newcommand*{\glossentryname}[1]{%
6126   \glsdoifexistsorwarn{#1}%
6127   {%
6128     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6129     \expandafter\glsnamefont\expandafter{\glo@name}%
6130   }%
6131 }
```

```
\Glossentryname
```

```
6132 \newcommand*{\Glossentryname}[1]{%
6133   \glsdoifexistsorwarn{#1}%
6134   {%
6135     \glsnamefont{\Glsentryname{#1}}%
6136   }%
6137 }
```

```
\glossentrydesc
```

```
6138 \newcommand*{\glossentrydesc}[1]{%
6139   \glsdoifexistsorwarn{#1}%
6140   {%
6141     \glsentrydesc{#1}%
6142   }%
6143 }
```

```
\Glossentrydesc
```

```

6144 \newcommand*\Glossentrydesc}[1]{%
6145   \glsdoifexistsorwarn{#1}%
6146   {%
6147     \Glsentrydesc{#1}%
6148   }%
6149 }

```

lossentrysymbol

```

6150 \newcommand*\glossentrysymbol}[1]{%
6151   \glsdoifexistsorwarn{#1}%
6152   {%
6153     \glsentrysymbol{#1}%
6154   }%
6155 }

```

lossentrysymbol

```

6156 \newcommand*\Glossentrysymbol}[1]{%
6157   \glsdoifexistsorwarn{#1}%
6158   {%
6159     \Glsentrysymbol{#1}%
6160   }%
6161 }

```

blesubglossentry

```
\subglossentry{<level>}{<label>}{<page-list>}
```

```

6162 \providecommand*\compatiblesubglossentry}[3]{%
6163   \toks@{#3}%
6164   \protected@edef\do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6165     {#2}%
6166     {\noexpand\glsnamefont
6167       {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
6168     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6169     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6170     {\the\toks@}%
6171   }%
6172   \@do@subglossentry
6173 }

```

rycompatibility

```

6174 \newcommand*\setglossentrycompatibility}{%
6175   \let\glossentry\compatibleglossentry
6176   \let\subglossentry\compatiblesubglossentry
6177 }
6178 \setglossentrycompatibility

```

ossaryentryfield

```
\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```
6179 \newcommand{\glossaryentryfield}[5]{%
6180   \GlossariesWarning
6181   {Deprecated use of \string\glossaryentryfield.^^J
6182    I recommend you change to \string\glossentry.^^J
6183    If you've just upgraded, try removing your gls auxiliary
6184    files^^J and recompile}%
6185   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

glossarysubentryfield

```
\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.)

```
6186 \newcommand*{\glossarysubentryfield}[6]{%
6187   \GlossariesWarning
6188   {Deprecated use of \string\glossarysubentryfield.^^J
6189    I recommend you change to \string\subglossentry.^^J
6190    If you've just upgraded, try removing your gls auxiliary
6191    files^^J and recompile}%
6192   \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`

```
6193 \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: `glsymbols`, `glsnumbers`, A, ..., Z. Glossary styles must be redefined this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
6194 \newcommand*{\glsgroupheading}[1]{} 
```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

```
\glsgetgrouptitle{<label>}
```

This command produces the title for the glossary group whose label is given by `<label>`. By default, the group labelled `glsymbols` produces `\glsymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glsymbols`, `glsnumbers`, `A`, ..., `Z`. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

`lsgetgrouptitle`

```
6195 \newcommand*{\glsgetgrouptitle}[1]{%
6196   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6197   \@gls@grptitle
6198 }
```

`s@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6199 \newcommand*{\@gls@getgrouptitle}[2]{%
    Even if the argument appears to be a single letter, it won't be considered a single letter by
    \dtl@ifsingle if it's an active character.
6200 \dtl@ifsingle{#1}%
6201 {%
6202   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6203 }%
6204 {%
6205   \ifboolexpr{test{\ifstrequal{#1}{glsymbols}}
6206               or test{\ifstrequal{#1}{glsnumbers}}}%
6207   {%
6208     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6209   }%
6210   {%
6211     \def#2{#1}%
6212   }%
6213 }%
6214 }
```

`othergrouptitle` Version for the no-indexing app option:

```

6215 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6216   \DTLifint{#1}%
6217   {\edef#2{\char#1\relax}}%
6218   {%
6219     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6220   }%
6221 }

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`\glsgetgrouplabel`

```

6222 \newcommand*{\glsgetgrouplabel}[1]{%
6223   \ifthenelse{equal{#1}{\glsymbolsgroupname}}{\glsymbols}{%
6224   \ifthenelse{equal{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}

```

The command `\setentrycounter` sets the entry's associated counter (required by `\glsnumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`\setentrycounter`

```

6225 \newcommand*{\setentrycounter}[2] []{%
6226   \def\@glo@counterprefix{#1}%
6227   \ifx\@glo@counterprefix\@empty
6228     \def\@glo@counterprefix{.}%
6229   \else
6230     \def\@glo@counterprefix{.#1.}%
6231   \fi
6232   \def\glsentrycounter{#2}%
6233 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6234 \newcommand*{\setglossarystyle}[1]{%
6235   \ifcsundef{@glsstyle@#1}%
6236   {%
6237     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6238   }%
6239   {%
6240     \csname @glsstyle@#1\endcsname
6241   }%

```

Set the default style if it's not already set.

```

6242   \ifx\@glossary@default@style\relax
6243     \protected@edef\@glossary@default@style{#1}%

```

```
6244 \fi
6245 }
```

`\glossarystyle`

```
6246 \newcommand*{\glossarystyle}[1]{%
6247   \ifcsundef{@glsstyle@#1}%
6248   {%
6249     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6250   }%
6251   {%
6252     \GlossariesWarning
6253     {Deprecated command \string\glossarystyle.^^J
6254     I recommend you switch to \string\setglossarystyle\space unless
6255     you want to maintain backward compatibility}%
6256     \setglossentrycompatibility
6257     \csname @glsstyle@#1\endcsname

6258     \ifcsdef{@glscompstyle@#1}%
6259     {\setglossentrycompatibility\cuse{@glscompstyle@#1}}%
6260   }%
6261 }%
```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```
6262 \ifx@glossary@default@style\relax
6263   \protected@edef@glossary@default@style{#1}%
6264 \fi
6265 }
```

`\ewglossarystyle` New glossary styles can be defined using:

```
\ewglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [section 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossary preamble` and `\glossary postamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```
6266 \newcommand{\ewglossarystyle}[2]{%
6267   \ifcsundef{@glsstyle@#1}%
6268   {%
6269     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6270   }%
6271   {%
6272     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6273   }%
6274 }
```

`\ewglossarystyle` Code for this macro supplied by Marco Daniel.

```

6275 \newcommand{\renewglossarystyle}[2]{%
6276   \ifcsundef{@glsstyle@#1}%
6277   {%
6278     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6279   }%
6280   {%
6281     \csdef{@glsstyle@#1}{#2}%
6282   }%
6283 }

```

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`

```

6284 \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`

```

6285 \ifcsundef{hyperlink}%
6286 {%
6287   \def\glshypernumber#1{#1}%
6288 }%
6289 {%
6290   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6291 }

```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

6292 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6293   \ifx\#1\%
6294   \else
6295     \@delimR#1\delimR\delimR\%
6296   \fi
6297   \ifx\#2\%

```



```

6298 \else
6299   #2%
6300 \fi
6301 \ifx\#3\%
6302 \else
6303   \@glshypernumber#3\@nil
6304 \fi
6305 }

```

`\@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`

```

6306 \def\@delimR#1\delimR #2\delimR #3\%
6307 \ifx\#2\%
6308   \@delimN{#1}%
6309 \else
6310   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6311 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

6312 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6313 \def\@@delimN#1\delimN #2\delimN#3\%
6314 \ifx\#3\%
6315   \@gls@numberlink{#1}%
6316 \else
6317   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6318 \fi
6319 }

```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

6320 \def\@gls@numberlink#1{%
6321 \begingroup
6322 \toks@={}%
6323 \@gls@removespaces#1 \@nil
6324 \endgroup}

6325 \def\@gls@removespaces#1 #2\@nil{%
6326 \toks@=\expandafter{\the\toks@#1}%
6327 \ifx\#2\%
6328   \edef\x{\the\toks@}%
6329   \ifx\x\empty
6330     \else

6331     \hyperlink{\glstentrycounter\@gls@counterprefix\the\toks@}%
6332     {\the\toks@}%
6333 \fi

```

```

6334 \else
6335   \@gls@ReturnAfterFi{%
6336     \@gls@removespaces#2\@nil
6337   }%
6338 \fi
6339 }
6340 \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`

```
6341 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}
```

`\hypersf`

```
6342 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}
```

`\hypertt`

```
6343 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}
```

`\hyperbf`

```
6344 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}
```

`\hypermd`

```
6345 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}
```

`\hyperit`

```
6346 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6347 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6348 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6349 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6350 \newcommand*\hyperemph[1]{\emph{\glshypernumber{#1}}}
```

1.17 Acronyms

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

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}` and it additionally defines the command `\⟨label⟩` which is equivalent to `\gls{⟨label⟩}` (thus `⟨label⟩` must only contain alphabetical characters). If `⟨label⟩` is omitted, `⟨abbrv⟩` 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 `\⟨label⟩` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\⟨label⟩[⟨insert⟩]` but you can't do `\⟨label⟩[⟨key-val list⟩]`. 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.

```
6351 \newcommand{\oldacronym}[4][\gls@label]{%
6352   \def\gls@label{#2}%
6353   \newacronym[#4]{#1}{#2}{#3}%
6354   \ifcsundef{xspace}%
6355     {%
6356       \expandafter\edef\csname#1\endcsname{%
6357         \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6358     }%
6359   }%
6360   {%
6361     \expandafter\edef\csname#1\endcsname{%
6362       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6363         \noexpand\gls{#1}\noexpand\xspace}%
6364     }%
6365   }%
6366 }
```

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

This is a quick way of defining acronyms, using `\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
6367 \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 need to cancel it out.

```
6368 \newcommand*\acrpluralsuffix{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6369 \newrobustcmd*\glstextup[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6370 \newcommand*\glsshortkey{short}
```

`\glsshortpluralkey`

```
6371 \newcommand*\glsshortpluralkey{shortplural}
```

`\glslongkey`

```
6372 \newcommand*\glslongkey{long}
```

`\glslongpluralkey`

```
6373 \newcommand*\glslongpluralkey{longplural}
```

`\acrfull` Full form of the acronym.

```
6374 \newrobustcmd*\acrfull{\@gls@hyp@opt\ns@acrfull}
```

```
6375 \newcommand*\ns@acrfull[2][\]{%
```

```
6376 \new@ifnextchar[\@acrfull{#1}{#2}}%
```

```
6377 \@acrfull{#1}{#2}[\]}%
```

```
6378 }
```

`\@acrfull` Low-level macro:

```
6379 \def\@acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6380 \acrfullfmt{#1}{#2}{#3}%
```

```
6381 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

`\acrfullfmt` No case change full format.

```
6382 \newcommand*\acrfullfmt}[3]{%
6383   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6384 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<in`

```
6385 \newcommand*\acrlinkfullformat}[5]{%
6386   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6387 }
```

`\acrfullformat` Default full form is *<long>* (*<short>*).

```
6388 \newcommand*\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6389 \newrobustcmd*\glsspace{\space}
```

Default format for full acronym

`\Acrfull`

```
6390 \newrobustcmd*\Acrfull{\@gls@hyp@opt\ns@Acrfull}
6391 \newcommand*\ns@Acrfull[2][]{%
6392   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6393     {\@Acrfull{#1}{#2}[]}%
6394 }
```

Low-level macro:

```
6395 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6396   \Acrfullfmt{#1}{#2}{#3}%
6397 }
```

`\Acrfullfmt` First letter upper case full format.

```
6398 \newcommand*\Acrfullfmt}[3]{%
6399   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6400 }
```

`\ACRfull`

```
6401 \newrobustcmd*\ACRfull{\@gls@hyp@opt\ns@ACRfull}
6402 \newcommand*\ns@ACRfull[2][]{%
6403   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6404     {\@ACRfull{#1}{#2}[]}%
6405 }
```

Low-level macro:

```
6406 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6407 \ACRfullfmt{#1}{#2}{#3}%  
6408 }
```

\ACRfullfmt All upper case full format.

```
6409 \newcommand*\ACRfullfmt [3] {%  
6410 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6411 }
```

Plural:

\acrfullpl

```
6412 \newrobustcmd*\acrfullpl{\@gls@hyp@opt\ns@acrfullpl}  
  
6413 \newcommand*\ns@acrfullpl [2] [] {%  
6414 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6415 {\@acrfullpl{#1}{#2} []}%  
6416 }
```

Low-level macro:

```
6417 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6418 \acrfullplfmt{#1}{#2}{#3}%  
6419 }
```

\acrfullplfmt No case change plural full format.

```
6420 \newcommand*\acrfullplfmt [3] {%  
6421 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6422 }
```

\Acrfullpl

```
6423 \newrobustcmd*\Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}  
  
6424 \newcommand*\ns@Acrfullpl [2] [] {%  
6425 \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6426 {\@Acrfullpl{#1}{#2} []}%  
6427 }
```

Low-level macro:

```
6428 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6429 \Acrfullplfmt{#1}{#2}{#3}%  
6430 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6431 \newcommand*\Acrfullplfmt [3] {%  
6432 \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6433 }
```

```

\ACRfullpl
6434 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}

6435 \newcommand*\ns@ACRfullpl [2] [] {%
6436   \new@ifnextchar [{\@ACRfullpl{#1}{#2}}%
6437     {\@ACRfullpl{#1}{#2} []}%
6438 }

```

Low-level macro:

```

6439 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6440   \ACRfullplfmt{#1}{#2}{#3}%
6441 }

```

```

\ACRfullplfmt  All upper case plural full format.
6442 \newcommand*\ACRfullplfmt [3] {%
6443   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6444 }

```

1.18 Predefined acronym styles

```

\acronymfont  This is only used with the additional acronym styles:
6445 \newcommand{\acronymfont} [1] {#1}

```

```

\firstacronymfont  This is only used with the additional acronym styles:
6446 \newcommand{\firstacronymfont} [1] {\acronymfont{#1}}

```

```

\acrnameformat  The styles that allow an additional description use \acrnameformat{<short>}{<long>} to de-
  termine what information is displayed in the name.
6447 \newcommand*\acrnameformat [2] {\acronymfont{#1}}

```

Define some tokens used by \newacronym:

```

\glskeylisttok
6448 \newtoks\glskeylisttok

```

```

\glslabeltok
6449 \newtoks\glslabeltok

```

```

\glsshorttok
6450 \newtoks\glsshorttok

```

```

\gslongtok
6451 \newtoks\gslongtok

```

```

\newacronymhook  Provide a hook for \newacronym:
6452 \newcommand*\newacronymhook{}

```

GenericNewAcronym New improved version of setting the acronym style.

```
6453 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc

```
6454 \let\@Gls@entryname\@Gls@acentryname
```

Change the way acronyms are defined:

```
6455 \renewcommand{\newacronym}[4][{}]{%
```

```
6456 \ifdefempty{\@glsacronymlists}%
```

```
6457 {%
```

```
6458 \def\@glo@type{\acronymtype}%
```

```
6459 \setkeys{glossentry}{##1}%
```

```
6460 \DeclareAcronymList{\@glo@type}%
```

```
6461 }%
```

```
6462 {}%
```

```
6463 \glskeylisttok{##1}%
```

```
6464 \glslabeltok{##2}%
```

```
6465 \glsshorttok{##3}%
```

```
6466 \glslongtok{##4}%
```

```
6467 \newacronymhook
```

```
6468 \protected@edef\@do@newglossaryentry{%
```

```
6469 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6470 {%
```

```
6471 type=\acronymtype,%
```

```
6472 name={\expandonce{\acronymentry{##2}}},%
```

```
6473 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
```

```
6474 text={\the\glsshorttok},%
```

```
6475 short={\the\glsshorttok},%
```

```
6476 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
```

```
6477 long={\the\glslongtok},%
```

```
6478 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
```

```
6479 \GenericAcronymFields,%
```

```
6480 \the\glskeylisttok
```

```
6481 }%
```

```
6482 }%
```

```
6483 \@do@newglossaryentry
```

```
6484 }%
```

Make sure that \acrfull etc reflects the new style:

```
6485 \renewcommand*{\acrfullfmt}[3]{%
```

```
6486 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
```

```
6487 \renewcommand*{\Acrfullfmt}[3]{%
```

```
6488 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
```

```
6489 \renewcommand*{\ACRfullfmt}[3]{%
```

```
6490 \glslink[##1]{##2}{%
```

```
6491 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
```

```
6492 \renewcommand*{\acrfullplfmt}[3]{%
```

```
6493 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
```

```
6494 \renewcommand*{\Acrfullplfmt}[3]{%
```



```

6495 \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}%
6496 \renewcommand*{\ACRfullplfmt}[3]{%
6497 \glslink{##1}{##2}{%
6498 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6499 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6500 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6501 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6502 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6503 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6504 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

```

\acronymentry \acronymentry{<label>}

```

Display style for the name field in the list of acronyms.

```

6505 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

```

\acronymsort \acronymsort{<short>}{<long>}

```

Default sort format for acronyms.

```

6506 \newcommand*{\acronymsort}[2]{#1}

```

```

\setacronymstyle \setacronymstyle{<style name>}

```

```

6507 \newcommand*{\setacronymstyle}[1]{%
6508 \ifcsundef{@glsacr@dispstyle@#1}
6509 {%
6510 \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
6511 }%
6512 {%
6513 \ifdefempty{\@glsacronymlists}%
6514 {%
6515 \DeclareAcronymList{\acronymtype}%
6516 }%
6517 }%
6518 \SetGenericNewAcronym
6519 \GlsUseAcrStyleDefs{#1}%
6520 \@for\@gls@type:=\@glsacronymlists\do{%
6521 \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6522 }%
6523 }%
6524 }

```

```
\newacronymstyle \newacronymstyle{<style name>}{<entry format definition>}{<display
definitions>}
```

Defines a new acronym style called *<style name>*.

```
6525 \newcommand*{\newacronymstyle}[3]{%
6526   \ifcsdef{@glsacr@dispstyle@#1}%
6527   {%
6528     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6529   }%
6530   {%
6531     \csdef{@glsacr@dispstyle@#1}{#2}%
6532     \csdef{@glsacr@styledefs@#1}{#3}%
6533   }%
6534 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6535 \newcommand*{\renewacronymstyle}[3]{%
6536   \ifcsdef{@glsacr@dispstyle@#1}%
6537   {%
6538     \csdef{@glsacr@dispstyle@#1}{#2}%
6539     \csdef{@glsacr@styledefs@#1}{#3}%
6540   }%
6541   {%
6542     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6543   }%
6544 }
```

`\GlsUseAcrEntryDispStyle`

```
6545 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

`\GlsUseAcrStyleDefs`

```
6546 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```
6547 \newacronymstyle{long-short}%
6548 {%
```

Check for long form in case this is a mixed glossary.

```
6549   \ifglshaslong{glslabel}{glsacrfmt}{glsacrfmt}%
6550 }%
6551 {%
6552   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6553   \renewcommand*{\genacrfullformat}[2]{%
6554     \glsentrylong{##1}##2\space
6555     (\protect\firstacronymfont{\glsentryshort{##1}})%
6556   }%
6557   \renewcommand*{\Genacrfullformat}[2]{%
```

```

6558 \Glsentrylong{##1}##2\space
6559 (\protect\firstacronymfont{\glsentryshort{##1}})%
6560 }%
6561 \renewcommand*\genplacrfullformat}[2]{%
6562 \glsentrylongpl{##1}##2\space
6563 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6564 }%
6565 \renewcommand*\Genplacrfullformat}[2]{%
6566 \Glsentrylongpl{##1}##2\space
6567 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6568 }%
6569 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6570 \renewcommand*\acronymsort}[2]{##1}%
6571 \renewcommand*\acronymfont}[1]{##1}%
6572 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6573 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6574 }

```

`long-sp-short` Similar to the previous style but allows the space between the long and short form to be customized.

```

6575 \newacronymstyle{long-sp-short}%
6576 {%

```

Check for long form in case this is a mixed glossary.

```

6577 \ifglshaslong{\glslabel}{\glsacacfmt}{\glsacacentryfmt}%
6578 }%
6579 {%
6580 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
6581 \renewcommand*\genacrfullformat}[2]{%
6582 \glsentrylong{##1}##2\glsacspace{##1}%
6583 (\protect\firstacronymfont{\glsentryshort{##1}})%
6584 }%
6585 \renewcommand*\Genacrfullformat}[2]{%
6586 \Glsentrylong{##1}##2\glsacspace{##1}%
6587 (\protect\firstacronymfont{\glsentryshort{##1}})%
6588 }%
6589 \renewcommand*\genplacrfullformat}[2]{%
6590 \glsentrylongpl{##1}##2\glsacspace{##1}%
6591 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6592 }%
6593 \renewcommand*\Genplacrfullformat}[2]{%
6594 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6595 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6596 }%
6597 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6598 \renewcommand*\acronymsort}[2]{##1}%
6599 \renewcommand*\acronymfont}[1]{##1}%
6600 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6601 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6602 }

```

`\glsacspace` Space between long and short form for the above style. This uses a non-breakable space if the short form is less than 3em, otherwise it uses a regular space.

```
6603 \newcommand*{\glsacspace}[1]{%
6604   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{##1}})}%
6605   \ifdim\dimen@<3em~\else\space\fi
6606 }
```

`short-long` (*short*) (*long*) acronym style.

```
6607 \newacronymstyle{short-long}%
6608 {%
```

Check for long form in case this is a mixed glossary.

```
6609   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6610 }%
6611 {%
6612   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6613   \renewcommand*{\genacrfullformat}[2]{%
6614     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6615     (\glsentrylong{##1})%
6616   }%
6617   \renewcommand*{\Genacrfullformat}[2]{%
6618     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6619     (\glsentrylong{##1})%
6620   }%
6621   \renewcommand*{\genplacrfullformat}[2]{%
6622     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6623     (\glsentrylongpl{##1})%
6624   }%
6625   \renewcommand*{\Genplacrfullformat}[2]{%
6626     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6627     (\glsentrylongpl{##1})%
6628   }%
6629   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6630   \renewcommand*{\acronymsort}[2]{##1}%
6631   \renewcommand*{\acronymfont}[1]{##1}%
6632   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6633   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6634 }
```

`long-sc-short` (*long*) (`\textsc{short}`) acronym style.

```
6635 \newacronymstyle{long-sc-short}%
6636 {%
6637   \GlsUseAcrEntryDispStyle{long-short}%
6638 }%
6639 {%
6640   \GlsUseAcrStyleDefs{long-short}%
6641   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6642   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6643 }
```

long-sm-short *<long>* (*\textsmaller{<short>}*) acronym style.

```
6644 \newacronymstyle{long-sm-short}%
6645 {%
6646   \GlsUseAcrEntryDispStyle{long-short}%
6647 }%
6648 {%
6649   \GlsUseAcrStyleDefs{long-short}%
6650   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6651   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6652 }
```

sc-short-long *<short>* (*\textsc{<long>}*) acronym style.

```
6653 \newacronymstyle{sc-short-long}%
6654 {%
6655   \GlsUseAcrEntryDispStyle{short-long}%
6656 }%
6657 {%
6658   \GlsUseAcrStyleDefs{short-long}%
6659   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6660   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6661 }
```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```
6662 \newacronymstyle{sm-short-long}%
6663 {%
6664   \GlsUseAcrEntryDispStyle{short-long}%
6665 }%
6666 {%
6667   \GlsUseAcrStyleDefs{short-long}%
6668   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6669   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6670 }
```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```
6671 \newacronymstyle{long-short-desc}%
6672 {%
6673   \GlsUseAcrEntryDispStyle{long-short}%
6674 }%
6675 {%
6676   \GlsUseAcrStyleDefs{long-short}%
6677   \renewcommand*{\GenericAcronymFields}{}%
6678   \renewcommand*{\acronymsort}[2]{##2}%
6679   \renewcommand*{\acronymentry}[1]{%
6680     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6681 }
```

g-sp-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply). The space between the long and short form is given by *\glsacspace*.

```

6682 \newacronymstyle{long-sp-short-desc}%
6683 {%
6684   \GlsUseAcrEntryDispStyle{long-sp-short}%
6685 }%
6686 {%
6687   \GlsUseAcrStyleDefs{long-sp-short}%
6688   \renewcommand*{\GenericAcronymFields}{}%
6689   \renewcommand*{\acronymsort}[2]{##2}%
6690   \renewcommand*{\acronymentry}[1]{%
6691     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6692 }

```

`g-sc-short-desc` *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6693 \newacronymstyle{long-sc-short-desc}%
6694 {%
6695   \GlsUseAcrEntryDispStyle{long-sc-short}%
6696 }%
6697 {%
6698   \GlsUseAcrStyleDefs{long-sc-short}%
6699   \renewcommand*{\GenericAcronymFields}{}%
6700   \renewcommand*{\acronymsort}[2]{##2}%
6701   \renewcommand*{\acronymentry}[1]{%
6702     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6703 }

```

`g-sm-short-desc` *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6704 \newacronymstyle{long-sm-short-desc}%
6705 {%
6706   \GlsUseAcrEntryDispStyle{long-sm-short}%
6707 }%
6708 {%
6709   \GlsUseAcrStyleDefs{long-sm-short}%
6710   \renewcommand*{\GenericAcronymFields}{}%
6711   \renewcommand*{\acronymsort}[2]{##2}%
6712   \renewcommand*{\acronymentry}[1]{%
6713     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6714 }

```

`short-long-desc` *<short>* (`{<long>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6715 \newacronymstyle{short-long-desc}%
6716 {%
6717   \GlsUseAcrEntryDispStyle{short-long}%
6718 }%
6719 {%
6720   \GlsUseAcrStyleDefs{short-long}%
6721   \renewcommand*{\GenericAcronymFields}{}%

```

```

6722 \renewcommand*\acronymsort}[2]{##2}%
6723 \renewcommand*\acronymentry}[1]{%
6724   \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6725 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6726 \newacronymstyle{sc-short-long-desc}%
6727 {%
6728   \GlsUseAcrEntryDispStyle{sc-short-long}%
6729 }%
6730 {%
6731   \GlsUseAcrStyleDefs{sc-short-long}%
6732   \renewcommand*\GenericAcronymFields{}%
6733   \renewcommand*\acronymsort}[2]{##2}%
6734   \renewcommand*\acronymentry}[1]{%
6735     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6736 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6737 \newacronymstyle{sm-short-long-desc}%
6738 {%
6739   \GlsUseAcrEntryDispStyle{sm-short-long}%
6740 }%
6741 {%
6742   \GlsUseAcrStyleDefs{sm-short-long}%
6743   \renewcommand*\GenericAcronymFields{}%
6744   \renewcommand*\acronymsort}[2]{##2}%
6745   \renewcommand*\acronymentry}[1]{%
6746     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6747 }

```

dua *<long>* only acronym style.

```

6748 \newacronymstyle{dua}%
6749 {%

```

Check for long form in case this is a mixed glossary.

```

6750 \ifdefempty\glscustomtext
6751   {%
6752     \ifglshaslong{\glslabel}%
6753     {%
6754       \glsifplural
6755       {%

```

Plural form:

```

6756     \glscapscase
6757     {%

```

Plural form, don't adjust case:

```
6758      \glsentrylongpl{\glslabel}\glsinsert
6759      }%
6760      {%
```

Plural form, make first letter upper case:

```
6761      \Glsentrylongpl{\glslabel}\glsinsert
6762      }%
6763      {%
```

Plural form, all caps:

```
6764      \mfirstucMakeUppercase
6765      {\glsentrylongpl{\glslabel}\glsinsert}%
6766      }%
6767      }%
6768      {%
```

Singular form

```
6769      \glscapscase
6770      {%
```

Singular form, don't adjust case:

```
6771      \glsentrylong{\glslabel}\glsinsert
6772      }%
6773      {%
```

Subsequent singular form, make first letter upper case:

```
6774      \Glsentrylong{\glslabel}\glsinsert
6775      }%
6776      {%
```

Subsequent singular form, all caps:

```
6777      \mfirstucMakeUppercase
6778      {\glsentrylong{\glslabel}\glsinsert}%
6779      }%
6780      }%
6781      }%
6782      {%
```

Not an acronym:

```
6783      \glsgenentryfmt
6784      }%
6785      }%
6786      {\glscustomtext\glsinsert}%
6787      }%
6788      {%
6789      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6790      \renewcommand*{\acrfullfmt}[3]{%
6791      \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6792      (\acronymfont{\glsentryshort{##2}})}}%
6793      \renewcommand*{\Acrfullfmt}[3]{%
```



```

6794 \glslink{##1}{##2}{\Glsentrylong{##2}##3\space
6795 (\acronymfont{\glsentryshort{##2}})}%
6796 \renewcommand*{\ACRfullfmt}[3]{%
6797 \glslink{##1}{##2}{%
6798 \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6799 (\acronymfont{\glsentryshort{##2}})}%

6800 \renewcommand*{\acrfullplfmt}[3]{%
6801 \glslink{##1}{##2}{\glsentrylongpl{##2}##3\space
6802 (\acronymfont{\glsentryshortpl{##2}})}%

6803 \renewcommand*{\Acrfullplfmt}[3]{%
6804 \glslink{##1}{##2}{\Glsentrylongpl{##2}##3\space
6805 (\acronymfont{\glsentryshortpl{##2}})}%
6806 \renewcommand*{\ACRfullplfmt}[3]{%
6807 \glslink{##1}{##2}{%
6808 \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6809 (\acronymfont{\glsentryshortpl{##2}})}%
6810 \renewcommand*{\glsentryfull}[1]{%
6811 \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6812 }%
6813 \renewcommand*{\Glsentryfull}[1]{%
6814 \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6815 }%
6816 \renewcommand*{\glsentryfullpl}[1]{%
6817 \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6818 }%
6819 \renewcommand*{\Glsentryfullpl}[1]{%
6820 \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6821 }%
6822 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6823 \renewcommand*{\acronymsort}[2]{##1}%
6824 \renewcommand*{\acronymfont}[1]{##1}%
6825 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6826 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6827 \newacronymstyle{dua-desc}%
6828 {%
6829 \GlsUseAcrEntryDispStyle{dua}%
6830 }%
6831 {%
6832 \GlsUseAcrStyleDefs{dua}%
6833 \renewcommand*{\GenericAcronymFields}{}%

6834 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6835 \renewcommand*{\acronymsort}[2]{##2}%
6836 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```
6837 \newacronymstyle{footnote}%  
6838 {%
```

Check for long form in case this is a mixed glossary.

```
6839 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%  
6840 }%  
6841 {%  
6842 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
6843 \glshyperfirstfalse  
6844 \renewcommand*{\genacrfullformat}[2]{%  
6845 \protect\firstacronymfont{\glsenentryshort{##1}}##2%  
6846 \protect\footnote{\glsenentrylong{##1}}%  
6847 }%  
6848 \renewcommand*{\Genacrfullformat}[2]{%  
6849 \firstacronymfont{\Glsenentryshort{##1}}##2%  
6850 \protect\footnote{\glsenentrylong{##1}}%  
6851 }%  
6852 \renewcommand*{\genplacrfullformat}[2]{%  
6853 \protect\firstacronymfont{\glsenentryshortpl{##1}}##2%  
6854 \protect\footnote{\glsenentrylongpl{##1}}%  
6855 }%  
6856 \renewcommand*{\Genplacrfullformat}[2]{%  
6857 \protect\firstacronymfont{\Glsenentryshortpl{##1}}##2%  
6858 \protect\footnote{\glsenentrylongpl{##1}}%  
6859 }%  
6860 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsenentryshort{##1}}}%  
6861 \renewcommand*{\acronymsort}[2]{##1}%  
6862 \renewcommand*{\acronymfont}[1]{##1}%  
6863 \renewcommand*{\acrpluralsuffix}{\glssacrpluralsuffix}%
```

Don't use footnotes for \acrfull:

```
6864 \renewcommand*{\acrfullfmt}[3]{%  
6865 \glslink[##1]{##2}{\acronymfont{\glsenentryshort{##2}}##3\space  
6866 (\glsenentrylong{##2})}%  
6867 \renewcommand*{\Acrfullfmt}[3]{%  
6868 \glslink[##1]{##2}{\acronymfont{\Glsenentryshort{##2}}##3\space  
6869 (\glsenentrylong{##2})}%  
6870 \renewcommand*{\ACRfullfmt}[3]{%  
6871 \glslink[##1]{##2}{%  
6872 \mfirstucMakeUppercase{\acronymfont{\glsenentryshort{##2}}##3\space  
6873 (\glsenentrylong{##2})}}}%  
6874 \renewcommand*{\acrfullplfmt}[3]{%  
6875 \glslink[##1]{##2}{\acronymfont{\glsenentryshortpl{##2}}##3\space  
6876 (\glsenentrylongpl{##2})}%  
6877 \renewcommand*{\Acrfullplfmt}[3]{%  
6878 \glslink[##1]{##2}{\acronymfont{\Glsenentryshortpl{##2}}##3\space  
6879 (\glsenentrylongpl{##2})}%
```

```

6880 \renewcommand*{\ACRfullplfmt}[3]{%
6881   \glslink[##1]{##2}{%
6882     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6883     (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

6884 \renewcommand*{\glsentryfull}[1]{%
6885   \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6886 \renewcommand*{\Glsentryfull}[1]{%
6887   \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6888 \renewcommand*{\glsentryfullpl}[1]{%
6889   \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6890 \renewcommand*{\Glsentryfullpl}[1]{%
6891   \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6892 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6893 \newacronymstyle{footnote-sc}%
6894 {%
6895   \GlsUseAcrEntryDisplayStyle{footnote}%
6896 }%
6897 {%
6898   \GlsUseAcrStyleDefs{footnote}%
6899   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6900   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6901   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6902 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6903 \newacronymstyle{footnote-sm}%
6904 {%
6905   \GlsUseAcrEntryDisplayStyle{footnote}%
6906 }%
6907 {%
6908   \GlsUseAcrStyleDefs{footnote}%
6909   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6910   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6911   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6912 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6913 \newacronymstyle{footnote-desc}%
6914 {%
6915   \GlsUseAcrEntryDisplayStyle{footnote}%
6916 }%
6917 {%
6918   \GlsUseAcrStyleDefs{footnote}%
6919   \renewcommand*{\GenericAcronymFields}{}%

```

```

6920 \renewcommand*\acronymsort}[2]{##2}%
6921 \renewcommand*\acronymentry}[1]{%
6922   \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6923 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6924 \newacronymstyle{footnote-sc-desc}%
6925 {%
6926   \GlsUseAcrEntryDispStyle{footnote-sc}%
6927 }%
6928 {%
6929   \GlsUseAcrStyleDefs{footnote-sc}%
6930   \renewcommand*\GenericAcronymFields{}%
6931   \renewcommand*\acronymsort}[2]{##2}%
6932   \renewcommand*\acronymentry}[1]{%
6933     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6934 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6935 \newacronymstyle{footnote-sm-desc}%
6936 {%
6937   \GlsUseAcrEntryDispStyle{footnote-sm}%
6938 }%
6939 {%
6940   \GlsUseAcrStyleDefs{footnote-sm}%
6941   \renewcommand*\GenericAcronymFields{}%
6942   \renewcommand*\acronymsort}[2]{##2}%
6943   \renewcommand*\acronymentry}[1]{%
6944     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6945 }

```

AcronymSynonyms

```

6946 \newcommand*\DefineAcronymSynonyms{%

```

Short form

\acs

```

6947 \let\acs\acrshort

```

First letter uppercase short form

\Acs

```

6948 \let\Acs\Acrshort

```

Plural short form

\acsp

```

6949 \let\acsp\acrshortpl

```

First letter uppercase plural short form

`\Acsp`

6950 `\let\Acsp\Acrshortpl`

Long form

`\acl`

6951 `\let\acl\acrlong`

Plural long form

`\aclp`

6952 `\let\aclp\acrlongpl`

First letter upper case long form

`\Acl`

6953 `\let\Acl\Acrlong`

First letter upper case plural long form

`\Aclp`

6954 `\let\Aclp\Acrlongpl`

Full form

`\acf`

6955 `\let\acf\acrfull`

Plural full form

`\acfp`

6956 `\let\acfp\acrfullpl`

First letter upper case full form

`\Acf`

6957 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

6958 `\let\Acfp\Acrfullpl`

Standard form

`\ac`

6959 `\let\ac\gls`

First upper case standard form

`\Ac`

6960 `\let\Ac\Gls`

Standard plural form

`\acp`

```
6961 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
6962 \let\Acp\Glspl
```

```
6963 }
```

Define synonyms if required

```
6964 \ifglsacrshortcuts
```

```
6965 \DefineAcronymSynonyms
```

```
6966 \fi
```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\glsAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
6967 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
```

```
6968 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
6969 }
```

`\glsNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens

`\glslabeltok`, `\glsshorttok`, `\gslongtok` and `\glskeylisttok`.

```
6970 \newcommand*{\DefaultNewAcronymDef}{%
```

```
6971 \edef\@do@newglossaryentry{%
```

```
6972 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6973 {%
```

```
6974 type=\acronymtype,%
```

```
6975 name={\the\glsshorttok},%
```

```
6976 sort={\the\glsshorttok},%
```

```
6977 text={\the\glsshorttok},%
```

```
6978 first={\acrfullformat{\the\gslongtok}{\the\glsshorttok}},%
```

```
6979 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
```

```
6980 firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
```

```
6981 {\noexpand\expandonce\noexpand\@glo@shortpl}},%
```

```
6982 short={\the\glsshorttok},%
```

```
6983 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
```

```
6984 long={\the\gslongtok},%
```

```
6985 longplural={\the\gslongtok\noexpand\acrpluralsuffix},%
```

```
6986 description={\the\gslongtok},%
```

```
6987 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6988 \the\glskeylisttok
```

```
6989 }%
```

```
6990 }%
```

```
6991 \let\@org@gls@assign@firstpl\gls@assign@firstpl
```

```

6992 \let\@org@gls@assign@plural\gls@assign@plural
6993 \let\@org@gls@assign@descplural\gls@assign@descplural
6994 \def\gls@assign@firstpl##1##2{%
6995   \@gls@expand@field{##1}{firstpl}{##2}%
6996 }%
6997 \def\gls@assign@plural##1##2{%
6998   \@gls@expand@field{##1}{plural}{##2}%
6999 }%
7000 \def\gls@assign@descplural##1##2{%
7001   \@gls@expand@field{##1}{descplural}{##2}%
7002 }%
7003 \@do@newglossaryentry
7004 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7005 \let\gls@assign@plural\@org@gls@assign@plural
7006 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7007 }

```

`\ultAcronymStyle` Set up the default acronym style:

```
7008 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```

7009   \@for\@gls@type:=\@gls@acronymlists\do{%
7010     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7011   }%

```

Set up the definition of `\newacronym`:

```
7012 \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).

```

7013   \ifx\@gls@acronymlists\@empty
7014     \def\@glo@type{\acronymtype}%
7015     \setkeys{glossentry}{##1}%
7016     \DeclareAcronymList{\@glo@type}%
7017     \SetDefaultAcronymDisplayStyle{\@glo@type}%
7018   \fi
7019   \glskeylisttok{##1}%
7020   \glslabeltok{##2}%
7021   \glsshorttok{##3}%
7022   \glslongtok{##4}%
7023   \newacronymhook
7024   \DefaultNewAcronymDef
7025 }%
7026 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7027 }

```

`\acrfootnote` Used by the footnote acronym styles.

```
7028 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

`\acrlinkfootnote`

```

7029 \newcommand*{\acrlinkfootnote}[3]{%
7030   \footnote{\glslink[#1]{#2}{#3}}%
7031 }

```

acrnolinkfootnote

```

7032 \newcommand*{\acrnolinkfootnote}[3]{%
7033   \footnote{#3}%
7034 }

```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```

7035 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7036   \defglsentryfmt[#1]{%
7037     \ifdefempty\glscustomtext
7038     {%
7039       \ifglsused{\glslabel}%
7040       {%
7041         \acronymfont{\glsentryfmt}%
7042       }%
7043       {%
7044         \firstacronymfont{\glsentryfmt}%
7045         \ifgls hassymbol{\glslabel}%
7046         {%
7047           \expandafter\protect\expandafter\acrfootnote\expandafter
7048             {\@gls@link@opts}{\@gls@link@label}%
7049           {%
7050             \glsifplural
7051               {\glsentrysymbolplural{\glslabel}}%
7052               {\glsentrysymbol{\glslabel}}%
7053             }%
7054           }%
7055         }%
7056       }%
7057       {\glscustomtext\glsinsert}%
7058     }%
7059 }

```

acronymNewAcronymDef

```

7060 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7061   \edef\@do@newglossaryentry{%
7062     \noexpand\newglossaryentry{\the\glslabeltok}%
7063     {%
7064       type=\acronymtype,%
7065       name={\noexpand\acronymfont{\the\glsshorttok}},%
7066       sort={\the\glsshorttok},%
7067       first={\the\glsshorttok},%
7068       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7069       text={\the\glsshorttok},%

```



```

7070 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7071 short={\the\glsshorttok},%
7072 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7073 long={\the\glslongtok},%
7074 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7075 symbol={\the\glslongtok},%
7076 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7077 \the\glskeylisttok
7078 }%
7079 }%
7080 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7081 \let\@org@gls@assign@plural\gls@assign@plural
7082 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7083 \def\gls@assign@firstpl##1##2{%
7084   \@@gls@expand@field{##1}{firstpl}{##2}%
7085 }%
7086 \def\gls@assign@plural##1##2{%
7087   \@@gls@expand@field{##1}{plural}{##2}%
7088 }%
7089 \def\gls@assign@symbolplural##1##2{%
7090   \@@gls@expand@field{##1}{symbolplural}{##2}%
7091 }%
7092 \do@newglossaryentry
7093 \let\gls@assign@plural\@org@gls@assign@plural
7094 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7095 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7096 }

```

`oteAcronymStyle` 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.

```

7097 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7098   \renewcommand{\newacronym}[4][[]]{%
7099     \ifx\@glsacronymlists\@empty
7100       \def\@glo@type{\acronymtype}%
7101       \setkeys{glossentry}{##1}%
7102       \DeclareAcronymList{\@glo@type}%
7103       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7104     \fi
7105     \glskeylisttok{##1}%
7106     \glslabeltok{##2}%
7107     \glsshorttok{##3}%
7108     \glslongtok{##4}%
7109     \newacronymhook
7110     \DescriptionFootnoteNewAcronymDef
7111   }%

```

If footnote package option is specified, set the first use to append the long form (stored in

symbol) as a footnote.

```
7112 \@for\@gls@type:=\@glsacronymlists\do{%
7113   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7114 }%
```

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.

```
7115 \ifglsacrsmallcaps
7116   \renewcommand*\acronymfont}[1]{\textsc{##1}}%
7117   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7118 \else
7119   \ifglsacrsmaller
7120     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
7121   \fi
7122 \fi
```

Check for package option clash

```
7123 \ifglsacrdua
7124   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7125   can’t both be set}{}%
7126 \fi
7127 }%
```

`AcronymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```
7128 \newcommand*\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7129   \defglsentryfmt[#1]{\glsgenentryfmt}%
7130 }
```

`UANewAcronymDef`

```
7131 \newcommand*\DescriptionDUANewAcronymDef}{%
7132   \edef\@do@newglossaryentry{%
7133     \noexpand\newglossaryentry{\the\glslabeltok}%
7134     {%
7135       type=\acronymtype,%
7136       name={\the\glslongtok},%
7137       sort={\the\glslongtok},%
7138       text={\the\glslongtok},%
7139       first={\the\glslongtok},%
7140       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7141       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7142       short={\the\glsshorttok},%
7143       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7144       long={\the\glslongtok},%
7145       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7146       symbol={\the\glsshorttok},%
7147       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7148       \the\glskeylisttok
7149     }%
7150   }%
```

```

7151 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7152 \let\@org@gls@assign@plural\gls@assign@plural
7153 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7154 \def\gls@assign@firstpl##1##2{%
7155   \@gls@expand@field{##1}{firstpl}{##2}%
7156 }%
7157 \def\gls@assign@plural##1##2{%
7158   \@gls@expand@field{##1}{plural}{##2}%
7159 }%
7160 \def\gls@assign@symbolplural##1##2{%
7161   \@gls@expand@field{##1}{symbolplural}{##2}%
7162 }%
7163 \@do@newglossaryentry
7164 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7165 \let\gls@assign@plural\@org@gls@assign@plural
7166 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7167 }

```

DUAACronymStyle 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.

```

7168 \newcommand*{\SetDescriptionDUAACronymStyle}{%
7169   \ifglsacrsmallcaps
7170     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7171       can't both be set}{}%
7172   \else
7173     \ifglsacrsmaller
7174       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7175         can't both be set}{}%
7176     \fi
7177   \fi
7178   \renewcommand{\newacronym}[4][]{%
7179     \ifx\@glsacronymlists\@empty
7180       \def\@glo@type{\acronymtype}%
7181       \setkeys{glossentry}{##1}%
7182       \DeclareAcronymList{\@glo@type}%
7183       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7184     \fi
7185     \glskeylisttok{##1}%
7186     \glslabeltok{##2}%
7187     \glsshorttok{##3}%
7188     \glslongtok{##4}%
7189     \newacronymhook
7190     \DescriptionDUANewAcronymDef
7191   }%

```

Set display.

```

7192 \for\@gls@type:=\@glsacronymlists\do{%
7193   \SetDescriptionDUAACronymDisplayStyle{\@gls@type}%

```

```
7194 }%
7195 }%
```

ymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```
7196 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7197   \def\glsentryfmt[#1]{%

7198     \ifdefempty\glscustomtext
7199     {%
7200       \ifglsused{\glslabel}%
7201       {%
```

Move the inserted text outside of \acronymfont

```
7202     \let\gls@org@insert\glsinsert
7203     \let\glsinsert\@empty
7204     \acronymfont{\glsgenentryfmt}\gls@org@insert
7205   }%
7206   {%
7207     \glsgenentryfmt
7208     \ifgls hassymbol{\glslabel}%
7209     {%
7210       \glsifplural
7211       {%
7212         \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7213       }%
7214       {%
7215         \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7216       }%
7217       \space(\protect\firstacronymfont
7218         {\gls caps case
7219           {\@glo@symbol}
7220           {\@glo@symbol}
7221           {\mfirstucMakeUppercase{\@glo@symbol}}})%
7222     }%
7223   }%
7224 }%
7225 }%
7226 {\gls custom text\glsinsert}%
7227 }%
7228 }
```

onNewAcronymDef

```
7229 \newcommand*{\DescriptionNewAcronymDef}{%
7230   \edef\@do@newglossaryentry{%
7231     \noexpand\newglossaryentry{\the\glslabeltok}%
7232     {%
7233       type=\acronymtype,%
7234       name={\noexpand
```

```

7235     \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
7236     sort={\the\glsshorttok},%
7237     first={\the\glslongtok},%
7238     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7239     text={\the\glsshorttok},%
7240     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7241     short={\the\glsshorttok},%
7242     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7243     long={\the\glslongtok},%
7244     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7245     symbol={\noexpand\@glo@text},%
7246     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7247     \the\glskeylisttok}%
7248 }%
7249 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7250 \let\@org@gls@assign@plural\gls@assign@plural
7251 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7252 \def\gls@assign@firstpl##1##2{%
7253   \@@gls@expand@field{##1}{firstpl}{##2}%
7254 }%
7255 \def\gls@assign@plural##1##2{%
7256   \@@gls@expand@field{##1}{plural}{##2}%
7257 }%
7258 \def\gls@assign@symbolplural##1##2{%
7259   \@@gls@expand@field{##1}{symbolplural}{##2}%
7260 }%
7261 \@do@newglossaryentry
7262 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7263 \let\gls@assign@plural\@org@gls@assign@plural
7264 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7265 }

```

ionAcronymStyle 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 \acronymformat to allow the user to override the way the name is displayed in the list of acronyms.

```

7266 \newcommand*{\SetDescriptionAcronymStyle}{%
7267   \renewcommand{\newacronym}[4] []{%
7268     \ifx\@glsacronymlists\@empty
7269       \def\@glo@type{\acronymtype}%
7270       \setkeys{glossentry}{##1}%
7271       \DeclareAcronymList{\@glo@type}%
7272       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7273     \fi
7274     \glskeylisttok{##1}%
7275     \glslabeltok{##2}%
7276     \glsshorttok{##3}%
7277     \glslongtok{##4}%
7278     \newacronymhook
7279     \DescriptionNewAcronymDef

```

7280 }%

Set display.

```
7281 \@for\@gls@type:=\@glsacronymlists\do{%
7282   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7283 }%
```

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.

```
7284 \ifglsacrsmallcaps
7285   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7286   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7287 \else
7288   \ifglsacrsmaller
7289     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
7290   \fi
7291 \fi
7292 }%
```

`\acronymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```
7293 \newcommand*\SetFootnoteAcronymDisplayStyle}[1]{%
7294   \defglsentryfmt[#1]{%
```

```
7295     \ifdefempty\glscustomtext
7296     {%
```

Move the inserted text outside of `\acronymfont`

```
7297     \let\gls@org@insert\glsinsert
7298     \let\glsinsert\@empty
7299     \ifglsused{\glslabel}%
7300     {%
7301       \acronymfont{\glsentryfmt}\gls@org@insert
7302     }%
7303     {%
7304       \firstacronymfont{\glsentryfmt}\gls@org@insert
7305       \ifglsahaslong{\glslabel}%
7306       {%
7307         \expandafter\protect\expandafter\acrfootnote\expandafter
7308         {\@gls@link@opts}{\@gls@link@label}%
7309         {%
7310           \glsifplural
7311             {\glsentrylongpl{\glslabel}}%
7312             {\glsentrylong{\glslabel}}%
7313           }%
7314         }%
7315       }%
7316     }%
7317 }%
```

```

7318   {\glscustomtext\glsinsert}%
7319   }%
7320 }

```

teNewAcronymDef

```

7321 \newcommand*{\FootnoteNewAcronymDef}{%
7322   \edef\@do@newglossaryentry{%
7323     \noexpand\newglossaryentry{\the\glslabeltok}%
7324     {%
7325       type=\acronymtype,%
7326       name={\noexpand\acronymfont{\the\glsshorttok}},%
7327       sort={\the\glsshorttok},%
7328       text={\the\glsshorttok},%
7329       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7330       first={\the\glsshorttok},%
7331       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7332       short={\the\glsshorttok},%
7333       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7334       long={\the\glslongtok},%
7335       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7336       description={\the\glslongtok},%
7337       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7338       \the\glskeylisttok
7339     }%
7340   }%
7341   \let\@org@gls@assign@plural\gls@assign@plural
7342   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7343   \let\@org@gls@assign@descplural\gls@assign@descplural
7344   \def\gls@assign@firstpl##1##2{%
7345     \@@gls@expand@field{##1}{firstpl}{##2}%
7346   }%
7347   \def\gls@assign@plural##1##2{%
7348     \@@gls@expand@field{##1}{plural}{##2}%
7349   }%
7350   \def\gls@assign@descplural##1##2{%
7351     \@@gls@expand@field{##1}{descplural}{##2}%
7352   }%
7353   \@do@newglossaryentry
7354   \let\gls@assign@plural\@org@gls@assign@plural
7355   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7356   \let\gls@assign@descplural\@org@gls@assign@descplural
7357 }

```

oteAcronymStyle 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.

```

7358 \newcommand*{\SetFootnoteAcronymStyle}{%
7359   \renewcommand{\newacronym}[4][]{%
7360     \ifx\@glsacronymlists\@empty
7361       \def\@glo@type{\acronymtype}%

```

```

7362     \setkeys{glossentry}{##1}%
7363     \DeclareAcronymList{\@glo@type}%
7364     \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7365     \fi
7366     \glskeylisttok{##1}%
7367     \glslabeltok{##2}%
7368     \glsshorttok{##3}%
7369     \glslongtok{##4}%
7370     \newacronymhook
7371     \FootnoteNewAcronymDef
7372 }%

```

Set display

```

7373 \@for\@gls@type:=\@gls@acronymlists\do{%
7374   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7375 }%

```

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.

```

7376 \ifglsacrsmallcaps
7377   \renewcommand*\acronymfont}[1]{\textsc{##1}}%
7378   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7379 \else
7380   \ifglsacrsmaller
7381     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
7382   \fi
7383 \fi

```

Check for option clash

```

7384 \ifglsacrdua
7385   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7386     can’t both be set}{}%
7387 \fi
7388 }%

```

`parenifnotempty` 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.

```

7389 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
7390   \protected@edef\gls@tmp{##1}%
7391   \ifdefempty\gls@tmp
7392   }{%
7393   {%
7394     \ifx\gls@tmp\@gls@default@value
7395     \else
7396       \space (#2{##1})%
7397     \fi
7398   }%
7399 }

```


`nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7400 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7401   \defglentryfmt[#1]{%

7402     \ifdefempty\glscustomtext
7403     {%

      Move the inserted text outside of \acronymfont

7404       \let\gls@org@insert\glsinsert
7405       \let\glsinsert\@empty
7406       \ifglsused{\glslabel}%
7407       {%
7408         \acronymfont{\glsentryfmt}\gls@org@insert
7409       }%
7410       {%
7411         \glsentryfmt
7412         \ifglshassymbol{\glslabel}%
7413         {%
7414           \glsifplural
7415           {%
7416             \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7417           }%
7418           {%
7419             \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7420           }%
7421           \space
7422           (\glscapscase
7423            {\firstacronymfont{\@glo@symbol}}%
7424            {\firstacronymfont{\@glo@symbol}}%
7425            {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7426           }%
7427         }%
7428       }%
7429     }%
7430   {\glscustomtext\glsinsert}%
7431 }%
7432 }

```

`llNewAcronymDef`

```

7433 \newcommand*{\SmallNewAcronymDef}{%
7434   \edef\@do@newglossaryentry{%
7435     \noexpand\newglossaryentry{\the\glslabeltok}%
7436     {%
7437       type=\acronymtype,%
7438       name={\noexpand\acronymfont{\the\glsshorttok}},%
7439       sort={\the\glsshorttok},%
7440       text={\the\glsshorttok},%

```

Default to the short plural.

```

7441 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7442 first={\the\glslongtok},%
Default to the long plural.
7443 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7444 short={\the\glsshorttok},%
7445 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7446 long={\the\glslongtok},%
7447 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7448 description={\noexpand\@glo@first},%
7449 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7450 symbol={\the\glsshorttok},%
Default to the short plural.
7451 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7452 \the\glskeylisttok
7453 }%
7454 }%
7455 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7456 \let\@org@gls@assign@plural\gls@assign@plural
7457 \let\@org@gls@assign@descplural\gls@assign@descplural
7458 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7459 \def\gls@assign@firstpl##1##2{%
7460 \@@gls@expand@field{##1}{firstpl}{##2}%
7461 }%
7462 \def\gls@assign@plural##1##2{%
7463 \@@gls@expand@field{##1}{plural}{##2}%
7464 }%
7465 \def\gls@assign@descplural##1##2{%
7466 \@@gls@expand@field{##1}{descplural}{##2}%
7467 }%
7468 \def\gls@assign@symbolplural##1##2{%
7469 \@@gls@expand@field{##1}{symbolplural}{##2}%
7470 }%
7471 \do@newglossaryentry
7472 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7473 \let\gls@assign@plural\@org@gls@assign@plural
7474 \let\gls@assign@descplural\@org@gls@assign@descplural
7475 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7476 }

```

`allAcronymStyle` 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.

```

7477 \newcommand*{\SetSmallAcronymStyle}{%
7478 \renewcommand{\newacronym}[4][ ]{%
7479 \ifx\@glsacronymlists\@empty
7480 \def\@glo@type{\acronymtype}%
7481 \setkeys{glossentry}{##1}%
7482 \DeclareAcronymList{\@glo@type}%
7483 \SetSmallAcronymDisplayStyle{\@glo@type}%

```

```

7484 \fi
7485 \glskeylisttok{##1}%
7486 \glslabeltok{##2}%
7487 \glsshorttok{##3}%
7488 \glslongtok{##4}%
7489 \newacronymhook
7490 \SmallNewAcronymDef
7491 }%

```

Change the display since first only contains long form.

```

7492 \@for\@gls@type:=\@gls@acronymlists\do{%
7493 \SetSmallAcronymDisplayStyle{\@gls@type}%
7494 }%

```

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.

```

7495 \ifglsacrsmallcaps
7496 \renewcommand*\acronymfont}[1]{\textsc{##1}}
7497 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7498 \else
7499 \renewcommand*\acronymfont}[1]{\textsmaller{##1}}
7500 \fi

```

check for option clash

```

7501 \ifglsacrdua
7502 \ifglsacrsmallcaps
7503 \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7504 can't both be set}{}%
7505 \else
7506 \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7507 can't both be set}{}%
7508 \fi
7509 \fi
7510 }%

```

`DUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```

7511 \newcommand*\SetDUADisplayStyle}[1]{%
7512 \defglsentryfmt[#1]{\glsentryfmt}%
7513 }

```

`UANewAcronymDef`

```

7514 \newcommand*\DUANewAcronymDef}{%
7515 \edef\@do@newglossaryentry{%
7516 \noexpand\newglossaryentry{\the\glslabeltok}%
7517 {%
7518 type=\acronymtype,%
7519 name={\the\glsshorttok},%
7520 text={\the\glslongtok},%
7521 first={\the\glslongtok},%
7522 plural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

```

7523     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7524     short={\the\glsshorttok},%
7525     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7526     long={\the\glslongtok},%
7527     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7528     description={\the\glslongtok},%
7529     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7530     symbol={\the\glsshorttok},%
7531     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7532     \the\glskeylisttok
7533   }%
7534 }%
7535 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7536 \let\@org@gls@assign@plural\gls@assign@plural
7537 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7538 \let\@org@gls@assign@descplural\gls@assign@descplural
7539 \def\gls@assign@firstpl##1##2{%
7540   \@@gls@expand@field{##1}{firstpl}{##2}%
7541 }%
7542 \def\gls@assign@plural##1##2{%
7543   \@@gls@expand@field{##1}{plural}{##2}%
7544 }%
7545 \def\gls@assign@symbolplural##1##2{%
7546   \@@gls@expand@field{##1}{symbolplural}{##2}%
7547 }%
7548 \def\gls@assign@descplural##1##2{%
7549   \@@gls@expand@field{##1}{descplural}{##2}%
7550 }%
7551 \@do@newglossaryentry
7552 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7553 \let\gls@assign@plural\@org@gls@assign@plural
7554 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7555 \let\gls@assign@descplural\@org@gls@assign@descplural
7556 }

```

\SetDUASyle Always expand acronyms.

```

7557 \newcommand*{\SetDUASyle}{%
7558   \renewcommand{\newacronym}[4][]{%
7559     \ifx\@glsacronymlists\@empty
7560       \def\@glo@type{\acronymtype}%
7561       \setkeys{glossentry}{##1}%
7562       \DeclareAcronymList{\@glo@type}%
7563       \SetDUADisplayStyle{\@glo@type}%
7564     \fi
7565     \glskeylisttok{##1}%
7566     \glslabeltok{##2}%
7567     \glsshorttok{##3}%
7568     \glslongtok{##4}%
7569     \newacronymhook

```

```

7570 \DUANewAcronymDef
7571 }%
    Set the display
7572 \@for\@gls@type:=\@glsacronymlists\do{%
7573 \SetDUADisplayStyle{\@gls@type}%
7574 }%
7575 }

```

SetAcronymStyle

```

7576 \newcommand*\SetAcronymStyle{%
7577 \SetDefaultAcronymStyle
7578 \ifglsacrdescription
7579 \ifglsacrfootnote
7580 \SetDescriptionFootnoteAcronymStyle
7581 \else
7582 \ifglsacrdua
7583 \SetDescriptionDUAAcronymStyle
7584 \else
7585 \SetDescriptionAcronymStyle
7586 \fi
7587 \fi
7588 \else
7589 \ifglsacrfootnote
7590 \SetFootnoteAcronymStyle
7591 \else
7592 \ifthenelse{\boolean{glsacrsmalldescription}\OR
7593 \boolean{glsacrsmaller}}{%
7594 }%
7595 \SetSmallAcronymStyle
7596 }%
7597 }%
7598 \ifglsacrdua
7599 \SetDUASyle
7600 \fi
7601 }%
7602 \fi
7603 \fi
7604 }

```

Set the acronym style according to the package options

```
7605 \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.

```
7606 \newcommand*\SetCustomDisplayStyle}[1]{%
```

```

7607 \defglentryfmt[#1]{\glsgenentryfmt}%
7608 }

```

omAcronymFields

```

7609 \newcommand*{\CustomAcronymFields}{%
7610   name={\the\glsshorttok},%
7611   description={\the\glslongtok},%
7612   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7613   firstplural={\acrfullformat
7614     {\noexpand\glsentrylongpl{\the\glslabeltok}}%
7615     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%
7616   text={\the\glsshorttok},%
7617   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7618 }

```

omNewAcronymDef

```

7619 \newcommand*{\CustomNewAcronymDef}{%
7620   \protected@edef\do@newglossaryentry{%
7621     \noexpand\newglossaryentry{\the\glslabeltok}%
7622     {%
7623       type=\acronymtype,%
7624       short={\the\glsshorttok},%
7625       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7626       long={\the\glslongtok},%
7627       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7628       user1={\the\glsshorttok},%
7629       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7630       user3={\the\glslongtok},%
7631       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7632       \CustomAcronymFields,%
7633       \the\glskeylisttok
7634     }%
7635   }%
7636   \do@newglossaryentry
7637 }

```

\SetCustomStyle

```

7638 \newcommand*{\SetCustomStyle}{%
7639   \renewcommand{\newacronym}[4][ ]{%
7640     \ifx\@glsacronymlists\@empty
7641       \def\@glo@type{\acronymtype}%
7642       \setkeys{glossentry}{##1}%
7643       \DeclareAcronymList{\@glo@type}%
7644       \SetCustomDisplayStyle{\@glo@type}%
7645     \fi
7646     \glskeylisttok{##1}%
7647     \glslabeltok{##2}%
7648     \glsshorttok{##3}%

```

```

7649 \glslongtok{##4}%
7650 \newacronymhook
7651 \CustomNewAcronymDef
7652 }%
Set the display
7653 \@for\@gls@type:=\@glsacronymlists\do{%
7654 \SetCustomDisplayStyle{\@gls@type}%
7655 }%
7656 }

```

1.19 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.

```
7657 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7658 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
7659 \@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.

```
7660 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7661 \@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.

```

7662 \ifx\@glossary@default@style\relax
7663 \else
7664 \setglossarystyle{\@glossary@default@style}
7665 \fi

```

1.20 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```

7666 \newcommand*{\showgloparent}[1]{%
7667 \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
7668 }

```

`\showglolevel` `\showglolevel{<label>}`

```
7669 \newcommand*{\showglolevel}[1]{%
7670   \expandafter\show\csname glo@glstetoklabel{#1}@level\endcsname
7671 }
```

`\showglotext` `\showglotext{<label>}`

```
7672 \newcommand*{\showglotext}[1]{%
7673   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
7674 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7675 \newcommand*{\showgloplural}[1]{%
7676   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7677 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7678 \newcommand*{\showglofirst}[1]{%
7679   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7680 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7681 \newcommand*{\showglofirstpl}[1]{%
7682   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7683 }
```

`\showglotype` `\showglotype{<label>}`

```
7684 \newcommand*{\showglotype}[1]{%
7685   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7686 }
```


`\showglocounter` `\showglocounter{<label>}`

```
7687 \newcommand*{\showglocounter}[1]{%
7688   \expandafter\show\csname glo@glstdetoklabel{#1}@counter\endcsname
7689 }
```

`\showglouser` `\showglouser{<label>}`

```
7690 \newcommand*{\showglouser}[1]{%
7691   \expandafter\show\csname glo@glstdetoklabel{#1}@user\endcsname
7692 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7693 \newcommand*{\showglouserii}[1]{%
7694   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7695 }
```

`\showglouseriii` `\showglouseriii{<label>}`

```
7696 \newcommand*{\showglouseriii}[1]{%
7697   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7698 }
```

`\showglouseriv` `\showglouseriv{<label>}`

```
7699 \newcommand*{\showglouseriv}[1]{%
7700   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7701 }
```

`\showglouserv` `\showglouserv{<label>}`

```
7702 \newcommand*{\showglouserv}[1]{%
7703   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
7704 }
```

`\showglouservi` `\showglouservi{<label>}`

```
7705 \newcommand*{\showglouservi}[1]{%
7706   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7707 }
```

`\showgloname` `\showgloname{<label>}`

```
7708 \newcommand*{\showgloname}[1]{%
7709   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7710 }
```

`\showglodesc` `\showglodesc{<label>}`

```
7711 \newcommand*{\showglodesc}[1]{%
7712   \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname
7713 }
```

`showglodescplural` `\showglodescplural{<label>}`

```
7714 \newcommand*{\showglodescplural}[1]{%
7715   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7716 }
```

`\showglosort` `\showglosort{<label>}`

```
7717 \newcommand*{\showglosort}[1]{%
7718   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7719 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
7720 \newcommand*{\showglosymbol}[1]{%
7721   \expandafter\show\csname glo@glstdetoklabel{#1}@symbol\endcsname
7722 }
```

glosymbolplural `\showglosymbolplural{<label>}`

```
7723 \newcommand*{\showglosymbolplural}[1]{%
7724   \expandafter\show\csname glo@glstdetoklabel{#1}@symbolplural\endcsname
7725 }
```

\showgloshort `\showgloshort{<label>}`

```
7726 \newcommand*{\showgloshort}[1]{%
7727   \expandafter\show\csname glo@glstdetoklabel{#1}@short\endcsname
7728 }
```

\showglolong `\showglolong{<label>}`

```
7729 \newcommand*{\showglolong}[1]{%
7730   \expandafter\show\csname glo@glstdetoklabel{#1}@long\endcsname
7731 }
```

\showgloindex `\showgloindex{<label>}`

```
7732 \newcommand*{\showgloindex}[1]{%
7733   \expandafter\show\csname glo@glstdetoklabel{#1}@index\endcsname
7734 }
```

\showgloflag `\showgloflag{<label>}`

```
7735 \newcommand*{\showgloflag}[1]{%
7736   \expandafter\show\csname ifglo@glstdetoklabel{#1}@flag\endcsname
7737 }
```

\showgloloclist `\showgloloclist{<label>}`

```
7738 \newcommand*{\showgloloclist}[1]{%
7739   \expandafter\show\csname glo@glstdetoklabel{#1}@loclist\endcsname
7740 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7741 \newcommand*{\showglofield}[2]{%
7742 \csshow{glo@glstetoklabel{#1}@#2}%
7743 }
```

`showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7744 \newcommand*{\showacronymlists}{%
7745 \show\@glsacronymlists
7746 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7747 \newcommand*{\showglossaries}{%
7748 \show\@glo@types
7749 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the 'in' extension for the given glossary.

```
7750 \newcommand*{\showglossaryin}[1]{%
7751 \expandafter\show\csname @glo@type@#1@in\endcsname
7752 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the 'out' extension for the given glossary.

```
7753 \newcommand*{\showglossaryout}[1]{%
7754 \expandafter\show\csname @glo@type@#1@out\endcsname
7755 }
```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7756 \newcommand*{\showglossarytitle}[1]{%
7757 \expandafter\show\csname @glo@type@#1@title\endcsname
7758 }
```

wglossarycounter `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7759 \newcommand*{\showglossarycounter}[1]{%
7760   \expandafter\show\csname @glotype@#1@counter\endcsname
7761 }
```

wglossaryentries `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7762 \newcommand*{\showglossaryentries}[1]{%
7763   \expandafter\show\csname glolist@#1\endcsname
7764 }
```

1.21 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.

```
7765 \csname ifglcompatible-2.07\endcsname
7766   \RequirePackage{glossaries-compatible-207}
7767 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`a \gls{<label>}`” on first use but use “`an \gls{<label>}`” on subsequent use.

```
7768 \NeedsTeXFormat{LaTeX2e}
```

```
7769 \ProvidesPackage{glossaries-prefix}[2016/06/09 v4.25 (NLCT)]
```

Pass all options to glossaries:

```
7770 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7771 \ProcessOptions
```

Load glossaries:

```
7772 \RequirePackage{glossaries}
```

Add the new keys:

```
7773 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7774 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7775 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7776 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7777 \appto\@gls@keymap{,%
```

```
7778   {prefixfirst}{prefixfirst},%
```

```
7779   {prefixfirstplural}{prefixfirstplural},%
```

```
7780   {prefix}{prefix},%
```

```
7781   {prefixplural}{prefixplural}}%
```

```
7782 }
```

Set the default values:

```
7783 \appto\@newglossaryentryprehook{%
```

```
7784   \def\@glo@entryprefix{}}%
```

```
7785   \def\@glo@entryprefixplural{}}%
```

```
7786   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7787   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7788 }
```

Set the assignment code:

```
7789 \appto\@newglossaryentryposthook{%
```

```
7790   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}}%
```

```
7791   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7792 \expandafter\gls@assign@field\expandafter
```

```
7793   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7794   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```
7795 \expandafter\gls@assign@field\expandafter
7796   {\csname glo@\glo@label @prefixplural\endcsname}{\@glo@label}%
7797   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7798 }
```

Define commands to access these fields:

entryprefixfirst

```
7799 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}
```

entryprefixfirstplural

```
7800 \newcommand*\glsentryprefixfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}
```

\glsentryprefix

```
7801 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}
```

entryprefixplural

```
7802 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}
```

Now for the initial upper case variants:

entryprefixfirst

```
7803 \newrobustcmd*\Glsentryprefixfirst[1]{%
7804   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7805   \xmakefirstuc\@glo@text
7806 }
```

entryprefixfirstplural

```
7807 \newrobustcmd*\Glsentryprefixfirstplural[1]{%
7808   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7809   \xmakefirstuc\@glo@text
7810 }
```

\Glsentryprefix

```
7811 \newrobustcmd*\Glsentryprefix[1]{%
7812   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7813   \xmakefirstuc\@glo@text
7814 }
```

entryprefixplural

```
7815 \newrobustcmd*\Glsentryprefixplural[1]{%
7816   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7817   \xmakefirstuc\@glo@text
7818 }
```

Define commands to determine if the prefix keys have been set:

`\ifglshasprefix`

```
7819 \newcommand*\ifglshasprefix}[3]{%
7820   \ifcseempty{glo@#1@prefix}%
7821   {#3}%
7822   {#2}%
7823 }
```

`hasprefixplural`

```
7824 \newcommand*\ifglshasprefixplural}[3]{%
7825   \ifcseempty{glo@#1@prefixplural}%
7826   {#3}%
7827   {#2}%
7828 }
```

`shasprefixfirst`

```
7829 \newcommand*\ifglshasprefixfirst}[3]{%
7830   \ifcseempty{glo@#1@prefixfirst}%
7831   {#3}%
7832   {#2}%
7833 }
```

`efixfirstplural`

```
7834 \newcommand*\ifglshasprefixfirstplural}[3]{%
7835   \ifcseempty{glo@#1@prefixfirstplural}%
7836   {#3}%
7837   {#2}%
7838 }
```

Define commands that insert the prefix before commands like `\gls`:

`\pgls`

```
7839 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

`\@pgls` Unstarred version.

```
7840 \newcommand*\@pgls}[2][ ]{%
7841   \new@ifnextchar[%
7842     {\@pgls@{#1}{#2}}%
7843     {\@pgls@{#1}{#2}[ ]}%
7844 }
```

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

```
7845 \def\@pgls@#1#2[#3]{%
7846   \glsdoifexists{#2}%
7847   {%
7848     \ifglsused{#2}%
7849     {%
7850       \glsentryprefix{#2}%
7851     }%

```



```

7852   {%
7853     \glsentryprefixfirst{#2}%
7854   }%
7855   \@gls@{#1}{#2}[#3]%
7856 }%
7857 }

```

Similarly for the plural version:

```

\pglsp1
7858 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7859 \newcommand*{\@pglsp1}[2][ ]{%
7860   \new@ifnextchar[%
7861     {\@pglsp1@{#1}{#2}}%
7862     {\@pglsp1@{#1}{#2}[ ]}%
7863 }

```

\@pglsp1@ Read in the final optional argument:

```

7864 \def\@pglsp1@#1#2[#3]{%
7865   \glsdoifexists{#2}%
7866   {%
7867     \ifglsused{#2}%
7868     {%
7869       \glsentryprefixplural{#2}%
7870     }%
7871     {%
7872       \glsentryprefixfirstplural{#2}%
7873     }%
7874     \@glspl@{#1}{#2}[#3]%
7875   }%
7876 }

```

Now for the first letter upper case versions:

```

\Pgls
7877 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7878 \newcommand*{\@Pgls}[2][ ]{%
7879   \new@ifnextchar[%
7880     {\@Pgls@{#1}{#2}}%
7881     {\@Pgls@{#1}{#2}[ ]}%
7882 }

```

\@Pgls@ Read in the final optional argument:

```

7883 \def\@Pgls@#1#2[#3]{%

```

```

7884 \glsdoifexists{#2}%
7885 {%
7886   \ifglsused{#2}%
7887   {%
7888     \ifglshasprefix{#2}%
7889     {%
7890       \Glsentryprefix{#2}%
7891       \@gls@{#1}{#2}[#3]%
7892     }%
7893     {\@Gls@{#1}{#2}[#3]}%
7894   }%
7895   {%
7896     \ifglshasprefixfirst{#2}%
7897     {%
7898       \Glsentryprefixfirst{#2}%
7899       \@gls@{#1}{#2}[#3]%
7900     }%
7901     {\@Gls@{#1}{#2}[#3]}%
7902   }%
7903 }%
7904 }

```

Similarly for the plural version:

`\Pglsp1`

```
7905 \newrobustcmd{\Pglsp1}{\@gls@hyp@opt\Pglsp1}
```

`\@Pglsp1` Unstarred version.

```

7906 \newcommand*{\@Pglsp1}[2] [] {%
7907   \new@ifnextchar [%
7908     {\@Pglsp1@{#1}{#2}}%
7909     {\@Pglsp1@{#1}{#2} []}%
7910 }

```

`\@Pglsp1@` Read in the final optional argument:

```

7911 \def\@Pglsp1@#1#2[#3] {%
7912   \glsdoifexists{#2}%
7913   {%
7914     \ifglsused{#2}%
7915     {%
7916       \ifglshasprefixplural{#2}%
7917       {%
7918         \Glsentryprefixplural{#2}%
7919         \@glspl@{#1}{#2}[#3]%
7920       }%
7921       {\@Glspl@{#1}{#2}[#3]}%
7922     }%
7923     {%
7924       \ifglshasprefixfirstplural{#2}%

```

```

7925     {%
7926         \Glsentryprefixfirstplural{#2}%
7927         \@glspl@{#1}{#2}[#3]%
7928     }%
7929     {\@Glspl@{#1}{#2}[#3]}%
7930 }%
7931 }%
7932 }

```

Finally the all upper case versions:

\PGLS

```
7933 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}
```

\@PGLS Unstarred version.

```

7934 \newcommand*{\@PGLS}[2][ ]{%
7935     \new@ifnextchar[%
7936     {\@PGLS@{#1}{#2}}%
7937     {\@PGLS@{#1}{#2}[ ]}%
7938 }

```

\@PGLS@ Read in the final optional argument:

```

7939 \def\@PGLS@#1#2[#3]{%
7940     \glsdoifexists{#2}%
7941     {%
7942         \ifglsused{#2}%
7943         {%
7944             \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7945         }%
7946         {%
7947             \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7948         }%
7949         \@GLS@{#1}{#2}[#3]%
7950     }%
7951 }

```

Plural version:

\PGLSp1

```
7952 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}
```

\@PGLSp1 Unstarred version.

```

7953 \newcommand*{\@PGLSp1}[2][ ]{%
7954     \new@ifnextchar[%
7955     {\@PGLSp1@{#1}{#2}}%
7956     {\@PGLSp1@{#1}{#2}[ ]}%
7957 }

```

\@PGLSp1@ Read in the final optional argument:

```
7958 \def\@PGLSp1@#1#2[#3]{%
7959   \glsdoifexists{#2}%
7960   {%
7961     \ifglsused{#2}%
7962     {%
7963       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7964     }%
7965     {%
7966       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7967     }%
7968     \@GLSp1@{#1}{#2}[#3]%
7969   }%
7970 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7971 \ProvidesPackage{glossary-hypernav}[2016/06/09 v4.25 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [section 1.16](#).) `\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`

```
7972 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7973   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7974   \@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`

```
7975 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7976   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7977   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7978   \expandafter\let
7979     \expandafter\@gls@list\csname @gls@hypergroup@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7980   \@for\@gls@elem:=\@gls@list\do{%
7981     \ifthenelse{equal{\@gls@elem}{#2}}{\@endfortrue}{}}}
```

Check if list terminated prematurely.

```
7982 \if@endfor
7983 \else
```

This group was not included in the list, so issue a warning.

```
7984 \GlossariesWarningNoLine{Navigation panel
7985     for glossary type ‘#1’^^Jmissing group ‘#2’}%
7986 \gdef\gls@hypergrouprerun{%
7987     \GlossariesWarningNoLine{Navigation panel
7988     has changed. Rerun LaTeX}}%
7989 \fi
7990 }
```

`hypergrouprerun` Give a warning at the end if re-run required

```
7991 \let\gls@hypergrouprerun\relax
7992 \AtEndDocument{\gls@hypergrouprerun}
```

`@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup@list@<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.

```
7993 \newcommand*{\@gls@hypergroup}[2]{%
7994 \@ifundefined{@gls@hypergroup@list@#1}{%
7995     \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
7996 }{%
7997     \expandafter\let\expandafter\@gls@tmp
7998         \csname @gls@hypergroup@list@#1\endcsname
7999     \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
8000         \@gls@tmp,#2}%
8001 }%
8002 }
```

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`

```
8003 \newcommand*{\glsnavigation}{%
8004     \def\@gls@between{}%
8005     \ifcsundef{@gls@hypergroup@list@\@glo@type}%
8006     {%
8007         \def\@gls@list{}%
8008     }%
8009     {%
8010         \expandafter\let\expandafter\@gls@list
8011             \csname @gls@hypergroup@list@\@glo@type\endcsname
8012     }%
8013     \@for\@gls@tmp:=\@gls@list\do{%
```

```

8014   \@gls@between

8015   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8016   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8017   \let\@gls@between\glshypernavsep
8018   }%
8019 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8020 \newcommand*{\glshypernavsep}{\space\textbar\space}
```

The `\glsymbolnav` 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.

`\glsymbolnav`

```

8021 \newcommand*{\glsymbolnav}{%
8022 \glsnavhyperlink{glsymbols}{\glsgetgrouptitle{glsymbols}}%
8023 \glshypernavsep
8024 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
8025 \glshypernavsep
8026 }

```

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.

```
8027 \ProvidesPackage{glossary-inline}[2016/06/09 v4.25 (NLCT)]
```

`inline` Define the inline style.

```
8028 \newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossentry`)

```

8029 \renewenvironment{theglossary}%
8030   {%
8031     \def\gls@inlinesep{}%
8032     \def\gls@inlinesubsep{}%
8033     \def\gls@inlinepostchild{}%
8034   }%
8035   {\glspostinline}%

```

No header:

```
8036 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
8037 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
8038 \renewcommand{\glossentry}[2]{%
8039   \glsinlinedopostchild
8040   \gls@inlinesep
8041   \glsentryitem{##1}%
8042   \glsinlinenameformat{##1}{%
8043     \glossentryname{##1}%
8044   }%
8045   \ifglsdescsuppressed{##1}%
8046   {%
8047     \glsinlineemptydescformat
8048     {%
8049       \glossentrysymbol{##1}%
8050     }%
8051     {%
8052       ##2%
8053     }%
8054   }%
8055   {%
8056     \ifglsdesc{##1}%
8057     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8058     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8059   }%
8060   \ifglschildren{##1}%
8061   {%
8062     \glsresetsubentrycounter
8063     \glsinlineparentchildseparator
8064     \def\gls@inlinesubsep{}%
8065     \def\gls@inlinepostchild{\glsinlinepostchild}%
8066   }%
8067   {}%
8068   \def\gls@inlinesep{\glsinlineseparator}%
8069 }%
```

Sub-entries display description:

```
8070 \renewcommand{\subglossentry}[3]{%
8071   \gls@inlinesubsep%
8072   \glsinlinesubnameformat{##2}{%
8073     \glossentryname{##2}}%
8074   \glsentryitem{##2}%
8075   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8076   \def\gls@inlinesubsep{\glsinlinesubseparator}%
8077 }%
```

Nothing special between groups:

```
8078 \renewcommand*{\glsgroupskip}{}%
8079 }
```

linedopostchild

```
8080 \newcommand*{\glsinlinedopostchild}{%
```



```

8081 \gls@inlinepostchild
8082 \def\gls@inlinepostchild{}%
8083 }

```

`inlineseparator` Separator to use between entries.

```
8084 \newcommand*\glsinlineseparator}{;\space}
```

`inlinesubseparator` Separator to use between sub-entries.

```
8085 \newcommand*\glsinlinesubseparator}{,\space}
```

`parentchildseparator` Separator to use between parent and children.

```
8086 \newcommand*\glsinlineparentchildseparator}{:\space}
```

`inlinepostchild` Hook to use between child and next entry

```
8087 \newcommand*\glsinlinepostchild}{}
```

`\glspostinline` Terminator for inline glossary.

```
8088 \newcommand*\glspostinline}{\glspostdescription\space}
```

`inlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
8089 \newcommand*\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`inlinedescformat` Formats the entry's description, symbol and location list:

```
8090 \newcommand*\glsinlinedescformat}[3]{\space#1}
```

`emptydescformat` Formats the entry's symbol and location list when the description is empty:

```
8091 \newcommand*\glsinlineemptydescformat}[2]{}
```

`inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
8092 \newcommand*\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
8093 \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.

```
8094 \ProvidesPackage{glossary-list}[2016/06/09 v4.25 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8095 \providecommand{\indexspace}{%
8096 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8097 }

```

`tgroupheaderfmt` Provide a way of adjusting the format of the group headings.

```
8098 \newcommand*{\glslistgroupheaderfmt}[1]{#1}
```

`tnavigationitem` Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of `item` to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```
8099 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}
```

`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.

```
8100 \newglossarystyle{list}{%
```

Use description environment:

```
8101 \renewenvironment{theglossary}{%
```

```
8102   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8103 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8104 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8105 \renewcommand*{\glossentry}[2]{%
```

```
8106   \item[\glsentryitem{##1}]%
```

```
8107       \glstarget{##1}{\glossentryname{##1}}
```

```
8108       \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
8109 \renewcommand*{\subglossentry}[3]{%
```

```
8110   \glssubentryitem{##2}%
```

```
8111   \glstarget{##2}{\strut}\space
```

```
8112   \glossentrydesc{##2}\glspostdescription\space ##3.}%
```

```
8113 % \end{macrocode}
```

```
8114 % Add vertical space between groups:
```

```
8115 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
```

```
8116 % \begin{macrocode}
```

```
8117 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
```

```
8118 }
```

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

```
8119 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
8120 \setglossarystyle{list}%
```

Each group has a heading:

```
8121 \renewcommand*\glsgroupheading}[1]{%
8122   \item[\glslistgroupheaderfmt{\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.

```
8123 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
8124 \setglossarystyle{list}%
```

Add navigation links at the start of the environment.

```
8125 \renewcommand*\glossaryheader}{%
```

```
8126   \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a hypertext:

```
8127 \renewcommand*\glsgroupheading}[1]{%
```

```
8128   \item[\glslistgroupheaderfmt
```

```
8129     {\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.

```
8130 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
8131 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
8132 \renewcommand*\glossentry}[2]{%
```

```
8133   \item[\glsentryitem{##1}%
```

```
8134     \glstarget{##1}{\glossentryname{##1}}}]%
```

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.

```
8135   \mbox{}\par\nobreak\@afterheading
```

```
8136   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
8137 \renewcommand{\subglossentry}[3]{%
```

```
8138   \par
```

```
8139   \glssubentryitem{##2}%
```

```
8140   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
```

```
8141 }
```

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

```
8142 \newglossarystyle{altlistgroup}{%
```

Base it on the `altlist` style:

```
8143 \setglossarystyle{altlist}%
```

Each group has a heading:

```
8144 \renewcommand*\glsgroupheading}[1]{%
8145   \item[\glslistgroupheaderfmt{\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.

```
8146 \newglossarystyle{altlisthypergroup}{%
```

Base it on the `altlist` style:

```
8147 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment.

```
8148 \renewcommand*\glossaryheader}{%
8149   \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a hypertarget:

```
8150 \renewcommand*\glsgroupheading}[1]{%
8151   \item[\glslistgroupheaderfmt
8152     {\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.

```
8153 \newglossarystyle{listdotted}{%
```

Base it on the `list` style:

```
8154 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
8155 \renewcommand*\glossentry}[2]{%
8156   \item[]\makebox[\glslistdottedwidth][l]{%
8157     \glsentryitem{##1}%
8158     \glstarget{##1}{\glossentryname{##1}}%
8159     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
8160 \renewcommand*\subglossentry}[3]{%
8161   \item[]\makebox[\glslistdottedwidth][l]{%
8162     \glssubentryitem{##2}%
8163     \glstarget{##2}{\glossentryname{##2}}%
8164     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8165 }
```

`listdottedwidth`

```
8166 \newlength\glslistdottedwidth
8167 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8168 \newglossarystyle{sublistdotted}{%
```

Base it on the listdotted style:

```
8169 \setglossarystyle{listdotted}%
Main (level 0) entries just display the name:
8170 \renewcommand*{\glossentry}[2]{%
8171 \item[\glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
8172 }
```

3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the longtable environment in the glossary.

```
8173 \ProvidesPackage{glossary-long}[2016/06/09 v4.25 (NLCT)]
```

Requires the package:

```
8174 \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`.)

```
8175 \@ifundefined{glsdescwidth}{%
8176 \newlength{glsdescwidth}
8177 \setlength{glsdescwidth}{0.6\hsize}
8178 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
8179 \@ifundefined{glspagelistwidth}{%
8180 \newlength{glspagelistwidth}
8181 \setlength{glspagelistwidth}{0.1\hsize}
8182 }{}
```

`long` The long glossary style command which uses the longtable environment:

```
8183 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8184 \renewenvironment{theglossary}%
8185 {\begin{longtable}[lp{glsdescwidth}]}%
8186 {\end{longtable}}%
```

Do nothing at the start of the environment:

```
8187 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8188 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8189 \renewcommand{\glossentry}[2]{%
8190 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8191 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8192 }%
```

Sub entries displayed on the following row without the name:

```
8193 \renewcommand{\subglossentry}[3]{%
8194     &
8195     \glssubentryitem{##2}%
8196     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8197     ##3\tabularnewline
8198 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip`
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8199 \ifglsnogroupskip
8200 \renewcommand*\glsgroupskip{}%
8201 \else
8202 \renewcommand*\glsgroupskip{ & \tabularnewline}%
8203 \fi
8204 }
```

`longborder` The `longborder` style is like the above, but with horizontal and vertical lines:

```
8205 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```
8206 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8207 \renewenvironment{theglossary}{%
```

```
8208 \begin{longtable}{|l|p{\glsdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8209 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8210 }
```

`longheader` The `longheader` style is like the `long` style but with a header:

```
8211 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8212 \setglossarystyle{long}%
```

Set the table's header:

```
8213 \renewcommand*\glossaryheader{%
```

```
8214 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
```

```
8215 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8216 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8217 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8218 \renewcommand*\glossaryheader{%
```

```
8219 \hline\bfseries \entryname & \bfseries
```

```
8220 \descriptionname\tabularnewline\hline
```

```

8221 \endhead
8222 \hline\endfoot}%
8223 }

```

`long3col` The `long3col` style is like `long` but with 3 columns

```

8224 \newglossarystyle{long3col}{%
    Use a longtable with 3 columns:
8225 \renewenvironment{theglossary}%
8226 {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
8227 {\end{longtable}}%

```

No table header:

```
8228 \renewcommand*\glossaryheader{}%
```

No headings between groups:

```
8229 \renewcommand*\glsgroupheading[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8230 \renewcommand{\glossentry}[2]{%
8231 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8232 \glossentrydesc{##1} & ##2\tabularnewline
8233 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8234 \renewcommand{\subglossentry}[3]{%
8235 &
8236 \glssubentryitem{##2}%
8237 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8238 ##3\tabularnewline
8239 }%

```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8240 \ifglsnogroupskip
8241 \renewcommand*\glsgroupskip{}%
8242 \else
8243 \renewcommand*\glsgroupskip{ & & \tabularnewline}%
8244 \fi
8245 }

```

`long3colborder` The `long3colborder` style is like the `long3col` style but with a border:

```

8246 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8247 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8248 \renewenvironment{theglossary}%
8249 {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8250 {\end{longtable}}%

```

Place horizontal lines at the head and foot of the table:

```
8251 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8252 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
8253 \newglossarystyle{long3colheader}{%
```

Base it on the `glostylelong3col` style:

```
8254 \setglossarystyle{long3col}%
```

Set the table's header:

```
8255 \renewcommand*{\glossaryheader}{%
```

```
8256 \bfseries\entryname&\bfseries\descriptionname&
```

```
8257 \bfseries\pagelistname\tabularnewline\endhead}%
```

```
8258 }
```

`colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
8259 \newglossarystyle{long3colheaderborder}{%
```

Base it on the `glostylelong3colborder` style:

```
8260 \setglossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8261 \renewcommand*{\glossaryheader}{%
```

```
8262 \hline
```

```
8263 \bfseries\entryname&\bfseries\descriptionname&
```

```
8264 \bfseries\pagelistname\tabularnewline\hline\endhead
```

```
8265 \hline\endfoot}%
```

```
8266 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8267 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8268 \renewenvironment{theglossary}%
```

```
8269 {\begin{longtable}{llll}}%
```

```
8270 {\end{longtable}}%
```

No table header:

```
8271 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8272 \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):

```
8273 \renewcommand{\glossentry}[2]{%
```

```
8274 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8275 \glossentrydesc{##1} &
```

```
8276 \glossentrysymbol{##1} &
```

```
8277 ##2\tabularnewline
```

```
8278 }%
```


Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8279 \renewcommand{\subglossentry}[3]{%
8280     &
8281     \glssubentryitem{##2}%
8282     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8283     \glossentrysymbol{##2} & ##3\tabularnewline
8284 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8285 \ifglsnogroupskip
8286   \renewcommand*{\glsgroupskip}{}%
8287 \else
8288   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8289 \fi
8290 }
```

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

```
8291 \newglossarystyle{long4colheader}{%
      Base it on the glostylelong4col style:
8292   \setglossarystyle{long4col}%
      Table has a header:
8293   \renewcommand*{\glossaryheader}{%
8294     \bfseries\entryname&\bfseries\descriptionname&
8295     \bfseries \symbolname&
8296     \bfseries\pagelistname\tabularnewline\endhead}%
8297 }
```

`long4colborder` The `long4colborder` style is like `long4col` but with a border.

```
8298 \newglossarystyle{long4colborder}{%
      Base it on the glostylelong4col style:
8299   \setglossarystyle{long4col}%
      Use a longtable with 4 columns surrounded by vertical lines:
8300   \renewenvironment{theglossary}%
8301     {\begin{longtable}{|l|l|l|l|}}%
8302     {\end{longtable}}%
      Add horizontal lines to the head and foot of the table:
8303   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8304 }
```

`colheaderborder` The `long4colheaderborder` style is like the above but with a border.

```
8305 \newglossarystyle{long4colheaderborder}{%
      Base it on the glostylelong4col style:
8306   \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8307 \renewenvironment{theglossary}%  
8308   {\begin{longtable}{|l|l|l|l|}}%  
8309   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8310 \renewcommand*{\glossaryheader}{%  
8311   \hline\bfseries\entryname&\bfseries\descriptionname&  
8312   \bfseries \symbolname&  
8313   \bfseries\pagelistname\tabularnewline\hline\endhead  
8314   \hline\endfoot}%  
8315 }
```

`altlong4col` The `altlong4col` style is like the `long4col` style but can have multiline descriptions and page lists.

```
8316 \newglossarystyle{altlong4col}{%
```

Base it on the `glostylelong4col` style:

```
8317   \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8318 \renewenvironment{theglossary}%  
8319   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%  
8320   {\end{longtable}}%  
8321 }
```

`long4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```
8322 \newglossarystyle{altlong4colheader}{%
```

Base it on the `glostylelong4colheader` style:

```
8323   \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8324 \renewenvironment{theglossary}%  
8325   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%  
8326   {\end{longtable}}%  
8327 }
```

`long4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
8328 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
8329   \setglossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8330 \renewenvironment{theglossary}%  
8331   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%  
8332   {\end{longtable}}%  
8333 }
```

colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
8334 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the glostylelong4colheaderborder style:

```
8335 \setglossarystyle{long4colheaderborder}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8336 \renewenvironment{theglossary}{%
```

```
8337   {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8338   {\end{longtable}}%
```

```
8339 }
```

3.5 Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package

The styles here are based on David Carlisle's patch at <http://tex.stackexchange.com/a/56890>

```
8340 \ProvidesPackage{glossary-longbooktabs}[2016/06/09 v4.25 (NLCT)]
```

Requires booktabs package:

```
8341 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8342 \RequirePackage{glossary-long}
```

```
8343 \RequirePackage{glossary-longragged}
```

(longtable and array loaded by those packages).

long-booktabs The long-booktabs style is similar to the longheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8344 \newglossarystyle{long-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8345 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8346 \setglossarystyle{long}{%
```

Add a header with rules.

```
8347 \renewcommand*{\glossaryheader}{%
```

```
8348   \toprule \bfseries \entryname & \bfseries
```

```
8349   \descriptionname\tabularnewline\midrule\endhead
```

```
8350   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8351 \ifglsnogroupskip
```

```

8352   \renewcommand*{\glsgroupskip}{}%
8353   \else
8354   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8355   \fi
8356 }

```

ng3col-booktabs The long3col-booktabs style is similar to the long3colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8357 \newglossarystyle{long3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8358   \glspatchLToutput
```

Use the long3col style as a base.

```
8359   \setglossarystyle{long3col}{%
```

Add a header with rules.

```

8360   \renewcommand*{\glossaryheader}{%
8361     \toprule \bfseries \entryname &
8362     \bfseries \descriptionname &
8363     \bfseries \pagelistname
8364     \tabularnewline\midrule\endhead
8365     \bottomrule\endfoot}%

```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```

8366   \ifglsnogroupskip
8367     \renewcommand*{\glsgroupskip}{}%
8368   \else
8369     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8370   \fi
8371 }

```

ng4col-booktabs The long4col-booktabs style is similar to the long4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8372 \newglossarystyle{long4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8373   \glspatchLToutput
```

Use the long4col style as a base.

```
8374   \setglossarystyle{long4col}{%
```

Add a header with rules.

```

8375   \renewcommand*{\glossaryheader}{%
8376     \toprule \bfseries \entryname &
8377     \bfseries \descriptionname &
8378     \bfseries \symbolname &

```

```

8379   \bfseries \pagelistname
8380   \tabularnewline\midrule\endhead
8381   \bottomrule\endfoot}%

```

Check for the `nogroupskip` package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for `nogroupskip` should occur outside `\glsgroupskip` to be on the safe side.

```

8382   \ifglsnogroupskip
8383     \renewcommand*{\glsgroupskip}{}%
8384   \else
8385     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8386   \fi
8387 }

```

`ng4col-booktabs` The `altlong4col-booktabs` style is similar to the `altlong4colheader` style but uses the `booktabs` rules and patches `longtable` to check for group skip occurring at a page break.

```
8388 \newglossarystyle{altlong4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8389   \glspatchLToutput
```

Use the `long4col-booktabs` style as a base.

```
8390   \setglossarystyle{long4col-booktabs}%
```

Change the column specifications:

```

8391   \renewenvironment{theglossary}%
8392     {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8393     {\end{longtable}}%
8394 }

```

Ragged styles.

`ragged-booktabs` The `longragged-booktabs` style is similar to the `longragged` style but uses the `booktabs` rules and patches `longtable` to check for group skip occurring at a page break.

```
8395 \newglossarystyle{longragged-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8396   \glspatchLToutput
```

Use the `long-booktabs` style as a base.

```
8397   \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```

8398   \renewenvironment{theglossary}%
8399     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8400     {\end{longtable}}%
8401 }

```

ed3col-booktabs The longragged3col-booktabs style is similar to the longragged3col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8402 \newglossarystyle{longragged3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8403 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8404 \setglossarystyle{long3col-booktabs}{%
```

Adjust the column specification.

```
8405 \renewenvironment{theglossary}{%
```

```
8406   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}%
```

```
8407     >{\raggedright}p{\glspagelistwidth}}}%
```

```
8408   {\end{longtable}}}%
```

```
8409 }
```

ed4col-booktabs The altlongragged4col-booktabs style is similar to the altlongragged4col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8410 \newglossarystyle{altlongragged4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8411 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8412 \setglossarystyle{altlong4col-booktabs}{%
```

Adjust the column specification.

```
8413 \renewenvironment{theglossary}{%
```

```
8414   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}l%
```

```
8415     >{\raggedright}p{\glspagelistwidth}}}%
```

```
8416   {\end{longtable}}}%
```

```
8417 }
```

sLTpenaltycheck

```
8418 \newcommand*{\glsLTpenaltycheck}{%
```

```
8419   \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
```

```
8420 }
```

enaltygroupskip

```
8421 \newcommand{\glspenaltygroupskip}{%
```

```
8422   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8423 \let\@gls@org@LT@output\LT@output
```

```
8424 \newcommand*{\glsrestoreLToutput}{\let\LT@output\@gls@org@LT@output}
```

This is David's patch, but I've replaced the hard-coded values with \glsLTpenaltycheck to make it easier to adjust.

lspatchLToutput

```
8425 \newcommand*{\glspatchLToutput}{%
8426 \renewcommand*{\LT@output}{%
8427 \ifnum\outputpenalty <-\@Mi
8428 \ifnum\outputpenalty > -\LT@end@pen
8429 \LT@err{floats and marginpars not allowed in a longtable}\@ehc
8430 \else
8431 \setbox\z@\vbox{\unvbox\@cclv}%
8432 \ifdim \ht\LT@lastfoot>\ht\LT@foot
8433 \dimen@\pagegoal
8434 \advance\dimen@-\ht\LT@lastfoot
8435 \ifdim\dimen@<\ht\z@
8436 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8437 \@makecol
8438 \@outputpage
8439 \setbox\z@\vbox{\box\LT@head\glstpenaltycheck}%
8440 \fi
8441 \fi
8442 \global\@colroom\@colht
8443 \global\vsizel\@colht
8444 {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8445 \fi
8446 \else
8447 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8448 \@makecol
8449 \@outputpage
8450 \global\vsizel\@colroom
8451 \copy\LT@head
8452 \glstpenaltycheck
8453 \nobreak
8454 \fi
8455 }%
8456 }
```

3.6 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.

```
8457 \ProvidesPackage{glossary-longragged}[2016/06/09 v4.25 (NLCT)]
```

Requires the package:

```
8458 \RequirePackage{array}
```

Requires the package:

```
8459 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
8460 \@ifundefined{glsdescwidth}{%
8461   \newlength{glsdescwidth
8462   \setlength{glsdescwidth}{0.6\hsize}
8463 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8464 \@ifundefined{glspagelistwidth}{%
8465   \newlength{glspagelistwidth
8466   \setlength{glspagelistwidth}{0.1\hsize}
8467 }{}
```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8468 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```
8469   \renewenvironment{theglossary}%
8470     {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%
8471     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
8472   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8473   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8474   \renewcommand{\glossentry}[2]{%
8475     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8476     \glossentrydesc{##1}\glspostdescription\space ##2%
8477     \tabularnewline
8478   }%
```

Sub entries displayed on the following row without the name:

```
8479   \renewcommand{\subglossentry}[3]{%
8480     &
8481     \glssubentryitem{##2}%
8482     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8483     \glspostdescription\space ##3%
8484     \tabularnewline
8485   }%
```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8486   \ifglsnogroupskip
8487   \renewcommand*{\glsgroupskip}{}%
8488   \else
8489   \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
```



```
8490 \fi
8491 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8492 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8493 \setglossarystyle{longragged}{%
```

Use `longtable` with two columns with vertical lines between each column:

```
8494 \renewenvironment{theglossary}{%
```

```
8495 \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}}%
```

```
8496 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8497 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
8498 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8499 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8500 \setglossarystyle{longragged}{%
```

Set the table's header:

```
8501 \renewcommand*{\glossaryheader}{%
```

```
8502 \bfseries \entryname & \bfseries \descriptionname
```

```
8503 \tabularnewline\endhead}%
```

```
8504 }
```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8505 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8506 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8507 \renewcommand*{\glossaryheader}{%
```

```
8508 \hline\bfseries \entryname & \bfseries \descriptionname
```

```
8509 \tabularnewline\hline
```

```
8510 \endhead
```

```
8511 \hline\endfoot}%
```

```
8512 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8513 \newglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns:

```
8514 \renewenvironment{theglossary}{%
```

```
8515 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
```

```
8516 >{\raggedright}p{\glspagelistwidth}}}%
```

```
8517 {\end{longtable}}%
```

No table header:

```
8518 \renewcommand*\glossaryheader}{}
```

No headings between groups:

```
8519 \renewcommand*\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8520 \renewcommand\glossentry}[2]{%
8521   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8522   \glossentrydesc{##1} & ##2\tabularnewline
8523 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8524 \renewcommand\subglossentry}[3]{%
8525   &
8526   \glssubentryitem{##2}%
8527   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8528   ##3\tabularnewline
8529 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8530 \ifglsnogroupskip
8531   \renewcommand*\glsgroupskip}{}%
8532 \else
8533   \renewcommand*\glsgroupskip}{ & & \tabularnewline}%
8534 \fi
8535 }
```

`ragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8536 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8537 \setglossarystyle{longragged3col}%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
8538 \renewenvironment{theglossary}%
8539   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
8540    >{\raggedright}p{\glspagelistwidth}|}%
8541   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8542 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8543 }
```

`ragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8544 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8545 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8546 \renewcommand*{\glossaryheader}{%
8547   \bfseries\entryname&\bfseries\descriptionname&
8548   \bfseries\pagelistname\tabularnewline\endhead}%
8549 }
```

colheaderborder The longragged3colheaderborder style is like the above but with a border

```
8550 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the glostylelongragged3colborder style:

```
8551 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8552 \renewcommand*{\glossaryheader}{%
8553   \hline
8554   \bfseries\entryname&\bfseries\descriptionname&
8555   \bfseries\pagelistname\tabularnewline\hline\endhead
8556   \hline\endfoot}%
8557 }
```

tlongragged4col 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.

```
8558 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8559 \renewenvironment{theglossary}%
8560   {\begin{longtable}{1>{\raggedright}p{\glsdescwidth}1%
8561     >{\raggedright}p{\glspagelistwidth}}}%
8562   {\end{longtable}}%
```

No table header:

```
8563 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8564 \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):

```
8565 \renewcommand{\glossentry}[2]{%
8566   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8567   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8568   ##2\tabularnewline
8569   }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8570 \renewcommand{\subglossentry}[3]{%
8571   &
8572   \glssubentryitem{##2}%
8573   \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8574 \glossentrysymbol{##2} & ##3\tabularnewline
8575 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8576 \ifglsnogroupskip
8577 \renewcommand*{\glsgroupskip}{}%
8578 \else
8579 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8580 \fi
8581 }
```

`ragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```
8582 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8583 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8584 \renewenvironment{theglossary}%
8585 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8586 >{\raggedright}p{\glspagelistwidth}}}%
8587 {\end{longtable}}%
```

Table has a header:

```
8588 \renewcommand*{\glossaryheader}{%
8589 \bfseries\entryname&\bfseries\descriptionname&
8590 \bfseries \symbolname&
8591 \bfseries\pagelistname\tabularnewline\endhead}%
8592 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8593 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8594 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8595 \renewenvironment{theglossary}%
8596 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
8597 >{\raggedright}p{\glspagelistwidth}|}%
8598 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8599 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8600 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8601 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8602 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8603 \renewenvironment{theglossary}%  
8604   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%  
8605     >{\raggedright}p{\glspagelistwidth}|}}%  
8606   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8607 \renewcommand*{\glossaryheader}{%  
8608   \hline\bfseries\entryname&\bfseries\descriptionname&  
8609   \bfseries \symbolname&  
8610   \bfseries\pagelistname\tabularnewline\hline\endhead  
8611   \hline\endfoot}%  
8612 }
```

3.7 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.

```
8613 \ProvidesPackage{glossary-mcols}[2016/06/09 v4.25 (NLCT)]
```

Required packages:

```
8614 \RequirePackage{multicol}  
8615 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8616 \providecommand{\indexspace}{%  
8617   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax  
8618 }
```

`\glscols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8619 \newcommand*{\glscols}{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.)

```
8620 \newglossarystyle{mcolindex}{%  
8621   \setglossarystyle{index}%  
8622   \renewenvironment{theglossary}%  
8623     {%  
  
8624     \begin{multicols}{\glscols}  
8625     \setlength{\parindent}{0pt}%  
8626     \setlength{\parskip}{0pt plus 0.3pt}%
```

```

8627     \let\item\@idxitem}%
8628     {\end{multicols}}}%
8629 }

```

`mcolindexgroup` As `mcolindex` but has headings:

```

8630 \newglossarystyle{mcolindexgroup}{%
8631  \setglossarystyle{mcolindex}%
8632  \renewcommand*{\glsgroupheading}[1]{%
8633    \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\indexspace}%
8634 }

```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```

8635 \newglossarystyle{mcolindexhypergroup}{%

```

Base it on the `glostylemcolindex` style:

```

8636  \setglossarystyle{mcolindex}%

```

Put navigation links to the groups at the start of the glossary:

```

8637  \renewcommand*{\glossaryheader}{%
8638    \item\glstreenavigationfmt{\glsnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8639  \renewcommand*{\glsgroupheading}[1]{%
8640    \item\glstreegroupheaderfmt
8641      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8642    \indexspace}%
8643 }

```

`colindexspannav` Similar to `mcolindexhypergroup`, but puts the navigation line in the optional argument of `multicols`.

```

8644 \newglossarystyle{mcolindexspannav}{%
8645  \setglossarystyle{index}%
8646  \renewenvironment{theglossary}%
8647    {%
8648      \begin{multicols}{\glscols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8649      \setlength{\parindent}{0pt}%
8650      \setlength{\parskip}{0pt plus 0.3pt}%
8651      \let\item\@idxitem}%
8652    {\end{multicols}}}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8653  \renewcommand*{\glsgroupheading}[1]{%
8654    \item\glstreegroupheaderfmt
8655      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8656    \indexspace}%
8657 }

```

`mcoltree` Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8658 \newglossarystyle{mcoltree}{%
8659   \setglossarystyle{tree}%
8660   \renewenvironment{theglossary}%
8661   {%
8662     \begin{multicols}{\glsmcols}
8663     \setlength{\parindent}{0pt}%
8664     \setlength{\parskip}{0pt plus 0.3pt}%
8665   }%
8666   {\end{multicols}}%
8667 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```
8668 \newglossarystyle{mcoltreegroup}{%
      Base it on the glostylemcoltree style:
8669   \setglossarystyle{mcoltree}%
      Each group has a heading (in bold) followed by a vertical gap):
8670   \renewcommand{\glsgroupheading}[1]{\par
8671     \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8672 }
```

`1treehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8673 \newglossarystyle{mcoltreehypergroup}{%
      Base it on the glostylemcoltree style:
8674   \setglossarystyle{mcoltree}%
      Put navigation links to the groups at the start of the theglossary environment:
8675   \renewcommand*{\glossaryheader}{%
8676     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8677   \renewcommand*{\glsgroupheading}[1]{%
8678     \par\noindent
8679     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8680     \indexspace}%
8681 }
```

`mcoltreespannav` Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the multicols environment.

```
8682 \newglossarystyle{mcoltreespannav}{%
8683   \setglossarystyle{tree}%
8684   \renewenvironment{theglossary}%
8685   {%
```

```

8686     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8687     \setlength{\parindent}{0pt}%
8688     \setlength{\parskip}{0pt plus 0.3pt}%
8689 }%
8690 {\end{multicols}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8691 \renewcommand*{\glsgroupheading}[1]{%
8692   \par\noindent
8693   \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8694   \indexspace}%
8695 }

```

mcoltreenoname Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```

8696 \newglossarystyle{mcoltreenoname}{%
8697   \setglossarystyle{treenoname}%
8698   \renewenvironment{theglossary}%
8699   {%
8700     \begin{multicols}{\glsmcols}
8701     \setlength{\parindent}{0pt}%
8702     \setlength{\parskip}{0pt plus 0.3pt}%
8703   }%
8704   {\end{multicols}}%
8705 }

```

treenonamegroup Like the `mcoltreenoname` style but the glossary groups have headings.

```

8706 \newglossarystyle{mcoltreenonamegroup}{%
8707   Base it on the glostylemcoltreenoname style:
8708   \setglossarystyle{mcoltreenoname}%
8709   Give each group a heading:
8710   \renewcommand{\glsgroupheading}[1]{\par
8711     \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8712 }

```

onamehypergroup The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8711 \newglossarystyle{mcoltreenonamehypergroup}{%
8712   Base it on the glostylemcoltreenoname style:
8713   \setglossarystyle{mcoltreenoname}%
8714   Put navigation links to the groups at the start of the theglossary environment:
8715   \renewcommand*{\glossaryheader}{%
8716     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
8717   Each group has a heading (in bold with a target) followed by a vertical gap):
8718   \renewcommand*{\glsgroupheading}[1]{%
8719     \par\noindent

```



```

8717 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8718 \indexspace}%
8719 }

```

`reenonamespannav` Similar to the `mcoltreenamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8720 \newglossarystyle{mcoltreenamepannav}{%
8721 \setglossarystyle{treename}%
8722 \renewenvironment{theglossary}%
8723 {%
8724 \begin{multicols}{\glsncols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8725 \setlength{\parindent}{0pt}%
8726 \setlength{\parskip}{0pt plus 0.3pt}%
8727 }%
8728 {\end{multicols}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8729 \renewcommand*{\glsgroupheading}[1]{%
8730 \par\noindent
8731 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8732 \indexspace}%
8733 }

```

`mcolalmtree` Multi-column index style. Same as the `almtree`, but puts the glossary in multiple columns.

```

8734 \newglossarystyle{mcolalmtree}{%
8735 \setglossarystyle{almtree}%
8736 \renewenvironment{theglossary}%
8737 {%
8738 \begin{multicols}{\glsncols}
8739 \def\@gls@prevlevel{-1}%
8740 \mbox{}\par
8741 }%
8742 {\par\end{multicols}}%
8743 }

```

`mcolalmtreegroup` Like the `mcolalmtree` style but the glossary groups have headings.

```

8744 \newglossarystyle{mcolalmtreegroup}{%

```

Base it on the `glostylemcolalmtree` style:

```

8745 \setglossarystyle{mcolalmtree}%

```

Give each group a heading.

```

8746 \renewcommand{\glsgroupheading}[1]{\par
8747 \def\@gls@prevlevel{-1}%
8748 \hangindent0pt\relax
8749 \parindent0pt\relax
8750 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8751 }

```

treehypergroup The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```
8752 \newglossarystyle{mcolalttreehypergroup}{%
```

```
    Base it on the glostylemcolalttree style:
```

```
8753 \setglossarystyle{mcolalttree}{%
```

```
    Put the navigation links in the header
```

```
8754 \renewcommand*{\glossaryheader}{%
```

```
8755 \par
```

```
8756 \def\@gls@prevlevel{-1}%
```

```
8757 \hangindent0pt\relax
```

```
8758 \parindent0pt\relax
```

```
8759 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

```
    Put a hypertarget at the start of each group
```

```
8760 \renewcommand*{\glsgroupheading}[1]{%
```

```
8761 \par
```

```
8762 \def\@gls@prevlevel{-1}%
```

```
8763 \hangindent0pt\relax
```

```
8764 \parindent0pt\relax
```

```
8765 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8766 \indexspace}%
```

```
8767 }
```

alttreespannav Similar to the mcolalttreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
8768 \newglossarystyle{mcolalttreespannav}{%
```

```
8769 \setglossarystyle{alttree}{%
```

```
8770 \renewenvironment{theglossary}{%
```

```
8771 {%
```

```
8772 \begin{multicols}{\glsncols}\noindent\glstreenavigationfmt{\glsnavigation}]
```

```
8773 \def\@gls@prevlevel{-1}%
```

```
8774 \mbox{}]\par
```

```
8775 }%
```

```
8776 {\par\end{multicols}}}%
```

```
    Put a hypertarget at the start of each group
```

```
8777 \renewcommand*{\glsgroupheading}[1]{%
```

```
8778 \par
```

```
8779 \def\@gls@prevlevel{-1}%
```

```
8780 \hangindent0pt\relax
```

```
8781 \parindent0pt\relax
```

```
8782 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8783 \indexspace}
```

```
8784 }
```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
8785 \ProvidesPackage{glossary-super}[2016/06/09 v4.25 (NLCT)]
```

Requires the package:

```
8786 \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.

```
8787 \@ifundefined{glsdescwidth}{%
8788   \newlength{glsdescwidth
8789   \setlength{glsdescwidth}{0.6\hsize}
8790 }{}
```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```
8791 \@ifundefined{glspagelistwidth}{%
8792   \newlength{glspagelistwidth
8793   \setlength{glspagelistwidth}{0.1\hsize}
8794 }{}
```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8795 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8796 \renewenvironment{theglossary}%
8797   {\tablehead{\tabletail}{%
8798   \begin{supertabular}[lp{glsdescwidth}]}%
8799   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8800 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8801 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8802 \renewcommand{\glossentry}[2]{%
8803   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8804   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8805 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8806 \renewcommand{\subglossentry}[3]{%
8807   &
8808   \glssubentryitem{##2}%
```

```

8809     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8810     ##3\tabularnewline
8811 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8812 \ifglsnogroupskip
8813   \renewcommand*{\glsgroupskip}{}%
8814 \else
8815   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
8816 \fi
8817 }

```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8818 \newglossarystyle{superborder}{%
```

Base it on the glostylesuper style:

```
8819 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

8820 \renewenvironment{theglossary}%
8821   {\tablehead{\hline}\tabletail{\hline}%
8822    \begin{supertabular}{|l|p{\glsdescwidth}|}%
8823    {\end{supertabular}}%
8824 }

```

superheader The superheader style is like the super style, but with a header:

```
8825 \newglossarystyle{superheader}{%
```

Base it on the glostylesuper style:

```
8826 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8827 \renewenvironment{theglossary}%
8828   {\tablehead{\bfseries \entryname &
8829    \bfseries\descriptionname\tabularnewline}%
8830    \tabletail{}}%
8831   \begin{supertabular}{lp{\glsdescwidth}}%
8832   {\end{supertabular}}%
8833 }

```

perheaderborder The superheaderborder style is like the super style but with a header and border:

```
8834 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
8835 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8836 \renewenvironment{theglossary}%
```

```

8837   {\tablehead{\hline\bfseries \entryname &
8838           \bfseries \descriptionname\abularnewline\hline}%
8839   \tabletail{\hline}
8840   \begin{supertabular}{|l|p{\glsdescwidth}|}%
8841   {\end{supertabular}}%
8842 }

```

`super3col` The `super3col` style is like the `super` style, but with 3 columns:

```
8843 \newglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```

8844 \renewenvironment{theglossary}%
8845   {\tablehead{}\tabletail{}%
8846   \begin{supertabular}{|p{\glsdescwidth}p{\glspagelistwidth}|}%
8847   {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8848 \renewcommand*\glossaryheader{}%
```

No group headings:

```
8849 \renewcommand*\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8850 \renewcommand{\glossentry}[2]{%
8851   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8852   \glossentrydesc{##1} & ##2\abularnewline
8853 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

8854 \renewcommand{\subglossentry}[3]{%
8855   &
8856   \glssubentryitem{##2}%
8857   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8858   ##3\abularnewline
8859 }%

```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8860 \ifglsnogroupskip
8861   \renewcommand*\glsgroupskip{}%
8862 \else
8863   \renewcommand*\glsgroupskip}{& & \abularnewline}%
8864 \fi
8865 }

```

`super3colborder` The `super3colborder` style is like the `super3col` style, but with a border:

```
8866 \newglossarystyle{super3colborder}{%
```

Base it on the `glostylesuper3col` style:

```
8867 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8868 \renewenvironment{theglossary}%
8869   {\tablehead{\hline}\tabletail{\hline}%
8870    \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8871    {\end{supertabular}}%
8872 }
```

`super3colheader` The `super3colheader` style is like the `super3col` style but with a header row:

```
8873 \newglossarystyle{super3colheader}{%
```

Base it on the `glostylesuper3col` style:

```
8874 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
8875 \renewenvironment{theglossary}%
8876   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8877    \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8878   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
8879   {\end{supertabular}}%
8880 }
```

`colheaderborder` The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```
8881 \newglossarystyle{super3colheaderborder}{%
```

Base it on the `glostylesuper3colborder` style:

```
8882 \setglossarystyle{super3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8883 \renewenvironment{theglossary}%
8884   {\tablehead{\hline
8885    \bfseries\entryname&\bfseries\descriptionname&
8886    \bfseries\pagelistname\tabularnewline\hline}%
8887   \tabletail{\hline}%
8888   \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8889   {\end{supertabular}}%
8890 }
```

`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.

```
8891 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8892 \renewenvironment{theglossary}%
8893   {\tablehead{} \tabletail{}}%
8894   \begin{supertabular}{l|l|l|l}}{%
8895   \end{supertabular}}%
```

Do nothing at the start of the table:

```
8896 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8897 \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:

```
8898 \renewcommand{\glossentry}[2]{%
8899   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8900   \glossentrydesc{##1} &
8901   \glossentrysymbol{##1} & ##2\tabularnewline
8902 }
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8903 \renewcommand{\subglossentry}[3]{%
8904   &
8905   \glssubentryitem{##2}%
8906   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8907   \glossentrysymbol{##2} & ##3\tabularnewline
8908 }
```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8909 \ifglsnogroupskip
8910 \renewcommand*\glsgroupskip}{}%
8911 \else
8912 \renewcommand*\glsgroupskip}{& & \tabularnewline}%
8913 \fi
8914 }
```

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

```
8915 \newglossarystyle{super4colheader}{%
```

Base it on the `glostylesuper4col` style:

```
8916 \setglossarystyle{super4col}{%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
8917 \renewenvironment{theglossary}{%
8918   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8919     \bfseries\symbolname &
8920     \bfseries\pagelistname\tabularnewline}%
8921   \tabletail}{%
8922   \begin{supertabular}{1111}}}%
8923   {\end{supertabular}}}%
8924 }
```

`super4colborder` The `super4colborder` style is like the `super4col` but with a border.

```
8925 \newglossarystyle{super4colborder}{%
```

Base it on the `glostylesuper4col` style:

```
8926 \setglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
8927 \renewenvironment{theglossary}%
8928   {\tablehead{\hline}\tabletail{\hline}%
8929    \begin{supertabular}{|l|l|l|l|}}%
8930   {\end{supertabular}}%
8931 }
```

`colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
8932 \newglossarystyle{super4colheaderborder}{%
```

Base it on the `glostylesuper4col` style:

```
8933 \setglossarystyle{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:

```
8934 \renewenvironment{theglossary}%
8935   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8936             \bfseries\symbolname &
8937             \bfseries\pagelistname\tabularnewline\hline}%
8938   \tabletail{\hline}%
8939   \begin{supertabular}{|l|l|l|l|}}%
8940   {\end{supertabular}}%
8941 }
```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8942 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostylesuper4col` style:

```
8943 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8944 \renewenvironment{theglossary}%
8945   {\tablehead{}\tabletail{}}%
8946   \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
8947   {\end{supertabular}}%
8948 }
```

`super4colheader` The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8949 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostylesuper4colheader` style:

```
8950 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8951 \renewenvironment{theglossary}%
8952   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8953             \bfseries\symbolname &
8954             \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8955   \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
8956   {\end{supertabular}}%
8957 }
```


`super4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```
8958 \newglossarystyle{altsuper4colborder}{%
```

Base it on the `glostylesuper4colborder` style:

```
8959 \setglossarystyle{super4colborder}{%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8960 \renewenvironment{theglossary}{%
8961   {\tablehead{\hline}\tabletail{\hline}%
8962   \begin{supertabular}%
8963     {|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
8964   {\end{supertabular}}%
8965 }
```

`colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```
8966 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
8967 \setglossarystyle{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:

```
8968 \renewenvironment{theglossary}{%
8969   {\tablehead{\hline
8970     \bfseries\entryname &
8971     \bfseries\descriptionname &
8972     \bfseries\symbolname &
8973     \bfseries\pagelistname\tabularnewline\hline}%
8974   \tabletail{\hline}%
8975   \begin{supertabular}%
8976     {|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
8977   {\end{supertabular}}%
8978 }
```

3.9 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.

```
8979 \ProvidesPackage{glossary-superragged}[2016/06/09 v4.25 (NLCT)]
```

Requires the package:

```
8980 \RequirePackage{array}
```

Requires the package:

```
8981 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
8982 \@ifundefined{glsdescwidth}{%
8983   \newlength\glsdescwidth
8984   \setlength{\glsdescwidth}{0.6\hsize}
8985 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8986 \@ifundefined{glspagelistwidth}{%
8987   \newlength\glspagelistwidth
8988   \setlength{\glspagelistwidth}{0.1\hsize}
8989 }{}
```

`superragged` The superragged glossary style uses the supertabular environment.

```
8990 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8991   \renewenvironment{theglossary}%
8992     {\tablehead{}\tabletail{}}%
8993     \begin{supertabular}[1>{\raggedright}p{\glsdescwidth}]{%
8994       {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8995   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8996   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8997   \renewcommand{\glossentry}[2]{%
8998     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8999     \glossentrydesc{##1}\glspostdescription\space ##2%
9000     \tabularnewline
9001   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9002   \renewcommand{\subglossentry}[3]{%
9003     &
9004     \glssubentryitem{##2}%
9005     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9006     ##3%
9007     \tabularnewline
9008   }%
```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9009   \ifglsnogroupskip
9010     \renewcommand*{\glsgroupskip}{}%
9011   \else
```

```

9012 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9013 \fi
9014 }

```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```

9015 \newglossarystyle{superraggedborder}{%
    Base it on the glostylesuperragged style:
9016 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns and a horizontal line in the
    head and tail:
9017 \renewenvironment{theglossary}%
9018 {\tablehead{\hline}\tabletail{\hline}%
9019 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
9020 {\end{supertabular}}%
9021 }

```

`superraggedheader` The `superraggedheader` style is like the super style, but with a header:

```

9022 \newglossarystyle{superraggedheader}{%
    Base it on the glostylesuperragged style:
9023 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and no tail:
9024 \renewenvironment{theglossary}%
9025 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
9026 \tabularnewline}%
9027 \tabletail{}}%
9028 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
9029 {\end{supertabular}}%
9030 }

```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```

9031 \newglossarystyle{superraggedheaderborder}{%
    Base it on the glostylesuper style:
9032 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and horizontal
    lines above and below the table:
9033 \renewenvironment{theglossary}%
9034 {\tablehead{\hline\bfseries \entryname &
9035 \bfseries \descriptionname\tabletail{\hline}}%
9036 \tabletail{\hline}}%
9037 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
9038 {\end{supertabular}}%
9039 }

```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```

9040 \newglossarystyle{superragged3col}{%

```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
9041 \renewenvironment{theglossary}%
9042   {\tablehead{}\tabletail{}}%
9043   \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
9044     >{\raggedright}p{\glspagelistwidth}}%
9045   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9046 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9047 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9048 \renewcommand{\glossentry}[2]{%
9049   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9050   \glossentrydesc{##1} &
9051   ##2\tabularnewline
9052 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
9053 \renewcommand{\subglossentry}[3]{%
9054   &
9055   \glssubentryitem{##2}%
9056   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9057   ##3\tabularnewline
9058 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9059 \ifglsnogroupskip
9060   \renewcommand*{\glsgroupskip}{}%
9061 \else
9062   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9063 \fi
9064 }
```

`ragged3colborder` The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
9065 \newglossarystyle{superragged3colborder}{}%
```

Base it on the `glostylesuperragged3col` style:

```
9066 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
9067 \renewenvironment{theglossary}%
9068   {\tablehead{\hline}\tabletail{\hline}%
9069   \begin{supertabular}{|1|>{\raggedright}p{\glsdescwidth}|%
9070     >{\raggedright}p{\glspagelistwidth}|}%
9071   {\end{supertabular}}%
9072 }
```

ragged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
9073 \newglossarystyle{superragged3colheader}{%
```

Base it on the glostylesuperragged3col style:

```
9074 \setglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
9075 \renewenvironment{theglossary}{%
9076   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9077     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9078   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9079     >{\raggedright}p{\glspagelistwidth}}}%
9080   {\end{supertabular}}%
9081 }
```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9082 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9083 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9084 \renewenvironment{theglossary}{%
9085   {\tablehead{\hline
9086     \bfseries\entryname&\bfseries\descriptionname&
9087     \bfseries\pagelistname\tabularnewline\hline}%
9088   \tabletail{\hline}%
9089   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9090     >{\raggedright}p{\glspagelistwidth}|}}%
9091   {\end{supertabular}}%
9092 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9093 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9094 \renewenvironment{theglossary}{%
9095   {\tablehead{} \tabletail{}}%
9096   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9097     >{\raggedright}p{\glspagelistwidth}}}%
9098   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9099 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9100 \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:

```
9101 \renewcommand{\glossentry}[2]{%
9102   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9103   \glossentrydesc{##1} &
9104   \glossentrysymbol{##1} & ##2\tabularnewline
9105 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9106 \renewcommand{\subglossentry}[3]{%
9107   &
9108   \glssubentryitem{##2}%
9109   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9110   \glossentrysymbol{##2} & ##3\tabularnewline
9111 }%
```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9112 \ifglsgroupskip
9113   \renewcommand*{\glsgroupskip}{}%
9114 \else
9115   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9116 \fi
9117 }
```

`ragged4colheader` The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```
9118 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
9119 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
9120 \renewenvironment{theglossary}{%
9121   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9122     \bfseries\symbolname &
9123     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9124   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9125     >{\raggedright}p{\glspagelistwidth}}}%
9126   {\end{supertabular}}}%
9127 }
```

`ragged4colborder` The `altsuperragged4colborder` style is like the `altsuperragged4col` style but with a border.

```
9128 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
9129 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
9130 \renewenvironment{theglossary}{%
```

```

9131   {\tablehead{\hline}\tabletail{\hline}%
9132   \begin{supertabular}%
9133     {||>{\raggedright}p{\glstdescwidth}||%
9134     >{\raggedright}p{\glspagelistwidth}||}}%
9135   {\end{supertabular}}%
9136 }

```

`colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```

9137 \newglossarystyle{altsuperragged4colheaderborder}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

9138 \setglossarystyle{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:

```

9139 \renewenvironment{theglossary}%
9140   {\tablehead{\hline
9141     \bfseries\entryname &
9142     \bfseries\descriptionname &
9143     \bfseries\symbolname &
9144     \bfseries\pagelistname\tabularnewline\hline}%
9145   \tabletail{\hline}%
9146   \begin{supertabular}%
9147     {||>{\raggedright}p{\glstdescwidth}||%
9148     >{\raggedright}p{\glspagelistwidth}||}}%
9149   {\end{supertabular}}%
9150 }

```

3.10 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```

9151 \ProvidesPackage{glossary-tree}[2016/06/09 v4.25 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9152 \providecommand{\indexspace}{%
9153   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9154 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command was previously also used to format the group headings.

```

9155 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}

```

`egrouphaderfmt` Format used to display the group header in the tree styles. Before v4.22, `\glstreenamefmt` was used for the group header, so the default definition uses that to help maintain backward-compatibility, since in previous versions redefining `\glstreenamefmt` would've also affected the group headings.

```
9156 \newcommand*\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}
```

`eenavigationfmt` Format used to display the navigation header in the tree styles.

```
9157 \newcommand*\glstreenavigationfmt}[1]{\glstreenamefmt{#1}}
```

`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.

```
9158 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9159 \renewenvironment{theglossary}{%
9160   {\setlength{\parindent}{0pt}}%
9161   \setlength{\parskip}{0pt plus 0.3pt}}%
9162   \let\item\@idxitem}%
```

```
9163   {\par}}%
```

Do nothing at the start of the environment:

```
9164 \renewcommand*\glossaryheader}{}
```

No group headers:

```
9165 \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.

```
9166 \renewcommand*\glossentry}[2]{%
9167   \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9168   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9169   \space \glossentrydesc{##1}\glspostdescription\space ##2%
9170 }%
```

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 `\glossentry`, 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.

```
9171 \renewcommand*\subglossentry}[3]{%
```

```
9172   \ifcase##1\relax
```

```
9173     % level 0
```

```
9174     \item
```

```
9175   \or
```

```
9176     % level 1
```

```
9177     \subitem
```



```

9178     \glssubentryitem{##2}%
9179   \else
9180     % all other levels
9181     \subsubitem
9182   \fi
9183   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9184   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9185   \space\glossentrydesc{##2}\glspostdescription\space ##3%
9186 }%

```

Vertical gap between groups is the same as that used by indices:

```

9187 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}

```

indexgroup The `indexgroup` style is like the `index` style but has headings.

```

9188 \newglossarystyle{indexgroup}{%

```

Base it on the `glostyleindex` style:

```

9189 \setglossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9190 \renewcommand*\glsgroupheading}[1]{%
9191   \item\glstreegroupheaderfmt{\glsggetgrouptitle{##1}}%
9192   \indexspace
9193 }%
9194 }

```

indexhypergroup The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9195 \newglossarystyle{indexhypergroup}{%

```

Base it on the `glostyleindex` style:

```

9196 \setglossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

9197 \renewcommand*\glossaryheader}{%
9198   \item\glstreenavigationfmt{\glsnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

9199 \renewcommand*\glsgroupheading}[1]{%
9200   \item\glstreegroupheaderfmt
9201     {\glsnavhypertarget{##1}{\glsggetgrouptitle{##1}}}%
9202   \indexspace}%
9203 }

```

tree The `tree` glossary style is similar in style to the `index` style, but can have arbitrary levels.

```

9204 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9205 \renewenvironment{theglossary}%
9206   {\setlength{\parindent}{0pt}%
9207   \setlength{\parskip}{0pt plus 0.3pt}}%
9208   {}%

```

Do nothing at the start of the theglossary environment:

```
9209 \renewcommand*{\glossaryheader}{}
```

No group headings:

```
9210 \renewcommand*{\glsgroupheading}[1]{}
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
9211 \renewcommand{\glossentry}[2]{%
9212   \hangindent0pt\relax
9213   \parindent0pt\relax
9214   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9215   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9216   \space\glossentrydesc{##1}\glspostdescription\space##2\par
9217 }
```

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.

```
9218 \renewcommand{\subglossentry}[3]{%
9219   \hangindent##1\glstreeindent\relax
9220   \parindent##1\glstreeindent\relax
9221   \ifnum##1=1\relax
9222     \glssubentryitem{##2}%
9223     \fi
9224   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9225   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9226   \space\glossentrydesc{##2}\glspostdescription\space ##3\par
9227 }
```

Vertical gap between groups is the same as that used by indices:

```
9228 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

treegroup Like the tree style but the glossary groups have headings.

```
9229 \newglossarystyle{treegroup}{%
```

Base it on the `glostyletree` style:

```
9230 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9231 \renewcommand{\glsgroupheading}[1]{\par
9232   \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par
9233   \indexspace}%
9234 }
```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9235 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```
9236 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
9237 \renewcommand*{\glossaryheader}{%
9238   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9239 \renewcommand*{\glsgroupheading}[1]{%
9240   \par\noindent
9241   \glstreegroupheaderfmt
9242   {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9243   \indexspace}%
9244 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```
9245 \newlength\glstreeindent
9246 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
9247 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9248 \renewenvironment{theglossary}%
9249   {\setlength{\parindent}{0pt}%
9250   \setlength{\parskip}{0pt plus 0.3pt}}%
9251   {}%
```

No header:

```
9252 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9253 \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.

```
9254 \renewcommand{\glossentry}[2]{%
9255   \hangindent0pt\relax
9256   \parindent0pt\relax
9257   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9258   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9259   \space\glossentrydesc{##1}\glspostdescription\space##2\par
9260   }%
```

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.

```
9261 \renewcommand{\subglossentry}[3]{%
9262   \hangindent##1\glstreeindent\relax
9263   \parindent##1\glstreeindent\relax
9264   \ifnum##1=1\relax
9265     \glssubentryitem{##2}%
9266     \fi
9267   \glstarget{##2}{\strut}%
```

```

9268   \glossentrydesc{##2}\glspostdescription\space##3\par
9269 }%

```

Vertical gap between groups is the same as that used by indices:

```

9270 \renewcommand*\glsgroupskip{\ifglsgnogroupskip\else\indexspace\fi}%
9271 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9272 \newglossarystyle{treenonamegroup}{%

```

Base it on the `glostyletreenoname` style:

```

9273 \setglossarystyle{treenoname}%

```

Give each group a heading:

```

9274 \renewcommand{\glsgroupheading}[1]{\par
9275   \noindent\glstreegroupheaderfmt
9276   {\glsggetgrouptitle{##1}}\par\indexspace}%
9277 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9278 \newglossarystyle{treenonamehypergroup}{%

```

Base it on the `glostyletreenoname` style:

```

9279 \setglossarystyle{treenoname}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

9280 \renewcommand*\glossaryheader}{%
9281   \par\noindent\glstreenavigationfmt{\glsgnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9282 \renewcommand*\glsgroupheading}[1]{%
9283   \par\noindent
9284   \glstreegroupheaderfmt
9285   {\glsgnavhypertarget{##1}}{\glsggetgrouptitle{##1}}\par
9286   \indexspace}%
9287 }

```

`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9288 \newrobustcmd*\glsgfindwidesttoplevelname}[1][\@glo@types]{%
9289   \dimen@=0pt\relax
9290   \glsg@tmplen=0pt\relax
9291   \forallglossaries[#1]{\@gls@type}%
9292   {%
9293     \forallglsentries[\@gls@type]{\@glo@label}%
9294     {%
9295       \ifglshasparent{\@glo@label}%
9296       }%
9297     }%
9298     \settowidth{\dimen@}%
9299     {\glgstreenamefmt{\glsgsentryname{\@glo@label}}}%

```

```

9300     \ifdim\dimen@>\gls@tmplen
9301     \gls@tmplen=\dimen@
9302     \letcs{\@glswidestname}{glo@\glsdetoklabel{\@glo@label}@name}%
9303     \fi
9304   }%
9305 }%
9306 }%
9307 }

```

`\glssetwidest` `\glssetwidest` [*level*] {*text*} sets the widest text for the given level. It is used by the alt-tree glossary styles to determine the indentation of each level.

```

9308 \newcommand*\glssetwidest}[2][0]{%
9309   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9310     #2}%
9311 }

```

`\@glswidestname` Initialise `\@glswidestname`.

```

9312 \newcommand*\@glswidestname{}

```

`\glsstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9313 \newcommand*\glsstreenamebox}[2]{%
9314   \makebox[#1][l]{#2}%
9315 }

```

`alttree` The alttree glossary style is similar in style to the tree style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glssetwidest`.

```

9316 \newglossarystyle{alttree}{%

```

Redefine theglossary environment.

```

9317   \renewenvironment{theglossary}%
9318     {\def\@gls@prevlevel{-1}%
9319     \mbox{}\par}%
9320     {\par}%

```

Set the header and group headers to nothing.

```

9321   \renewcommand*\glossaryheader{}%
9322   \renewcommand*\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```

9323   \renewcommand*\glossentry}[2]{%
9324     \ifnum\@gls@prevlevel=0\relax
9325     \else

```

Find out how big the indentation should be by measuring the widest entry.

```

9326     \settowidth{\glsstreeindent}{\glsstreenamefmt{\@glswidestname\space}}%
9327     \fi

```

Set the hangindent and paragraph indent.

```

9328     \hangindent\glsstreeindent
9329     \parindent\glsstreeindent

```

Put the name to the left of the paragraph block.

```
9330 \makebox[Opt][r]{\glstreenamebox{\glstreeindent}{%
9331 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9332 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9333 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9334 \def\@gls@prevlevel{0}%
9335 }%
```

Redefine the way sub-entries are displayed.

```
9336 \renewcommand{\subglossentry}[3]{%
```

Increment and display the sub-entry counter if this is a level 1 entry and the sub-entry counter is in use.

```
9337 \ifnum##1=1\relax
9338 \glssubentryitem{##2}%
9339 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9340 \ifnum\@gls@prevlevel=##1\relax
9341 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9342 \@ifundefined{@glswidestname\romannumeral##1}{%
9343 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}}%
9344 \settowidth{\gls@tmplen}{\glstreenamefmt{%
9345 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9346 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9347 \setlength\glstreeindent\gls@tmplen
9348 \addtolength\glstreeindent\parindent
9349 \parindent\glstreeindent
9350 \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`.

```
9351 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9352 \settowidth{\glstreeindent}{\glstreenamefmt{%
9353 \@glswidestname\space}}}%
9354 \settowidth{\glstreeindent}{\glstreenamefmt{%
9355 \csname @glswidestname\romannumeral\@gls@prevlevel
9356 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9357      \addtolength\parindent{-\glstreeindent}%
9358      \setlength\glstreeindent\parindent
9359      \fi
9360      \fi
```

Set the hanging indentation.

```
9361      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9362      \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
9363      \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9364      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9365      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9366      \def\@gls@prevlevel{##1}%
9367      }%
```

Vertical gap between groups is the same as that used by indices:

```
9368      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9369      }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
9370 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9371      \setglossarystyle{almtree}%
```

Give each group a heading.

```
9372      \renewcommand{\glsgroupheading}[1]{\par
9373      \def\@gls@prevlevel{-1}%
9374      \hangindent0pt\relax
9375      \parindent0pt\relax
9376      \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
9377      \par\indexspace}%
9378      }
```

`almtreehypergroup` The `almtreehypergroup` style is like the `almtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
9379 \newglossarystyle{almtreehypergroup}{%
```

Base it on the `glostylealmtree` style:

```
9380      \setglossarystyle{almtree}%
```

Put the navigation links in the header

```
9381      \renewcommand*{\glossaryheader}{%
9382      \par
```

```
9383 \def\@gls@prevlevel{-1}%
9384 \hangindent0pt\relax
9385 \parindent0pt\relax
9386 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
9387 \renewcommand*\@gls@groupheading[1]{%
9388 \par
9389 \def\@gls@prevlevel{-1}%
9390 \hangindent0pt\relax
9391 \parindent0pt\relax
9392 \glstreegroupheaderfmt
9393 {\@glsnavhypertarget{##1}{\@glsgetgrouptitle{##1}}}\par
9394 \indexspace}}
```


4 Backwards Compatibility

4.1 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.

```
9395 \NeedsTeXFormat{LaTeX2e}
9396 \ProvidesPackage{glossaries-compatible-207}[2016/06/09 v4.25 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9397 \ifglxsindy
9398   \renewcommand*\GlsAddXdyAttribute[1]{%
9399     \edef\xdyattributes{\xdyattributes ^^J \string"#1\string"}%
9400     \expandafter\toks@\expandafter{\xdylocref}%
9401     \edef\xdylocref{\the\toks@ ^^J%
9402     (markup-locref
9403     :open \string"\string~n\string\setentrycounter
9404     {\noexpand\glscounter}%
9405     \expandafter\string\csname#1\endcsname
9406     \expandafter@gobble\string\{\string" ^^J
9407     :close \string"\expandafter@gobble\string}\string" ^^J
9408     :attr \string"#1\string")}}
```

Only has an effect before `\writeist`:

```
9409 \fi
```

sAddXdyCounters

```
9410 \renewcommand*\GlsAddXdyCounters[1]{%
9411   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9412   in compatibility mode.}%
9413 }
```

Add predefined attributes

```
9414 \GlsAddXdyAttribute{glsnumberformat}
9415 \GlsAddXdyAttribute{textrm}
9416 \GlsAddXdyAttribute{textsf}
9417 \GlsAddXdyAttribute{texttt}
9418 \GlsAddXdyAttribute{textbf}
9419 \GlsAddXdyAttribute{textmd}
9420 \GlsAddXdyAttribute{textit}
9421 \GlsAddXdyAttribute{textup}
9422 \GlsAddXdyAttribute{textsl}
```

```

9423 \GlsAddXdyAttribute{textsc}
9424 \GlsAddXdyAttribute{emph}
9425 \GlsAddXdyAttribute{glshypernumber}
9426 \GlsAddXdyAttribute{hyperrm}
9427 \GlsAddXdyAttribute{hypersf}
9428 \GlsAddXdyAttribute{hypertt}
9429 \GlsAddXdyAttribute{hyperbf}
9430 \GlsAddXdyAttribute{hypermd}
9431 \GlsAddXdyAttribute{hyperit}
9432 \GlsAddXdyAttribute{hyperup}
9433 \GlsAddXdyAttribute{hypersl}
9434 \GlsAddXdyAttribute{hypersc}
9435 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9436 \ifglxindy
9437 \renewcommand*\GlsAddXdyLocation}[2]{%
9438   \edef\xdyuserlocationdefs{%
9439     \x dyuserlocationdefs ^^J%
9440     (define-location-class \string"#1\string"^^J\space\space
9441     \space(#2))
9442   }%
9443   \edef\xdyuserlocationnames{%
9444     \x dyuserlocationnames^^J\space\space\space
9445     \string"#1\string"%
9446   }
9447 \fi

```

\@do@wrglossary

```

9448 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9449 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9450 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9451 \def\@glo@range{}%
9452 \expandafter\if\@glo@prefix(\relax
9453   \def\@glo@range{:open-range}%
9454   \else
9455     \expandafter\if\@glo@prefix)\relax
9456     \def\@glo@range{:close-range}%
9457   \fi
9458 \fi

  Get the location and escape any special characters
9459 \protected@edef\@glslocref{\theglsentrycounter}%
9460 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9461 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9462 (indexentry :tkey (\csname glo@#1@index\endcsname)
9463   :locref \string"\@glslocref\string" %
9464   :attr \string"\@glo@suffix\string" \@glo@range
9465 )
9466 }%
9467 \else

```

Convert the format information into the format required for makeindex

```

9468 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9469 \glossary[\csname glo@#1@type\endcsname]{%
9470 \string\glossaryentry{\csname glo@#1@index\endcsname
9471   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9472 \fi
9473 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9474 \def\@set@glo@numformat#1#2#3{%
9475   \expandafter\@glo@check@mkidxrangechar#3\@nil
9476   \protected@edef#1{%
9477     \@glo@prefix setentrycounter[] {#2}%
9478     \expandafter\string\csname\@glo@suffix\endcsname
9479   }%
9480   \@gls@checkmkidxchars#1%
9481 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9482 \ifglxindy
9483 \def\writeist{%
9484   \openout\glswrite=\istfilename
9485   \write\glswrite{;; xindy style file created by the glossaries
9486     package in compatible-2.07 mode}%
9487   \write\glswrite{;; for document '\jobname' on
9488     \the\year-\the\month-\the\day}%
9489   \write\glswrite{^^J; required styles^^J}
9490   \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9491     \ifx\@xdystyle\@empty
9492     \else
9493       \protected@write\glswrite{{(require
9494         \string"\@xdystyle.xdy\string")}}%
9495     \fi
9496   }%
9497   \write\glswrite{^^J%
9498     ; list of allowed attributes (number formats)^^J}%
9499   \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9500   \write\glswrite{^^J; user defined alphabets^^J}%
9501   \write\glswrite{\@xdyuseralphabets}%
9502   \write\glswrite{^^J; location class definitions^^J}%
9503   \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9504     \string"roman-numbers-lowercase\string" :sep \string}}}%
9505 \@onelevel@sanitize\@gls@roman
9506 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9507     :sep \string}}}%
9508 \@onelevel@sanitize\@tmp
9509 \ifx\@tmp\@gls@roman
9510     \write\glswrite{(define-location-class
9511         \string"roman-page-numbers\string"^^J\space\space\space
9512         (\string"roman-numbers-lowercase\string")
9513         :min-range-length \@glsminrange)}}%
9514 \else
9515     \write\glswrite{(define-location-class
9516         \string"roman-page-numbers\string"^^J\space\space\space
9517         (:sep "\@gls@roman")
9518         :min-range-length \@glsminrange)}}%
9519 \fi
9520 \write\glswrite{(define-location-class
9521     \string"Roman-page-numbers\string"^^J\space\space\space
9522     (\string"roman-numbers-uppercase\string")
9523     :min-range-length \@glsminrange)}}%
9524 \write\glswrite{(define-location-class
9525     \string"arabic-page-numbers\string"^^J\space\space\space
9526     (\string"arabic-numbers\string")
9527     :min-range-length \@glsminrange)}}%
9528 \write\glswrite{(define-location-class
9529     \string"alpha-page-numbers\string"^^J\space\space\space
9530     (\string"alpha\string")
9531     :min-range-length \@glsminrange)}}%
9532 \write\glswrite{(define-location-class
9533     \string"Alpha-page-numbers\string"^^J\space\space\space
9534     (\string"ALPHA\string")
9535     :min-range-length \@glsminrange)}}%
9536 \write\glswrite{(define-location-class
9537     \string"Appendix-page-numbers\string"^^J\space\space\space
9538     (\string"ALPHA\string"
9539     :sep \string"\@glsAlphacompositor\string"
9540     \string"arabic-numbers\string")
9541     :min-range-length \@glsminrange)}}%
9542 \write\glswrite{(define-location-class
9543     \string"arabic-section-numbers\string"^^J\space\space\space
9544     (\string"arabic-numbers\string"
9545     :sep \string"\glscompositor\string"
9546     \string"arabic-numbers\string")
9547     :min-range-length \@glsminrange)}}%
9548 \write\glswrite{^^J; user defined location classes}%
9549 \write\glswrite{\@xdyuserlocationdefs}%
9550 \write\glswrite{^^J; define cross-reference class^^J}%
9551 \write\glswrite{(define-crossref-class \string"see\string"
9552     :unverified )}%

```

```

9553 \write\glswrite{(markup-crossref-list
9554   :class \string"see\string"^^J\space\space\space
9555   :open \string"\string\glsseeformat\string"
9556   :close \string"{}\string")}%
9557 \write\glswrite{^^J; define the order of the location classes}%
9558 \write\glswrite{(define-location-class-order
9559   (\@xdylocationclassorder))}%
9560 \write\glswrite{^^J; define the glossary markup^^J}%
9561 \write\glswrite{(markup-index^^J\space\space\space
9562   :open \string"\string
9563     \glossarysection[\string\glossarytoctitle]{\string
9564     \glossarytitle}\string\glossarypreamble\string~n\string\begin
9565     {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9566     \space\space:close \string"\expandafter\@gobble
9567     \string%\string~n\string
9568     \end{theglossary}\string\glossarypostamble
9569     \string~n\string" ^^J\space\space\space
9570   :tree)}}%
9571 \write\glswrite{(markup-letter-group-list
9572   :sep \string"\string\glsgroupskip\string~n\string")}%
9573 \write\glswrite{(markup-indexentry
9574   :open \string"\string\relax \string\glsresetentrylist
9575     \string~n\string")}%
9576 \write\glswrite{(markup-locclass-list :open
9577   \string"\glsopenbrace\string\glossaryentrynumbers
9578     \glsopenbrace\string\relax\space \string"^^J\space\space\space
9579   :sep \string", \string"
9580   :close \string"\glsclosebrace\glsclosebrace\string")}%
9581 \write\glswrite{(markup-locref-list
9582   :sep \string"\string\delimN\space\string")}%
9583 \write\glswrite{(markup-range
9584   :sep \string"\string\delimR\space\string")}%
9585 \@onelevel@sanitize\gls@suffixF
9586 \@onelevel@sanitize\gls@suffixFF
9587 \ifx\gls@suffixF\@empty
9588 \else
9589   \write\glswrite{(markup-range
9590     :close "\gls@suffixF" :length 1 :ignore-end)}%
9591 \fi
9592 \ifx\gls@suffixFF\@empty
9593 \else
9594   \write\glswrite{(markup-range
9595     :close "\gls@suffixFF" :length 2 :ignore-end)}%
9596 \fi
9597 \write\glswrite{^^J; define format to use for locations^^J}%
9598 \write\glswrite{\@xdylocref}%
9599 \write\glswrite{^^J; define letter group list format^^J}%
9600 \write\glswrite{(markup-letter-group-list
9601   :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9602 \write\glswrite{^^J; letter group headings^^J}%
9603 \write\glswrite{(markup-letter-group
9604 :open-head \string"\string\glsgroupheading
9605 \glsopenbrace\string"^^J\space\space\space
9606 :close-head \string"\glsclosebrace\string")}%
9607 \write\glswrite{^^J; additional letter groups^^J}%
9608 \write\glswrite{\@xdylettergroups}%
9609 \write\glswrite{^^J; additional sort rules^^J}
9610 \write\glswrite{\@xdysortrules}%
9611 \noist}
9612 \else
9613 \edef\@gls@actualchar{\string?}
9614 \edef\@gls@encapchar{\string|}
9615 \edef\@gls@levelchar{\string!}
9616 \edef\@gls@quotechar{\string"}
9617 \def\writeist{\relax
9618 \openout\glswrite=\istfilename
9619 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9620 created by the glossaries package}
9621 \write\glswrite{\expandafter\@gobble\string\% for document
9622 'jobname' on \the\year-\the\month-\the\day}
9623 \write\glswrite{actual '@gls@actualchar'}
9624 \write\glswrite{encap '@gls@encapchar'}
9625 \write\glswrite{level '@gls@levelchar'}
9626 \write\glswrite{quote '@gls@quotechar'}
9627 \write\glswrite{keyword \string"\string\glossaryentry\string"}
9628 \write\glswrite{preamble \string"\string\glossarysection[\string
9629 \glossarytoctitle]{\string\glossarytitle}\string
9630 \glossarypreamble\string\n\string\begin{theglossary}\string
9631 \glossaryheader\string\n\string"}
9632 \write\glswrite{postamble \string"\string%\string\n\string
9633 \end{theglossary}\string\glossarypostamble\string\n
9634 \string"}
9635 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9636 \string"}
9637 \write\glswrite{item_0 \string"\string%\string\n\string"}
9638 \write\glswrite{item_1 \string"\string%\string\n\string"}
9639 \write\glswrite{item_2 \string"\string%\string\n\string"}
9640 \write\glswrite{item_01 \string"\string%\string\n\string"}
9641 \write\glswrite{item_x1
9642 \string"\string\relax \string\glsresetentrylist\string\n
9643 \string"}
9644 \write\glswrite{item_12 \string"\string%\string\n\string"}
9645 \write\glswrite{item_x2
9646 \string"\string\relax \string\glsresetentrylist\string\n
9647 \string"}
9648 \write\glswrite{delim_0 \string"\string{\string
9649 \glossaryentrynumbers\string\{\string\relax \string"}
9650 \write\glswrite{delim_1 \string"\string{\string

```

```

9651     \glossaryentrynumbers\string\{\string\relax \string}
9652 \write\glswrite{delim_2 \string}\string\{\string
9653     \glossaryentrynumbers\string\{\string\relax \string}
9654 \write\glswrite{delim_t \string}\string}\string}\string}
9655 \write\glswrite{delim_n \string}\string\delimN \string}
9656 \write\glswrite{delim_r \string}\string\delimR \string}
9657 \write\glswrite{headings_flag 1}
9658 \write\glswrite{heading_prefix
9659     \string}\string\glsgroupheading\string\{\string}
9660 \write\glswrite{heading_suffix
9661     \string}\string}\string\relax
9662     \string\glsresetentrylist \string}
9663 \write\glswrite{symhead_positive \string"glssymbols\string"}
9664 \write\glswrite{numhead_positive \string"glnumbers\string"}
9665 \write\glswrite{page_compositor \string"glscpositor\string"}
9666 \@gls@escbsdq\gls@suffixF
9667 \@gls@escbsdq\gls@suffixFF
9668 \ifx\gls@suffixF\@empty
9669 \else
9670     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
9671 \fi
9672 \ifx\gls@suffixFF\@empty
9673 \else
9674     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
9675 \fi
9676 \noist
9677 }
9678 \fi
\noist
9679 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9680 \NeedsTeXFormat{LaTeX2e}
9681 \ProvidesPackage{glossaries-compatible-307}[2016/06/09 v4.25 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9682 \newcommand{\compatglossarystyle}[2]{%
9683   \ifcsundef{@glscompstyle@#1}%
9684   {%
9685     \csdef{@glscompstyle@#1}{#2}%
9686   }%
9687   {%
9688     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{%
9689   }%
9690 }

```

Backward compatible inline style.

```
9691 \compatglossarystyle{inline}{%
9692   \renewcommand{\glossaryentryfield}[5]{%
9693     \glsinlinedopostchild
9694     \gls@inlinesep
9695     \def\glo@desc{##3}%
9696     \def\@no@post@desc{\nopostdesc}%
9697     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9698     \ifx\glo@desc\@no@post@desc
9699       \glsinlineemptydescformat{##4}{##5}%
9700     \else
9701       \ifstrempy{##3}%
9702         {\glsinlineemptydescformat{##4}{##5}}%
9703         {\glsinlinedescformat{##3}{##4}{##5}}%
9704     \fi
9705     \ifglshaschildren{##1}%
9706     {%
9707       \glsresetsubentrycounter
9708       \glsinlineparentchildseparator
9709       \def\gls@inlinesubsep{}%
9710       \def\gls@inlinepostchild{\glsinlinepostchild}%
9711     }%
9712     {}%
9713     \def\gls@inlinesep{\glsinlineseparator}%
9714   }%
```

Sub-entries display description:

```
9715 \renewcommand{\glossarysubentryfield}[6]{%
9716   \gls@inlinesubsep%
9717   \glsinlinesubnameformat{##2}{##3}%
9718   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9719   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9720 }%
9721 }
```

Backward compatible list style.

```
9722 \compatglossarystyle{list}{%
9723   \renewcommand*{\glossaryentryfield}[5]{%
9724     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9725     ##3\glspostdescription\space ##5}%
9726 }
```

Sub-entries continue on the same line:

```
9726 \renewcommand*{\glossarysubentryfield}[6]{%
9727   \glssubentryitem{##2}%
9728   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9729 }
```

Backward compatible listgroup style.

```
9730 \compatglossarystyle{listgroup}{%
9731   \csuse{@glscompstyle@list}%
9732 }%
```


Backward compatible listhypergroup style.

```
9733 \compatglossarystyle{listhypergroup}{%
9734 \csuse{@glscompstyle@list}%
9735 }%
```

Backward compatible altlist style.

```
9736 \compatglossarystyle{altlist}{%
9737 \renewcommand*{\glossaryentryfield}[5]{%
9738 \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
9739 \mbox{}\par\nobreak\@afterheading
9740 ##3\glspostdescription\space ##5}%
9741 \renewcommand{\glossarysubentryfield}[6]{%
9742 \par
9743 \glssubentryitem{##2}%
9744 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9745 }%
```

Backward compatible altlistgroup style.

```
9746 \compatglossarystyle{altlistgroup}{%
9747 \csuse{@glscompstyle@altlist}%
9748 }%
```

Backward compatible altlisthypergroup style.

```
9749 \compatglossarystyle{altlisthypergroup}{%
9750 \csuse{@glscompstyle@altlist}%
9751 }%
```

Backward compatible listdotted style.

```
9752 \compatglossarystyle{listdotted}{%
9753 \renewcommand*{\glossaryentryfield}[5]{%
9754 \item[]\makebox[\glslistdottedwidth][l]{%
9755 \glsentryitem{##1}\glstarget{##1}{##2}%
9756 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9757 \renewcommand*{\glossarysubentryfield}[6]{%
9758 \item[]\makebox[\glslistdottedwidth][l]{%
9759 \glssubentryitem{##2}%
9760 \glstarget{##2}{##3}%
9761 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9762 }%
```

Backward compatible sublistdotted style.

```
9763 \compatglossarystyle{sublistdotted}{%
9764 \csuse{@glscompstyle@listdotted}%
9765 \renewcommand*{\glossaryentryfield}[5]{%
9766 \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
9767 }%
```

Backward compatible long style.

```
9768 \compatglossarystyle{long}{%
9769 \renewcommand*{\glossaryentryfield}[5]{%
9770 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\}%
9771 \renewcommand*{\glossarysubentryfield}[6]{%

```

```

9772      &
9773      \glssubentryitem{##2}%
9774      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9775 }%

```

Backward compatible longborder style.

```

9776 \compatglossarystyle{longborder}{%
9777 \csuse{@glscompstyle@long}%
9778 }%

```

Backward compatible longheader style.

```

9779 \compatglossarystyle{longheader}{%
9780 \csuse{@glscompstyle@long}%
9781 }%

```

Backward compatible longheaderborder style.

```

9782 \compatglossarystyle{longheaderborder}{%
9783 \csuse{@glscompstyle@long}%
9784 }%

```

Backward compatible long3col style.

```

9785 \compatglossarystyle{long3col}{%
9786 \renewcommand*{\glossaryentryfield}[5]{%
9787 \glstarget{##1}{\strut}##4 & ##3 & ##5\\}%
9788 \renewcommand*{\glossarysubentryfield}[6]{%
9789 &
9790 \glssubentryitem{##2}%
9791 \glstarget{##2}{\strut}##4 & ##6\\}%
9792 }%

```

Backward compatible long3colborder style.

```

9793 \compatglossarystyle{long3colborder}{%
9794 \csuse{@glscompstyle@long3col}%
9795 }%

```

Backward compatible long3colheader style.

```

9796 \compatglossarystyle{long3colheader}{%
9797 \csuse{@glscompstyle@long3col}%
9798 }%

```

Backward compatible long3colheaderborder style.

```

9799 \compatglossarystyle{long3colheaderborder}{%
9800 \csuse{@glscompstyle@long3col}%
9801 }%

```

Backward compatible long4col style.

```

9802 \compatglossarystyle{long4col}{%
9803 \renewcommand*{\glossaryentryfield}[5]{%
9804 \glstarget{##1}{\strut}##4 & ##3 & ##4 & ##5\\}%
9805 \renewcommand*{\glossarysubentryfield}[6]{%
9806 &
9807 \glssubentryitem{##2}%

```

```

9808     \glstarget{##2}{\strut}##4 & ##5 & ##6\}%
9809 }%

  Backward compatible long4colheader style.
9810 \compatglossarystyle{long4colheader}{%
9811   \csuse{@glscompstyle@long4col}%
9812 }%

  Backward compatible long4colborder style.
9813 \compatglossarystyle{long4colborder}{%
9814   \csuse{@glscompstyle@long4col}%
9815 }%

  Backward compatible long4colheaderborder style.
9816 \compatglossarystyle{long4colheaderborder}{%
9817   \csuse{@glscompstyle@long4col}%
9818 }%

  Backward compatible altlong4col style.
9819 \compatglossarystyle{altlong4col}{%
9820   \csuse{@glscompstyle@long4col}%
9821 }%

  Backward compatible altlong4colheader style.
9822 \compatglossarystyle{altlong4colheader}{%
9823   \csuse{@glscompstyle@long4col}%
9824 }%

  Backward compatible altlong4colborder style.
9825 \compatglossarystyle{altlong4colborder}{%
9826   \csuse{@glscompstyle@long4col}%
9827 }%

  Backward compatible altlong4colheaderborder style.
9828 \compatglossarystyle{altlong4colheaderborder}{%
9829   \csuse{@glscompstyle@long4col}%
9830 }%

  Backward compatible long style.
9831 \compatglossarystyle{longragged}{%
9832   \renewcommand*{\glossaryentryfield}[5]{%
9833     \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9834     \tabularnewline}%
9835   \renewcommand*{\glossarysubentryfield}[6]{%
9836     &
9837     \glssubentryitem{##2}%
9838     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9839     \tabularnewline}%
9840 }%

  Backward compatible longraggedborder style.
9841 \compatglossarystyle{longraggedborder}{%
9842   \csuse{@glscompstyle@longragged}%
9843 }%

```

Backward compatible longraggedheader style.

```
9844 \compatglossarystyle{longraggedheader}{%  
9845 \csuse{@glscompstyle@longragged}%  
9846 }%
```

Backward compatible longraggedheaderborder style.

```
9847 \compatglossarystyle{longraggedheaderborder}{%  
9848 \csuse{@glscompstyle@longragged}%  
9849 }%
```

Backward compatible longragged3col style.

```
9850 \compatglossarystyle{longragged3col}{%  
9851 \renewcommand*{\glossaryentryfield}[5]{%  
9852 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9853 \renewcommand*{\glossarysubentryfield}[6]{%  
9854 &  
9855 \glssubentryitem{##2}%  
9856 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9857 }%
```

Backward compatible longragged3colborder style.

```
9858 \compatglossarystyle{longragged3colborder}{%  
9859 \csuse{@glscompstyle@longragged3col}%  
9860 }%
```

Backward compatible longragged3colheader style.

```
9861 \compatglossarystyle{longragged3colheader}{%  
9862 \csuse{@glscompstyle@longragged3col}%  
9863 }%
```

Backward compatible longragged3colheaderborder style.

```
9864 \compatglossarystyle{longragged3colheaderborder}{%  
9865 \csuse{@glscompstyle@longragged3col}%  
9866 }%
```

Backward compatible altlongragged4col style.

```
9867 \compatglossarystyle{altlongragged4col}{%  
9868 \renewcommand*{\glossaryentryfield}[5]{%  
9869 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9870 \renewcommand*{\glossarysubentryfield}[6]{%  
9871 &  
9872 \glssubentryitem{##2}%  
9873 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9874 }%
```

Backward compatible altlongragged4colheader style.

```
9875 \compatglossarystyle{altlongragged4colheader}{%  
9876 \csuse{@glscompstyle@altlong4col}%  
9877 }%
```

Backward compatible altlongragged4colborder style.

```
9878 \compatglossarystyle{altlongragged4colborder}{%
```

```
9879 \csuse{@glscompstyle@altlong4col}%
9880 }%
```

Backward compatible altlongragged4colheaderborder style.

```
9881 \compatglossarystyle{altlongragged4colheaderborder}{%
9882 \csuse{@glscompstyle@altlong4col}%
9883 }%
```

Backward compatible index style.

```
9884 \compatglossarystyle{index}{%
9885 \renewcommand*{\glossaryentryfield}[5]{%
9886 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9887 \ifx\relax##4\relax
9888 \else
9889 \space{##4}%
9890 \fi
9891 \space ##3\glspostdescription \space ##5}%
9892 \renewcommand*{\glossarysubentryfield}[6]{%
9893 \ifcase##1\relax
9894 % level 0
9895 \item
9896 \or
9897 % level 1
9898 \subitem
9899 \glssubentryitem{##2}%
9900 \else
9901 % all other levels
9902 \subsubitem
9903 \fi
9904 \textbf{\glstarget{##2}{##3}}%
9905 \ifx\relax##5\relax
9906 \else
9907 \space{##5}%
9908 \fi
9909 \space##4\glspostdescription\space ##6}%
9910 }%
```

Backward compatible indexgroup style.

```
9911 \compatglossarystyle{indexgroup}{%
9912 \csuse{@glscompstyle@index}%
9913 }%
```

Backward compatible indexhypergroup style.

```
9914 \compatglossarystyle{indexhypergroup}{%
9915 \csuse{@glscompstyle@index}%
9916 }%
```

Backward compatible tree style.

```
9917 \compatglossarystyle{tree}{%
9918 \renewcommand{\glossaryentryfield}[5]{%
9919 \hangindent0pt\relax
```

```

9920 \parindent0pt\relax
9921 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9922 \ifx\relax##4\relax
9923 \else
9924 \space{##4}%
9925 \fi
9926 \space ##3\glspostdescription \space ##5\par}%
9927 \renewcommand{\glossarysubentryfield}[6]{%
9928 \hangindent##1\glstreeindent\relax
9929 \parindent##1\glstreeindent\relax
9930 \ifnum##1=1\relax
9931 \glssubentryitem{##2}%
9932 \fi
9933 \textbf{\glstarget{##2}{##3}}%
9934 \ifx\relax##5\relax
9935 \else
9936 \space{##5}%
9937 \fi
9938 \space##4\glspostdescription\space ##6\par}%
9939 }%

```

Backward compatible treegroup style.

```

9940 \compatglossarystyle{treegroup}{%
9941 \csuse{@glscompstyle@tree}%
9942 }%

```

Backward compatible treehypergroup style.

```

9943 \compatglossarystyle{treehypergroup}{%
9944 \csuse{@glscompstyle@tree}%
9945 }%

```

Backward compatible treenoname style.

```

9946 \compatglossarystyle{treenoname}{%
9947 \renewcommand{\glossaryentryfield}[5]{%
9948 \hangindent0pt\relax
9949 \parindent0pt\relax
9950 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9951 \ifx\relax##4\relax
9952 \else
9953 \space{##4}%
9954 \fi
9955 \space ##3\glspostdescription \space ##5\par}%
9956 \renewcommand{\glossarysubentryfield}[6]{%
9957 \hangindent##1\glstreeindent\relax
9958 \parindent##1\glstreeindent\relax
9959 \ifnum##1=1\relax
9960 \glssubentryitem{##2}%
9961 \fi
9962 \glstarget{##2}{\strut}%
9963 ##4\glspostdescription\space ##6\par}%
9964 }%

```

Backward compatible treenonamegroup style.

```
9965 \compatglossarystyle{treenonamegroup}{%
9966 \csuse{@glscompstyle@treenoname}%
9967 }%
```

Backward compatible treenonamehypergroup style.

```
9968 \compatglossarystyle{treenonamehypergroup}{%
9969 \csuse{@glscompstyle@treenoname}%
9970 }%
```

Backward compatible altree style.

```
9971 \compatglossarystyle{almtree}{%
9972 \renewcommand{\glossaryentryfield}[5]{%
9973 \ifnum\@gls@prevlevel=0\relax
9974 \else
9975 \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9976 \hangindent\glstreeindent
9977 \parindent\glstreeindent
9978 \fi
9979 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9980 \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
9981 \ifx\relax##4\relax
9982 \else
9983 (##4)\space
9984 \fi
9985 ##3\glspostdescription \space ##5\par
9986 \def\@gls@prevlevel{0}%
9987 }%
9988 \renewcommand{\glossarysubentryfield}[6]{%
9989 \ifnum##1=1\relax
9990 \glsesubentryitem{##2}%
9991 \fi
9992 \ifnum\@gls@prevlevel=##1\relax
9993 \else
9994 \@ifundefined{@glswidestname\romannumeral##1}{%
9995 \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9996 \settowidth{\gls@tmplen}{\textbf{%
9997 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9998 \ifnum\@gls@prevlevel<##1\relax
9999 \setlength\glstreeindent\gls@tmplen
10000 \addtolength\glstreeindent\parindent
10001 \parindent\glstreeindent
10002 \else
10003 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10004 \settowidth{\glstreeindent}{\textbf{%
10005 \@glswidestname\space}}{%
10006 \settowidth{\glstreeindent}{\textbf{%
10007 \csname @glswidestname\romannumeral\@gls@prevlevel
10008 \endcsname\space}}}%
10009 \addtolength\parindent{-\glstreeindent}}%
```

```

10010      \setlength\glstreeindent\parindent
10011      \fi
10012      \fi
10013      \hangindent\glstreeindent
10014      \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
10015        \textbf{\glstarget{##2}{##3}}}}%
10016      \ifx##5\relax\relax
10017      \else
10018        (##5)\space
10019      \fi
10020      ##4\glspostdescription\space ##6\par
10021      \def\@gls@prevlevel{##1}%
10022    }%
10023 }%

```

Backward compatible alltreegroup style.

```

10024 \compatglossarystyle{alltreegroup}{%
10025 \csuse{@glscompstyle@almtree}%
10026 }%

```

Backward compatible almtreehypergroup style.

```

10027 \compatglossarystyle{almtreehypergroup}{%
10028 \csuse{@glscompstyle@almtree}%
10029 }%

```

Backward compatible mcolindex style.

```

10030 \compatglossarystyle{mcolindex}{%
10031 \csuse{@glscompstyle@index}%
10032 }%

```

Backward compatible mcolindexgroup style.

```

10033 \compatglossarystyle{mcolindexgroup}{%
10034 \csuse{@glscompstyle@index}%
10035 }%

```

Backward compatible mcolindexhypergroup style.

```

10036 \compatglossarystyle{mcolindexhypergroup}{%
10037 \csuse{@glscompstyle@index}%
10038 }%

```

Backward compatible mcoltree style.

```

10039 \compatglossarystyle{mcoltree}{%
10040 \csuse{@glscompstyle@tree}%
10041 }%

```

Backward compatible mcoltreegroup style.

```

10042 \compatglossarystyle{mcolindextreegroup}{%
10043 \csuse{@glscompstyle@tree}%
10044 }%

```

Backward compatible mcoltreehypergroup style.

```

10045 \compatglossarystyle{mcolindextreehypergroup}{%

```



```

10046 \csuse{@glscompstyle@tree}%
10047 }%

    Backward compatible mcoltreenoname style.
10048 \compatglossarystyle{mcoltreenoname}{%
10049 \csuse{@glscompstyle@tree}%
10050 }%

    Backward compatible mcoltreenonamegroup style.
10051 \compatglossarystyle{mcoltreenonamegroup}{%
10052 \csuse{@glscompstyle@tree}%
10053 }%

    Backward compatible mcoltreenonamehypergroup style.
10054 \compatglossarystyle{mcoltreenonamehypergroup}{%
10055 \csuse{@glscompstyle@tree}%
10056 }%

    Backward compatible mcolalmtree style.
10057 \compatglossarystyle{mcolalmtree}{%
10058 \csuse{@glscompstyle@almtree}%
10059 }%

    Backward compatible mcolalmtreegroup style.
10060 \compatglossarystyle{mcolalmtreegroup}{%
10061 \csuse{@glscompstyle@almtree}%
10062 }%

    Backward compatible mcolalmtreehypergroup style.
10063 \compatglossarystyle{mcolalmtreehypergroup}{%
10064 \csuse{@glscompstyle@almtree}%
10065 }%

    Backward compatible superragged style.
10066 \compatglossarystyle{superragged}{%
10067 \renewcommand*{\glossaryentryfield}[5]{%
10068 \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10069 \tabularnewline}%
10070 \renewcommand*{\glossarysubentryfield}[6]{%
10071 &
10072 \glssubentryitem{##2}%
10073 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10074 \tabularnewline}%
10075 }%

    Backward compatible superraggedborder style.
10076 \compatglossarystyle{superraggedborder}{%
10077 \csuse{@glscompstyle@superragged}%
10078 }%

    Backward compatible superraggedheader style.
10079 \compatglossarystyle{superraggedheader}{%
10080 \csuse{@glscompstyle@superragged}%
10081 }%

```

Backward compatible superraggedheaderborder style.

```
10082 \compatglossarystyle{superraggedheaderborder}{%
10083 \csuse{@glscompstyle@superragged}%
10084 }%
```

Backward compatible superragged3col style.

```
10085 \compatglossarystyle{superragged3col}{%
10086 \renewcommand*{\glossaryentryfield}[5]{%
10087 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10088 \renewcommand*{\glossarysubentryfield}[6]{%
10089 &
10090 \glssubentryitem{##2}%
10091 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10092 }%
```

Backward compatible superragged3colborder style.

```
10093 \compatglossarystyle{superragged3colborder}{%
10094 \csuse{@glscompstyle@superragged3col}%
10095 }%
```

Backward compatible superragged3colheader style.

```
10096 \compatglossarystyle{superragged3colheader}{%
10097 \csuse{@glscompstyle@superragged3col}%
10098 }%
```

Backward compatible superragged3colheaderborder style.

```
10099 \compatglossarystyle{superragged3colheaderborder}{%
10100 \csuse{@glscompstyle@superragged3col}%
10101 }%
```

Backward compatible altsuperragged4col style.

```
10102 \compatglossarystyle{altsuperragged4col}{%
10103 \renewcommand*{\glossaryentryfield}[5]{%
10104 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10105 \renewcommand*{\glossarysubentryfield}[6]{%
10106 &
10107 \glssubentryitem{##2}%
10108 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10109 }%
```

Backward compatible altsuperragged4colheader style.

```
10110 \compatglossarystyle{altsuperragged4colheader}{%
10111 \csuse{@glscompstyle@altsuperragged4col}%
10112 }%
```

Backward compatible altsuperragged4colborder style.

```
10113 \compatglossarystyle{altsuperragged4colborder}{%
10114 \csuse{@glscompstyle@altsuperragged4col}%
10115 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10116 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10117 \csuse{@glscompstyle@altsuperragged4col}%
10118 }%
```

Backward compatible super style.

```
10119 \compatglossarystyle{super}{%
10120 \renewcommand*{\glossaryentryfield}[5]{%
10121 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10122 \renewcommand*{\glossarysubentryfield}[6]{%
10123 &
10124 \glssubentryitem{##2}%
10125 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10126 }%
```

Backward compatible superborder style.

```
10127 \compatglossarystyle{superborder}{%
10128 \csuse{@glscompstyle@super}%
10129 }%
```

Backward compatible superheader style.

```
10130 \compatglossarystyle{superheader}{%
10131 \csuse{@glscompstyle@super}%
10132 }%
```

Backward compatible superheaderborder style.

```
10133 \compatglossarystyle{superheaderborder}{%
10134 \csuse{@glscompstyle@super}%
10135 }%
```

Backward compatible super3col style.

```
10136 \compatglossarystyle{super3col}{%
10137 \renewcommand*{\glossaryentryfield}[5]{%
10138 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10139 \renewcommand*{\glossarysubentryfield}[6]{%
10140 &
10141 \glssubentryitem{##2}%
10142 \glstarget{##2}{\strut}##4 & ##6\\}%
10143 }%
```

Backward compatible super3colborder style.

```
10144 \compatglossarystyle{super3colborder}{%
10145 \csuse{@glscompstyle@super3col}%
10146 }%
```

Backward compatible super3colheader style.

```
10147 \compatglossarystyle{super3colheader}{%
10148 \csuse{@glscompstyle@super3col}%
10149 }%
```

Backward compatible super3colheaderborder style.

```
10150 \compatglossarystyle{super3colheaderborder}{%
10151 \csuse{@glscompstyle@super3col}%
10152 }%
```

Backward compatible super4col style.

```
10153 \compatglossarystyle{super4col}{%
10154   \renewcommand*\glossaryentryfield}[5]{%
10155     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10156   \renewcommand*\glossarysubentryfield}[6]{%
10157     &
10158     \glssubentryitem{##2}%
10159     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10160 }%
```

Backward compatible super4colheader style.

```
10161 \compatglossarystyle{super4colheader}{%
10162   \csuse{@glscompstyle@super4col}%
10163 }%
```

Backward compatible super4colborder style.

```
10164 \compatglossarystyle{super4colborder}{%
10165   \csuse{@glscompstyle@super4col}%
10166 }%
```

Backward compatible super4colheaderborder style.

```
10167 \compatglossarystyle{super4colheaderborder}{%
10168   \csuse{@glscompstyle@super4col}%
10169 }%
```

Backward compatible altsuper4col style.

```
10170 \compatglossarystyle{altsuper4col}{%
10171   \csuse{@glscompstyle@super4col}%
10172 }%
```

Backward compatible altsuper4colheader style.

```
10173 \compatglossarystyle{altsuper4colheader}{%
10174   \csuse{@glscompstyle@super4col}%
10175 }%
```

Backward compatible altsuper4colborder style.

```
10176 \compatglossarystyle{altsuper4colborder}{%
10177   \csuse{@glscompstyle@super4col}%
10178 }%
```

Backward compatible altsuper4colheaderborder style.

```
10179 \compatglossarystyle{altsuper4colheaderborder}{%
10180   \csuse{@glscompstyle@super4col}%
10181 }%
```

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.

```
10182 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10183 \ProvidesPackage{glossaries-accsupp}[2016/06/09 v4.25 (NLCT)
```

```
10184 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10185 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10186 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10187 \@ifpackageloaded{glossaries-extra}
```

```
10188 {%
```

```
10189 \PackageWarning{glossaries-accsupp}{The ‘glossaries-accsupp’
```

```
10190 package has been loaded after the ‘glossaries-extra’
```

```
10191 package. This can cause a failure to integrate both
```

```
10192 packages. Either use the ‘accsupp’ option when you
```

```
10193 load ‘glossaries-extra’ or load ‘glossaries-accsupp’
```

```
10194 before loading ‘glossaries-extra’}}%
```

```
10195 }
```

```
10196 {}
```

tibleglossentry Override style compatibility macros:

```
10197 \def\compatibleglossentry#1#2{%
```

```
10198 \toks@{#2}%
```

```
10199 \protected@edef\do@glossentry{%
```

```
10200 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10201 {\noexpand\glsnamefont
```

```
10202 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```
10203 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
```

```
10204 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
```

```
10205 {\the\toks@}%
```

```
10206 }%
```

```
10207 \do@glossentry
```

```
10208 }
```

lesubglossentry

```
10209 \def\compatiblesubglossentry#1#2#3{%
10210   \toks@{#3}%
10211   \protected@edef\@do@subglossentry{%
10212     \noexpand\accsuppglossarysubentryfield{\number#1}%
10213     {#2}%
10214     {\noexpand\glsnamefont
10215       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
10216     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
10217     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
10218     {\the\toks@}%
10219   }%
10220   \@do@subglossentry
10221 }
```

Required packages:

```
10222 \RequirePackage{glossaries}
10223 \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:

```
10224 \define@key{glossentry}{access}{%
10225   \def\@glo@access{#1}%
10226 }
```

textaccess The replacement text corresponding to the text key:

```
10227 \define@key{glossentry}{textaccess}{%
10228   \def\@glo@textaccess{#1}%
10229 }
```

firstaccess The replacement text corresponding to the first key:

```
10230 \define@key{glossentry}{firstaccess}{%
10231   \def\@glo@firstaccess{#1}%
10232 }
```

pluralaccess The replacement text corresponding to the plural key:

```
10233 \define@key{glossentry}{pluralaccess}{%
10234   \def\@glo@pluralaccess{#1}%
10235 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
10236 \define@key{glossentry}{firstpluralaccess}{%
10237   \def\@glo@firstpluralaccess{#1}%
10238 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
10239 \define@key{glossentry}{symbolaccess}{%
10240   \def\@glo@symbolaccess{#1}%
10241 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
10242 \define@key{glossentry}{symbolpluralaccess}{%
10243   \def\@glo@symbolpluralaccess{#1}%
10244 }
```

descriptionaccess The replacement text corresponding to the description key:

```
10245 \define@key{glossentry}{descriptionaccess}{%
10246   \def\@glo@descaccess{#1}%
10247 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
10248 \define@key{glossentry}{descriptionpluralaccess}{%
10249   \def\@glo@descpluralaccess{#1}%
10250 }
```

shortaccess The replacement text corresponding to the short key:

```
10251 \define@key{glossentry}{shortaccess}{%
10252   \def\@glo@shortaccess{#1}%
10253 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
10254 \define@key{glossentry}{shortpluralaccess}{%
10255   \def\@glo@shortpluralaccess{#1}%
10256 }
```

longaccess The replacement text corresponding to the long key:

```
10257 \define@key{glossentry}{longaccess}{%
10258   \def\@glo@longaccess{#1}%
10259 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
10260 \define@key{glossentry}{longpluralaccess}{%
10261   \def\@glo@longpluralaccess{#1}%
10262 }
```

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={\glsaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
10263 \appto\@gls@keymap{,%  
10264   {access}{access},%  
10265   {textaccess}{textaccess},%  
10266   {firstaccess}{firstaccess},%  
10267   {pluralaccess}{pluralaccess},%  
10268   {firstpluralaccess}{firstpluralaccess},%  
10269   {symbolaccess}{symbolaccess},%  
10270   {symbolpluralaccess}{symbolpluralaccess},%  
10271   {descaccess}{descaccess},%  
10272   {descpluralaccess}{descpluralaccess},%  
10273   {shortaccess}{shortaccess},%  
10274   {shortpluralaccess}{shortpluralaccess},%  
10275   {longaccess}{longaccess},%  
10276   {longpluralaccess}{longpluralaccess}%  
10277 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
10278 \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):

```
10279 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook  
10280 \renewcommand*{\@newglossaryentryprehook}{%  
10281   \@gls@oldnewglossaryentryprehook  
10282   \def\@glo@access{\@glo@symbol}}%
```

Initialise the other keys:

```
10283 \def\@glo@textaccess{\@glo@access}%  
10284 \def\@glo@firstaccess{\@glo@access}%  
10285 \def\@glo@pluralaccess{\@glo@textaccess}%  
10286 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%  
10287 \def\@glo@symbolaccess{\relax}%  
10288 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%  
10289 \def\@glo@descaccess{\relax}%  
10290 \def\@glo@descpluralaccess{\@glo@descaccess}%  
10291 \def\@glo@shortaccess{\relax}%  
10292 \def\@glo@shortpluralaccess{\@glo@shortaccess}%  
10293 \def\@glo@longaccess{\relax}%  
10294 \def\@glo@longpluralaccess{\@glo@longaccess}%  
10295 }
```

Add to the end hook:

```
10296 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook  
10297 \renewcommand*{\@newglossaryentryposthook}{%  
10298   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```
10299 \expandafter  
10300 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
```



```

10301     \@glo@access}%
10302 \expandafter
10303     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10304     \@glo@textaccess}%
10305 \expandafter
10306     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10307     \@glo@firstaccess}%
10308 \expandafter
10309     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10310     \@glo@pluralaccess}%
10311 \expandafter
10312     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10313     \@glo@firstpluralaccess}%
10314 \expandafter
10315     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10316     \@glo@symbolaccess}%
10317 \expandafter
10318     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10319     \@glo@symbolpluralaccess}%
10320 \expandafter
10321     \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10322     \@glo@descaccess}%
10323 \expandafter
10324     \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10325     \@glo@descpluralaccess}%
10326 \expandafter
10327     \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10328     \@glo@shortaccess}%
10329 \expandafter
10330     \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10331     \@glo@shortpluralaccess}%
10332 \expandafter
10333     \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10334     \@glo@longaccess}%
10335 \expandafter
10336     \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10337     \@glo@longpluralaccess}%
10338 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10339 \newcommand*{\glsentryaccess}[1]{%
10340   \@gls@entry@field{#1}{access}%
10341 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

10342 \newcommand*{\glsentrytextaccess}[1]{%

```

```
10343 \@gls@entry@field{#1}{textaccess}%
10344 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10345 \newcommand*{\glsentryfirstaccess}[1]{%
10346 \@gls@entry@field{#1}{firstaccess}%
10347 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10348 \newcommand*{\glsentrypluralaccess}[1]{%
10349 \@gls@entry@field{#1}{pluralaccess}%
10350 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10351 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10352 \csname glo#1@firstpluralaccess\endcsname
10353 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10354 \newcommand*{\glsentrysymbolaccess}[1]{%
10355 \@gls@entry@field{#1}{symbolaccess}%
10356 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10357 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10358 \@gls@entry@field{#1}{symbolpluralaccess}%
10359 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10360 \newcommand*{\glsentrydescaccess}[1]{%
10361 \@gls@entry@field{#1}{descaccess}%
10362 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10363 \newcommand*{\glsentrydescpluralaccess}[1]{%
10364 \@gls@entry@field{#1}{descaccess}%
10365 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10366 \newcommand*{\glsentryshortaccess}[1]{%
10367 \@gls@entry@field{#1}{shortaccess}%
10368 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10369 \newcommand*{\glsentryshortpluralaccess}[1]{%
10370 \@gls@entry@field{#1}{shortpluralaccess}%
10371 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10372 \newcommand*{\glentrylongaccess}[1]{%
10373   \@gls@entry@field{#1}{longaccess}%
10374 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10375 \newcommand*{\glentrylongpluralaccess}[1]{%
10376   \@gls@entry@field{#1}{longpluralaccess}%
10377 }
```

`\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.)

```
10378 \newcommand*{\glsaccsupp}[2]{%
10379   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10380 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
10381 \newcommand*{\xglsaccsupp}[2]{%
10382   \protected@edef\@gls@replacementtext{#1}%
10383   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10384 }
```

@access@display

```
10385 \newcommand*{\@gls@access@display}[2]{%
10386   \protected@edef\@glo@access{#2}%
10387   \ifx\@glo@access\@gls@noaccess
10388     #1%
10389   \else
10390     \xglsaccsupp{\@glo@access}{#1}%
10391   \fi
10392 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10393 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10394   \@gls@access@display{#1}{\glentryaccess{#2}}%
10395 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10396 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10397   \@gls@access@display{#1}{\glentrytextaccess{#2}}%
10398 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10399 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10400   \@gls@access@display{#1}{\glentrypluralaccess{#2}}%
10401 }
```

staccessdisplay As above but for the firstaccess replacement text.

```

10402 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
10403 \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
10404 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10405 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
10406 \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
10407 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10408 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
10409 \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
10410 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10411 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
10412 \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
10413 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10414 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10415 \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
10416 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10417 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10418 \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
10419 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10420 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10421 \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
10422 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10423 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10424 \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
10425 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10426 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10427 \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
10428 }

```

alaccessdisplay As above but for the longpluralaccess replacement text.

```

10429 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10430 \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
10431 }

```

`glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
10432 \DeclareRobustCommand*{\glsaccessdisplay}[3]{%
10433   \@ifundefined{gls#1accessdisplay}%
10434   {%
10435     \PackageError{glossaries-accsupp}{No accessibility support
10436       for key ‘#1’}{%
10437     }%
10438   }%
10439   \csname gls#1accessdisplay\endcsname{#2}{#3}%
10440 }%
10441 }
```

`default@entryfmt` Redefine the default entry format to use accessibility information

```
10442 \renewcommand*{\@@gls@default@entryfmt}[2]{%
10443   \ifdefempty\glscustomtext
10444   {%
10445     \glsifplural
10446     {%
10447       Plural form
10447       \glscapscase
10448       {%
10449         Don't adjust case
10449         \ifglsused\glslabel
10450         {%
10451           Subsequent use
10451           #2{\glspluralaccessdisplay
10452             {\glsentryplural{\glslabel}}{\glslabel}}%
10453             {\glsdescriptionpluralaccessdisplay
10454               {\glsentrydescplural{\glslabel}}{\glslabel}}%
10455               {\glsymbolpluralaccessdisplay
10456                 {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10457             {\glsinsert}}%
10458           }%
10459           {%
10460             First use
10460             #1{\glsfirstpluralaccessdisplay
10461               {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10462               {\glsdescriptionpluralaccessdisplay
10463                 {\glsentrydescplural{\glslabel}}{\glslabel}}%
10464                 {\glsymbolpluralaccessdisplay
10465                   {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10466                 {\glsinsert}}%
10467             }%
10468           }%
10469           {%
```

Make first letter upper case

```
10470     \ifglsused\glslabel
10471     {%
```

Subsequent use.

```
10472     #2{\glspluralaccessdisplay
10473         {\Glsentryplural{\glslabel}}{\glslabel}}%
10474         {\glsdescriptionpluralaccessdisplay
10475         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10476         {\glsymbolpluralaccessdisplay
10477         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10478         {\glsinsert}}%
10479     }%
10480     {%
```

First use

```
10481     #1{\glsfirstpluralaccessdisplay
10482         {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10483         {\glsdescriptionpluralaccessdisplay
10484         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10485         {\glsymbolpluralaccessdisplay
10486         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10487         {\glsinsert}}%
10488     }%
10489     }%
10490     {%
```

Make all upper case

```
10491     \ifglsused\glslabel
10492     {%
```

Subsequent use

```
10493     \MakeUppercase{%
10494     #2{\glspluralaccessdisplay
10495         {\glsentryplural{\glslabel}}{\glslabel}}%
10496         {\glsdescriptionpluralaccessdisplay
10497         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10498         {\glsymbolpluralaccessdisplay
10499         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10500         {\glsinsert}}%
10501     }%
10502     {%
```

First use

```
10503     \MakeUppercase{%
10504     #1{\glsfirstpluralaccessdisplay
10505         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10506         {\glsdescriptionpluralaccessdisplay
10507         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10508         {\glsymbolpluralaccessdisplay
10509         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10510         {\glsinsert}}%
10511     }%
10512 }%
10513 }%
10514 {%
```

Singular form

```

10515     \glscapscase
10516     {%
```

Don't adjust case

```

10517     \ifglsused\glslabel
10518     {%
```

Subsequent use

```

10519     #2{\glstextaccessdisplay
10520         {\glsentrytext{\glslabel}}{\glslabel}}%
10521     {\glsdescriptionaccessdisplay
10522         {\glsentrydesc{\glslabel}}{\glslabel}}%
10523     {\glssymbolaccessdisplay
10524         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10525     {\glsinsert}%
10526 }%
10527 {%
```

First use

```

10528     #1{\glsfirstaccessdisplay
10529         {\glsentryfirst{\glslabel}}{\glslabel}}%
10530     {\glsdescriptionaccessdisplay
10531         {\glsentrydesc{\glslabel}}{\glslabel}}%
10532     {\glssymbolaccessdisplay
10533         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10534     {\glsinsert}%
10535 }%
10536 }%
10537 {%
```

Make first letter upper case

```

10538     \ifglsused\glslabel
10539     {%
```

Subsequent use

```

10540     #2{\glstextaccessdisplay
10541         {\Glsentrytext{\glslabel}}{\glslabel}}%
10542     {\glsdescriptionaccessdisplay
10543         {\Glsentrydesc{\glslabel}}{\glslabel}}%
10544     {\glssymbolaccessdisplay
10545         {\Glsentrysymbol{\glslabel}}{\glslabel}}%
10546     {\glsinsert}%
10547 }%
10548 {%
```

First use

```
10549      #1{\glsfirstaccessdisplay
10550          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10551          {\glsdescriptionaccessdisplay
10552           {\glsentrydesc{\glslabel}}{\glslabel}}%
10553          {\glsymbolaccessdisplay
10554           {\glsentrysymbol{\glslabel}}{\glslabel}}%
10555          {\glsinsert}}%
10556      }%
10557  }%
10558  {%
```

Make all upper case

```
10559      \ifglsused\glslabel
10560      {%
```

Subsequent use

```
10561      \MakeUppercase{%
10562          #2{\glsfirstaccessdisplay
10563             {\glsentrytext{\glslabel}}{\glslabel}}%
10564             {\glsdescriptionaccessdisplay
10565              {\glsentrydesc{\glslabel}}{\glslabel}}%
10566             {\glsymbolaccessdisplay
10567              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10568             {\glsinsert}}%
10569      }%
10570  }%
```

First use

```
10571      \MakeUppercase{%
10572          #1{\glsfirstaccessdisplay
10573             {\glsentryfirst{\glslabel}}{\glslabel}}%
10574             {\glsdescriptionaccessdisplay
10575              {\glsentrydesc{\glslabel}}{\glslabel}}%
10576             {\glsymbolaccessdisplay
10577              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10578             {\glsinsert}}%
10579      }%
10580  }%
10581 }%
10582 }%
10583 {%
```

Custom text provided in \glsdisp

```
10584      \ifglsused{\glslabel}%
10585      {%
```

Subsequent use

```
10586      #2{\glscustomtext}%
10587      {\glsdescriptionaccessdisplay
10588       {\glsentrydesc{\glslabel}}{\glslabel}}%
```



```

10589     {\glssymbolaccessdisplay
10590       {\glseentrysymbol{\glslabel}}{\glslabel}}%
10591     {\glsinsert}%
10592   }%
10593   {%

```

First use

```

10594     #1{\glscustomtext}%
10595     {\glsdescriptionaccessdisplay
10596       {\glseentrydesc{\glslabel}}{\glslabel}}%
10597     {\glssymbolaccessdisplay
10598       {\glseentrysymbol{\glslabel}}{\glslabel}}%
10599     {\glsinsert}%
10600   }%
10601 }%
10602 }

```

`\glsentryfmt` Redefine to use accessibility information.

```

10603 \renewcommand*{\glsentryfmt}{%
10604   \ifdefempty\glscustomtext
10605   {%
10606     \glsifplural
10607     {%

```

Plural form

```

10608     \glscapscase
10609     {%

```

Don't adjust case

```

10610     \ifglsused\glslabel
10611     {%

```

Subsequent use

```

10612     \glspluralaccessdisplay
10613       {\glsentryplural{\glslabel}}{\glslabel}%
10614     \glsinsert
10615   }%
10616   {%

```

First use

```

10617     \glsfirstpluralaccessdisplay
10618       {\glsentryfirstplural{\glslabel}}{\glslabel}%
10619     \glsinsert
10620   }%
10621 }%
10622 {%

```

Make first letter upper case

```

10623     \ifglsused\glslabel
10624     {%

```

Subsequent use.

```
10625      \glspluralaccessdisplay
10626      {\Glsentryplural{\glslabel}}{\glslabel}%
10627      \glsinsert
10628      }%
10629      {%
```

First use

```
10630      \glsfirstpluralaccessdisplay
10631      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10632      \glsinsert
10633      }%
10634      }%
10635      {%
```

Make all upper case

```
10636      \ifglsused\glslabel
10637      {%
```

Subsequent use

```
10638      \glspluralaccessdisplay
10639      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
10640      {\glslabel}%
10641      \mfirstucMakeUppercase{\glsinsert}%
10642      }%
10643      {%
```

First use

```
10644      \glsfirstpluralaccessdisplay
10645      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
10646      {\glslabel}%
10647      \mfirstucMakeUppercase{\glsinsert}%
10648      }%
10649      }%
10650      }%
10651      {%
```

Singular form

```
10652      \glscapscale
10653      {%
```

Don't adjust case

```
10654      \ifglsused\glslabel
10655      {%
```

Subsequent use

```
10656      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
10657      \glsinsert
10658      }%
10659      {%
```

First use

```
10660      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10661      \glsinsert
10662      }%
10663      }%
10664      {%
```

Make first letter upper case

```
10665      \ifglsused\glslabel
10666      {%
```

Subsequent use

```
10667      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10668      \glsinsert
10669      }%
10670      {%
```

First use

```
10671      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10672      \glsinsert
10673      }%
10674      }%
10675      {%
```

Make all upper case

```
10676      \ifglsused\glslabel
10677      {%
```

Subsequent use

```
10678      \glstextaccessdisplay
10679      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10680      \mfirstucMakeUppercase{\glsinsert}%
10681      }%
10682      {%
```

First use

```
10683      \glsfirstaccessdisplay
10684      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10685      \mfirstucMakeUppercase{\glsinsert}%
10686      }%
10687      }%
10688      }%
10689      }%
10690      {%
```

Custom text provided in `\glsdisp`. (The insert should be empty at this point.) The accessibility information, if required, will have to be explicitly included in the custom text.

```
10691      \glscustomtext\glsinsert
10692      }%
10693      }
```

`\glsgenacfmt` Redefine to include accessibility information.

```
10694 \renewcommand*{\glsgenacfmt}{%
10695   \ifdefempty\glscustomtext
10696   {%
10697     \ifglused\glslabel
10698     {%
```

Subsequent use:

```
10699     \glsifplural
10700     {%
```

Subsequent plural form:

```
10701     \glscapscase
10702     {%
```

Subsequent plural form, don't adjust case:

```
10703     \acronymfont
10704     {\glsshortpluralaccessdisplay
10705       {\glentryshortpl{\glslabel}}{\glslabel}}%
10706     \glsinsert
10707     }%
10708     {%
```

Subsequent plural form, make first letter upper case:

```
10709     \acronymfont
10710     {\glsshortpluralaccessdisplay
10711       {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10712     \glsinsert
10713     }%
10714     {%
```

Subsequent plural form, all caps:

```
10715     \mfirstucMakeUppercase
10716     {\acronymfont
10717       {\glsshortpluralaccessdisplay
10718         {\glentryshortpl{\glslabel}}{\glslabel}}%
10719       \glsinsert}%
10720     }%
10721     }%
10722     {%
```

Subsequent singular form

```
10723     \glscapscase
10724     {%
```

Subsequent singular form, don't adjust case:

```
10725     \acronymfont
10726     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10727     \glsinsert
10728     }%
10729     {%
```

Subsequent singular form, make first letter upper case:

```
10730      \acronymfont
10731      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10732      \glsinsert
10733      }%
10734      {%
```

Subsequent singular form, all caps:

```
10735      \mfirstucMakeUppercase
10736      {\acronymfont{%
10737      \glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
10738      \glsinsert}%
10739      }%
10740      }%
10741      }%
10742      {%
```

First use:

```
10743      \glsifplural
10744      {%
```

First use plural form:

```
10745      \glscapscase
10746      {%
```

First use plural form, don't adjust case:

```
10747      \genplacrfullformat{\glslabel}{\glsinsert}%
10748      }%
10749      {%
```

First use plural form, make first letter upper case:

```
10750      \Genplacrfullformat{\glslabel}{\glsinsert}%
10751      }%
10752      {%
```

First use plural form, all caps:

```
10753      \mfirstucMakeUppercase
10754      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10755      }%
10756      }%
10757      {%
```

First use singular form

```
10758      \glscapscase
10759      {%
```

First use singular form, don't adjust case:

```
10760      \genacrfullformat{\glslabel}{\glsinsert}%
10761      }%
10762      {%
```

First use singular form, make first letter upper case:

```
10763      \Genacrfullformat{\glslabel}{\glsinsert}%
10764      }%
10765      {%
```

First use singular form, all caps:

```
10766      \mfirstucMakeUppercase
10767      {\genacrfullformat{\glslabel}{\glsinsert}}%
10768      }%
10769      }%
10770      }%
10771      }%
10772      {%
```

User supplied text. (The insert should be empty at this point.) The accessibility information, if required, will have to be explicitly included in the custom text.

```
10773      \glscustomtext
10774      }%
10775 }
```

enacrfullformat Redefine to include accessibility information.

```
10776 \renewcommand*{\genacrfullformat}[2]{%
10777   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10778   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10779 }
```

enacrfullformat Redefine to include accessibility information.

```
10780 \renewcommand*{\Genacrfullformat}[2]{%
10781   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10782   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10783 }
```

placrfullformat Redefine to include accessibility information.

```
10784 \renewcommand*{\genplacrfullformat}[2]{%
10785   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10786   (\glsshortpluralaccessdisplay
10787     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10788 }
```

placrfullformat Redefine to include accessibility information.

```
10789 \renewcommand*{\Genplacrfullformat}[2]{%
10790   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10791   (\glsshortpluralaccessdisplay
10792     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10793 }
```

\@acrshort

```
10794 \def\@acrshort#1#2[#3]{%
10795   \glsdoifexists{#2}%
```

```

10796 {%
10797   \let\do@gls@link@checkfirsthyper\relax

10798   \let\glsifplural\@secondoftwo
10799   \let\glsapscase\@firstofthree
10800   \let\glsinsert\@empty
10801   \def\glscustomtext{%
10802     \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
10803   }%

   Call \@gls@link
10804   \@gls@link[#1]{#2}{\cename gls@\glstype @entryfmt\endcename}%
10805 }%

10806 \glspostlinkhook
10807 }

```

\@Acrshort

```

10808 \def\@Acrshort#1#2[#3]{%
10809   \glsdoifexists{#2}%
10810   {%
10811     \let\do@gls@link@checkfirsthyper\relax

10812     \let\glsifplural\@secondoftwo
10813     \let\glsapscase\@secondofthree
10814     \let\glsinsert\@empty
10815     \def\glscustomtext{%
10816       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10817     }%

     Call \@gls@link
10818     \@gls@link[#1]{#2}{\cename gls@\glstype @entryfmt\endcename}%
10819   }%

10820   \glspostlinkhook
10821 }

```

\@ACRshort

```

10822 \def\@ACRshort#1#2[#3]{%
10823   \glsdoifexists{#2}%
10824   {%
10825     \let\do@gls@link@checkfirsthyper\relax

10826     \let\glsifplural\@secondoftwo
10827     \let\glsapscase\@thirdofthree
10828     \let\glsinsert\@empty
10829     \def\glscustomtext{%
10830       \acronymfont{\glsshortaccessdisplay
10831         {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
10832     }%

```

```

Call \@gls@link
10833   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10834   }%

10835   \glspostlinkhook
10836 }

```

\@acrlong

```

10837 \def\@acrlong#1#2[#3]{%
10838   \glsdoifexists{#2}%
10839   {%
10840     \let\do@gls@link@checkfirsthyper\relax

10841     \let\glsifplural\@secondoftwo
10842     \let\glscapscase\@firstofthree
10843     \let\glsinsert\@empty
10844     \def\glscustomtext{%
10845       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10846     }%

```

Call \@gls@link

```

10847   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10848   }%

10849   \glspostlinkhook
10850 }

```

\@Acrlong

```

10851 \def\@Acrlong#1#2[#3]{%
10852   \glsdoifexists{#2}%
10853   {%
10854     \let\do@gls@link@checkfirsthyper\relax

10855     \let\glsifplural\@secondoftwo
10856     \let\glsapspace\@firstofthree
10857     \let\glsinsert\@empty
10858     \def\glscustomtext{%
10859       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10860     }%

```

Call \@gls@link

```

10861   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10862   }%

10863   \glspostlinkhook
10864 }

```

\@ACRlong

```

10865 \def\@ACRlong#1#2[#3]{%
10866   \glsdoifexists{#2}%
10867   {%
10868     \let\do@gls@link@checkfirsthyper\relax

```



```

10869 \let\glsifplural\@secondoftwo
10870 \let\glsifscapscase\@firstofthree
10871 \let\glsinsert\@empty
10872 \def\glscustomtext{%
10873   \acronymfont{\glslongaccessdisplay{%
10874     \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10875   }%

Call \@gls@link
10876 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10877 }%

10878 \glspostlinkhook
10879 }

```

5.3 Displaying the Glossary

We need to redefine the way the glossary entries are formatted to include the accessibility support. The predefined glossary styles use `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol`, but we need to provide compatibility with earlier versions in case users have defined their own styles using `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

Now redefine `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol` etc so they use the accessibility stuff.

```

10880 \renewcommand*\glossentryname}[1]{%
10881   \glsdoifexists{#1}%
10882   {%
10883     \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10884   }%
10885 }

10886 \renewcommand*\glossentryname}[1]{%
10887   \glsdoifexists{#1}%
10888   {%
10889     \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10890   }%
10891 }

10892 \renewcommand*\glossentrydesc}[1]{%
10893   \glsdoifexists{#1}%
10894   {%
10895     \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10896   }%
10897 }

10898 \renewcommand*\Glossentrydesc}[1]{%
10899   \glsdoifexists{#1}%
10900   {%
10901     \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10902   }%
10903 }

```

```

10904 \renewcommand*{\glossentrysymbol}[1]{%
10905   \glsdoifexists{#1}%
10906   {%
10907     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10908   }%
10909 }

10910 \renewcommand*{\Glossentrysymbol}[1]{%
10911   \glsdoifexists{#1}%
10912   {%
10913     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10914   }%
10915 }

```

ssaryentryfield

```

10916 \newcommand*{\accsuppglossaryentryfield}[5]{%
10917   \glossaryentryfield{#1}%
10918   {\glsnameaccessdisplay{#2}{#1}}%
10919   {\glsdescriptionaccessdisplay{#3}{#1}}%
10920   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10921 }

```

rysubentryfield

```

10922 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10923   \glossarysubentryfield{#1}{#2}%
10924   {\glsnameaccessdisplay{#3}{#2}}%
10925   {\glsdescriptionaccessdisplay{#4}{#2}}%
10926   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10927 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10928 \renewacronymstyle{long-short}%
10929 {}%

```

Check for long form in case this is a mixed glossary.

```

10930 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
10931 }%
10932 {}%
10933 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10934 \renewcommand*{\genacrfullformat}[2]{%
10935   \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10936   (\glsshortaccessdisplay
10937     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10938 }%
10939 \renewcommand*{\Genacrfullformat}[2]{%

```

```

10940 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10941 (\glsshortaccessdisplay
10942   {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10943 }%
10944 \renewcommand*{\genplacrformat}[2]{%
10945   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10946   (\glsshortpluralaccessdisplay
10947     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10948   }%
10949 \renewcommand*{\Genplacrformat}[2]{%
10950   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10951   (\glsshortpluralaccessdisplay
10952     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10953   }%
10954 \renewcommand*{\acronymentry}[1]{%
10955   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10956 \renewcommand*{\acronymsort}[2]{##1}%
10957 \renewcommand*{\acronymfont}[1]{##1}%
10958 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10959 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10960 }

```

short-long (*short*) (*long*) acronym style.

```

10961 \renewacronymstyle{short-long}%
10962 {%

```

Check for long form in case this is a mixed glossary.

```

10963   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
10964 }%
10965 {%
10966   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10967   \renewcommand*{\genacrformat}[2]{%
10968     \glsshortaccessdisplay
10969     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
10970     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10971   }%
10972   \renewcommand*{\Genacrformat}[2]{%
10973     \glsshortaccessdisplay
10974     {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
10975     (\glslongaccessdisplay{\Glsentrylong{##1}}{##1})%
10976   }%
10977   \renewcommand*{\genplacrformat}[2]{%
10978     \glsshortpluralaccessdisplay
10979     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
10980     (\glslongpluralaccessdisplay
10981       {\glsentrylongpl{##1}}{##1})%
10982   }%
10983   \renewcommand*{\Genplacrformat}[2]{%
10984     \glsshortpluralaccessdisplay
10985     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space

```

```

10986 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10987 }%
10988 \renewcommand*{\acronymentry}[1]{%
10989 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10990 \renewcommand*{\acronymsort}[2]{##1}%
10991 \renewcommand*{\acronymfont}[1]{##1}%
10992 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10993 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10994 }

```

long-short-desc *long* (*short*) acronym style that has an accompanying description (which the user needs to supply).

```

10995 \renewacronymstyle{long-short-desc}%
10996 {%
10997 \GlsUseAcrEntryDispStyle{long-short}%
10998 }%
10999 {%
11000 \GlsUseAcrStyleDefs{long-short}%
11001 \renewcommand*{\GenericAcronymFields}{}%
11002 \renewcommand*{\acronymsort}[2]{##2}%
11003 \renewcommand*{\acronymentry}[1]{%
11004 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11005 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1})}%
11006 }

```

g-sc-short-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11007 \renewacronymstyle{long-sc-short-desc}%
11008 {%
11009 \GlsUseAcrEntryDispStyle{long-sc-short}%
11010 }%
11011 {%
11012 \GlsUseAcrStyleDefs{long-sc-short}%
11013 \renewcommand*{\GenericAcronymFields}{}%
11014 \renewcommand*{\acronymsort}[2]{##2}%
11015 \renewcommand*{\acronymentry}[1]{%
11016 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11017 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1})}%
11018 }

```

g-sm-short-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11019 \renewacronymstyle{long-sm-short-desc}%
11020 {%
11021 \GlsUseAcrEntryDispStyle{long-sm-short}%
11022 }%
11023 {%
11024 \GlsUseAcrStyleDefs{long-sm-short}%
11025 \renewcommand*{\GenericAcronymFields}{}%

```

```

11026 \renewcommand*\acronymsort}[2]{##2}%
11027 \renewcommand*\acronymentry}[1]{%
11028   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11029   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11030 }

```

short-long-desc *(short)* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11031 \renewacronymstyle{short-long-desc}%
11032 {%
11033   \GlsUseAcrEntryDispStyle{short-long}%
11034 }%
11035 {%
11036   \GlsUseAcrStyleDefs{short-long}%
11037   \renewcommand*\GenericAcronymFields{}%
11038   \renewcommand*\acronymsort}[2]{##2}%
11039   \renewcommand*\acronymentry}[1]{%
11040     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11041     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11042 }

```

short-long-desc *(long)* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11043 \renewacronymstyle{sc-short-long-desc}%
11044 {%
11045   \GlsUseAcrEntryDispStyle{sc-short-long}%
11046 }%
11047 {%
11048   \GlsUseAcrStyleDefs{sc-short-long}%
11049   \renewcommand*\GenericAcronymFields{}%
11050   \renewcommand*\acronymsort}[2]{##2}%
11051   \renewcommand*\acronymentry}[1]{%
11052     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11053     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11054 }

```

short-long-desc *(long)* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11055 \renewacronymstyle{sm-short-long-desc}%
11056 {%
11057   \GlsUseAcrEntryDispStyle{sm-short-long}%
11058 }%
11059 {%
11060   \GlsUseAcrStyleDefs{sm-short-long}%
11061   \renewcommand*\GenericAcronymFields{}%
11062   \renewcommand*\acronymsort}[2]{##2}%
11063   \renewcommand*\acronymentry}[1]{%
11064     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11065     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11066 }

dua *<long>* only acronym style.

```
11067 \renewacronymstyle{dua}%
11068 {%
```

Check for long form in case this is a mixed glossary.

```
11069 \ifdefempty\glscustomtext
11070 {%
11071 \ifglshaslong{\glslabel}%
11072 {%
11073 \glsifplural
11074 {%
```

Plural form:

```
11075 \glscapscase
11076 {%
```

Plural form, don't adjust case:

```
11077 \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
11078 \glsinsert
11079 }%
11080 {%
```

Plural form, make first letter upper case:

```
11081 \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
11082 \glsinsert
11083 }%
11084 {%
```

Plural form, all caps:

```
11085 \glslongpluralaccessdisplay
11086 {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
11087 \mfirstucMakeUppercase{\glsinsert}%
11088 }%
11089 }%
11090 {%
```

Singular form

```
11091 \glscapscase
11092 {%
```

Singular form, don't adjust case:

```
11093 \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
11094 }%
11095 {%
```

Subsequent singular form, make first letter upper case:

```
11096 \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
11097 }%
11098 {%
```

Subsequent singular form, all caps:

```

11099     \glslongaccessdisplay
11100     {\mfirstucMakeUppercase
11101         {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11102     \mfirstucMakeUppercase{\glsinsert}%
11103     }%
11104     }%
11105     }%
11106     {%

```

Not an acronym:

```

11107     \glsgenentryfmt
11108     }%
11109     }%
11110     {\glscustomtext\glsinsert}%
11111     }%
11112     {%
11113     \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11114     \renewcommand*\acrfullfmt}[3]{%
11115         \glslink[##1]{##2}{%
11116             \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11117             (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
11118     \renewcommand*\Acrfullfmt}[3]{%
11119         \glslink[##1]{##2}{%
11120             \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11121             (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
11122     \renewcommand*\ACRfullfmt}[3]{%
11123         \glslink[##1]{##2}{%
11124             \glslongaccessdisplay
11125             {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11126             (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}}%
11127     \renewcommand*\acrfullplfmt}[3]{%
11128         \glslink[##1]{##2}{%
11129             \glslongpluralaccessdisplay
11130             {\glsentrylongpl{##2}}{##2}##3\space
11131             (\glsshortpluralaccessdisplay
11132             {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
11133     \renewcommand*\Acrfullplfmt}[3]{%
11134         \glslink[##1]{##2}{%
11135             \glslongpluralaccessdisplay
11136             {\Glsentrylongpl{##2}}{##2}##3\space
11137             (\glsshortpluralaccessdisplay
11138             {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
11139     \renewcommand*\ACRfullplfmt}[3]{%
11140         \glslink[##1]{##2}{%
11141             \glslongpluralaccessdisplay
11142             {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11143             (\glsshortpluralaccessdisplay
11144             {\acronymfont{\glsentryshortpl{##2}}}{##2})}}}%
11145     \renewcommand*\glsentryfull}[1]{%

```

```

11146 \glslongaccessdisplay{\glsentrylong{##1}}\space
11147 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11148 }%
11149 \renewcommand*{\Glsentryfull}[1]{%
11150 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11151 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11152 }%
11153 \renewcommand*{\glsentryfullpl}[1]{%
11154 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11155 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11156 }%
11157 \renewcommand*{\Glsentryfullpl}[1]{%
11158 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11159 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11160 }%
11161 \renewcommand*{\acronymentry}[1]{%
11162 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11163 \renewcommand*{\acronymsort}[2]{##1}%
11164 \renewcommand*{\acronymfont}[1]{##1}%
11165 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11166 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11167 \renewacronymstyle{dua-desc}%
11168 {%
11169 \GlsUseAcrEntryDispStyle{dua}%
11170 }%
11171 {%
11172 \GlsUseAcrStyleDefs{dua}%
11173 \renewcommand*{\GenericAcronymFields}{}%
11174 \renewcommand*{\acronymentry}[1]{%
11175 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11176 \renewcommand*{\acronymsort}[2]{##2}%
11177 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11178 \renewacronymstyle{footnote}%
11179 {%
11180 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
11181 }%
11182 {%
11183 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11184 \glshyperfirstfalse
11185 \renewcommand*{\genacrfullformat}[2]{%
11186 \glsshortaccessdisplay
11187 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```



```

11188 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11189 }%
11190 \renewcommand*{\Genacrfullformat}[2]{%
11191 \glsshortaccessdisplay
11192   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11193 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11194 }%
11195 \renewcommand*{\genplacrfullformat}[2]{%
11196 \glsshortpluralaccessdisplay
11197   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11198 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11199 }%
11200 \renewcommand*{\Genplacrfullformat}[2]{%
11201 \glsshortpluralaccessdisplay
11202   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11203 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11204 }%
11205 \renewcommand*{\acronymentry}[1]{%
11206 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11207 \renewcommand*{\acronymsort}[2]{##1}%
11208 \renewcommand*{\acronymfont}[1]{##1}%
11209 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11210 \renewcommand*{\acrfullfmt}[3]{%
11211 \glslink[##1]{##2}{%
11212 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11213 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11214 \renewcommand*{\Acrfullfmt}[3]{%
11215 \glslink[##1]{##2}{%
11216 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11217 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11218 \renewcommand*{\ACRfullfmt}[3]{%
11219 \glslink[##1]{##2}{%
11220 \glsshortaccessdisplay
11221   {\mfirstucMakeUppercase
11222   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11223   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11224 \renewcommand*{\acrfullplfmt}[3]{%
11225 \glslink[##1]{##2}{%
11226 \glsshortpluralaccessdisplay
11227   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11228   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11229 \renewcommand*{\Acrfullplfmt}[3]{%
11230 \glslink[##1]{##2}{%
11231 \glsshortpluralaccessdisplay
11232   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11233   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11234 \renewcommand*{\ACRfullplfmt}[3]{%
11235 \glslink[##1]{##2}{%

```

```

11236 \glsshortpluralaccessdisplay
11237   {\mfirstucMakeUppercase
11238     {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
11239     (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2})}}}%

```

Similarly for \glentryfull etc:

```

11240 \renewcommand*{\glentryfull}[1]{%
11241   \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
11242   (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
11243 \renewcommand*{\Glsentryfull}[1]{%
11244   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
11245   (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
11246 \renewcommand*{\glentryfullpl}[1]{%
11247   \glsshortpluralaccessdisplay
11248   {\acronymfont{\glentryshortpl{##1}}{##1}\space
11249   (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
11250 \renewcommand*{\Glsentryfullpl}[1]{%
11251   \glsshortpluralaccessdisplay
11252   {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11253   (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
11254 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11255 \renewacronymstyle{footnote-sc}%
11256 {%
11257   \GlsUseAcrEntryDispStyle{footnote}%
11258 }%
11259 {%
11260   \GlsUseAcrStyleDefs{footnote}%
11261   \renewcommand{\acronymentry}[1]{%
11262     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11263     \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11264     \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11265 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11266 \renewacronymstyle{footnote-sm}%
11267 {%
11268   \GlsUseAcrEntryDispStyle{footnote}%
11269 }%
11270 {%
11271   \GlsUseAcrStyleDefs{footnote}%
11272   \renewcommand{\acronymentry}[1]{%
11273     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11274     \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11275     \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11276 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11277 \renewacronymstyle{footnote-desc}%
11278 {%
11279   \GlsUseAcrEntryDispStyle{footnote}%
11280 }%
11281 {%
11282   \GlsUseAcrStyleDefs{footnote}%
11283   \renewcommand*{\GenericAcronymFields}{}%
11284   \renewcommand*{\acronymsort}[2]{##2}%
11285   \renewcommand*{\acronymentry}[1]{%
11286     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11287     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11288 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11289 \renewacronymstyle{footnote-sc-desc}%
11290 {%
11291   \GlsUseAcrEntryDispStyle{footnote-sc}%
11292 }%
11293 {%
11294   \GlsUseAcrStyleDefs{footnote-sc}%
11295   \renewcommand*{\GenericAcronymFields}{}%
11296   \renewcommand*{\acronymsort}[2]{##2}%
11297   \renewcommand*{\acronymentry}[1]{%
11298     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11299     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11300 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11301 \renewacronymstyle{footnote-sm-desc}%
11302 {%
11303   \GlsUseAcrEntryDispStyle{footnote-sm}%
11304 }%
11305 {%
11306   \GlsUseAcrStyleDefs{footnote-sm}%
11307   \renewcommand*{\GenericAcronymFields}{}%
11308   \renewcommand*{\acronymsort}[2]{##2}%
11309   \renewcommand*{\acronymentry}[1]{%
11310     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11311     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11312 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11313 \renewcommand*{\newacronymhook}{%
11314   \edef\@gls@keylist{shortaccess=\the\gls\longtok,%
11315     \the\glskeylisttok}%
11316   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11317 }

ltNewAcronymDef Modify default style to use access text:

```
11318 \renewcommand*{\DefaultNewAcronymDef}{%
11319   \edef\@do@newglossaryentry{%
11320     \noexpand\newglossaryentry{\the\glslabeltok}%
11321     {%
11322       type=\acronymtype,%
11323       name={\the\glsshorttok},%
11324       description={\the\glslongtok},%
11325       descriptionaccess=\relax,
11326       text={\the\glsshorttok},%
11327       access={\noexpand\@glo@textaccess},%
11328       sort={\the\glsshorttok},%
11329       short={\the\glsshorttok},%
11330       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11331       shortaccess={\the\glslongtok},%
11332       long={\the\glslongtok},%
11333       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11334       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11335       first={\noexpand\glslongaccessdisplay
11336         {\the\glslongtok}{\the\glslabeltok}\space
11337         {\noexpand\glsshortaccessdisplay
11338           {\the\glsshorttok}{\the\glslabeltok}}},%
11339       plural={\the\glsshorttok\acrpluralsuffix},%
11340       firstplural={\noexpand\glslongpluralaccessdisplay
11341         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11342         {\noexpand\glsshortpluralaccessdisplay
11343           {\noexpand\@glo@shortpl}{\the\glslabeltok}}},%
11344       firstaccess=\relax,
11345       firstpluralaccess=\relax,
11346       textaccess={\noexpand\@glo@shortaccess},%
11347       \the\glskeylisttok
11348     }%
11349   }%
11350   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11351   \let\@org@gls@assign@plural\gls@assign@plural
11352   \let\@org@gls@assign@descplural\gls@assign@descplural
11353   \def\gls@assign@firstpl##1##2{%
11354     \@gls@expand@field{##1}{firstpl}{##2}%
11355   }%
11356   \def\gls@assign@plural##1##2{%
11357     \@gls@expand@field{##1}{plural}{##2}%
11358   }%
11359   \def\gls@assign@descplural##1##2{%
11360     \@gls@expand@field{##1}{descplural}{##2}%
11361   }%
11362   \@do@newglossaryentry
11363   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11364 \let\gls@assign@plural\@org@gls@assign@plural
11365 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11366 }

```

teNewAcronymDef

```

11367 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11368 \edef\@do@newglossaryentry{%
11369 \noexpand\newglossaryentry{\the\glslabeltok}%
11370 {%
11371 type=\acronymtype,%
11372 name={\noexpand\acronymfont{\the\glsshorttok}},%
11373 sort={\the\glsshorttok},%
11374 text={\the\glsshorttok},%
11375 short={\the\glsshorttok},%
11376 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11377 shortaccess={\the\glslongtok},%
11378 long={\the\glslongtok},%
11379 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11380 access={\noexpand\@glo@textaccess},%
11381 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11382 symbol={\the\glslongtok},%
11383 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11384 firstpluralaccess=\relax,
11385 textaccess={\noexpand\@glo@shortaccess},%
11386 \the\glskeylisttok
11387 }%
11388 }%
11389 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11390 \let\@org@gls@assign@plural\gls@assign@plural
11391 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11392 \def\gls@assign@firstpl##1##2{%
11393 \@@gls@expand@field{##1}{firstpl}{##2}%
11394 }%
11395 \def\gls@assign@plural##1##2{%
11396 \@@gls@expand@field{##1}{plural}{##2}%
11397 }%
11398 \def\gls@assign@symbolplural##1##2{%
11399 \@@gls@expand@field{##1}{symbolplural}{##2}%
11400 }%
11401 \@do@newglossaryentry
11402 \let\gls@assign@plural\@org@gls@assign@plural
11403 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11404 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11405 }

```

onNewAcronymDef

```

11406 \renewcommand*{\DescriptionNewAcronymDef}{%
11407 \edef\@do@newglossaryentry{%
11408 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11409  {%
11410      type=\acronymtype,%
11411      name={\noexpand
11412          \acronymformat{\the\glssshorttok}{\the\glslongtok}},%
11413      access={\noexpand\@glo@textaccess},%
11414      sort={\the\glssshorttok},%
11415      short={\the\glssshorttok},%
11416      shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11417      shortaccess={\the\glslongtok},%
11418      long={\the\glslongtok},%
11419      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11420      first={\the\glslongtok},%
11421      firstaccess=\relax,
11422      firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11423      text={\the\glssshorttok},%
11424      textaccess={\the\glslongtok},%
11425      plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11426      symbol={\noexpand\@glo@text},%
11427      symbolaccess={\noexpand\@glo@textaccess},%
11428      symbolplural={\noexpand\@glo@plural},%
11429      firstpluralaccess=\relax,
11430      textaccess={\noexpand\@glo@shortaccess},%
11431      \the\glskeylisttok}%
11432  }%
11433  \let\@org@gls@assign@firstpl\gls@assign@firstpl
11434  \let\@org@gls@assign@plural\gls@assign@plural
11435  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11436  \def\gls@assign@firstpl##1##2{%
11437      \@gls@expand@field{##1}{firstpl}{##2}%
11438  }%
11439  \def\gls@assign@plural##1##2{%
11440      \@gls@expand@field{##1}{plural}{##2}%
11441  }%
11442  \def\gls@assign@symbolplural##1##2{%
11443      \@gls@expand@field{##1}{symbolplural}{##2}%
11444  }%
11445  \@do@newglossaryentry
11446  \let\gls@assign@firstpl\@org@gls@assign@firstpl
11447  \let\gls@assign@plural\@org@gls@assign@plural
11448  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11449  }

```

teNewAcronymDef

```

11450 \renewcommand*{\FootnoteNewAcronymDef}{%
11451     \edef\@do@newglossaryentry{%
11452         \noexpand\newglossaryentry{\the\glslabeltok}%
11453     }%
11454     type=\acronymtype,%
11455     name={\noexpand\acronymfont{\the\glssshorttok}},%

```

```

11456     sort={\the\glsshorttok},%
11457     text={\the\glsshorttok},%
11458     textaccess={\the\glslongtok},%
11459     access={\noexpand\@glo@textaccess},%
11460     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11461     short={\the\glsshorttok},%
11462     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11463     long={\the\glslongtok},%
11464     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11465     description={\the\glslongtok},%
11466     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11467     \the\glskeylisttok
11468   }%
11469 }%
11470 \let\@org@gls@assign@plural\gls@assign@plural
11471 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11472 \let\@org@gls@assign@descplural\gls@assign@descplural
11473 \def\gls@assign@firstpl##1##2{%
11474   \@@gls@expand@field{##1}{firstpl}{##2}%
11475 }%
11476 \def\gls@assign@plural##1##2{%
11477   \@@gls@expand@field{##1}{plural}{##2}%
11478 }%
11479 \def\gls@assign@descplural##1##2{%
11480   \@@gls@expand@field{##1}{descplural}{##2}%
11481 }%
11482 \do@newglossaryentry
11483 \let\gls@assign@plural\@org@gls@assign@plural
11484 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11485 \let\gls@assign@descplural\@org@gls@assign@descplural
11486 }

```

11NewAcronymDef

```

11487 \renewcommand*{\SmallNewAcronymDef}{%
11488   \edef\@do@newglossaryentry{%
11489     \noexpand\newglossaryentry{\the\glslabeltok}%
11490     {%
11491       type=\acronymtype,%
11492       name={\noexpand\acronymfont{\the\glsshorttok}},%
11493       access={\noexpand\@glo@symbolaccess},%
11494       sort={\the\glsshorttok},%
11495       short={\the\glsshorttok},%
11496       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11497       shortaccess={\the\glslongtok},%
11498       long={\the\glslongtok},%
11499       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11500       text={\noexpand\@glo@short},%
11501       textaccess={\noexpand\@glo@shortaccess},%
11502       plural={\noexpand\@glo@shortpl},%

```

```

11503     first={\the\glslongtok},%
11504     firstaccess=\relax,
11505     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11506     description={\noexpand\@glo@first},%
11507     descriptionplural={\noexpand\@glo@firstplural},%
11508     symbol={\the\glsshorttok},%
11509     symbolaccess={\the\glslongtok},%
11510     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11511     \the\glskeylisttok
11512   }%
11513 }%
11514 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11515 \let\@org@gls@assign@plural\gls@assign@plural
11516 \let\@org@gls@assign@descplural\gls@assign@descplural
11517 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11518 \def\gls@assign@firstpl##1##2{%
11519   \@@gls@expand@field{##1}{firstpl}{##2}%
11520 }%
11521 \def\gls@assign@plural##1##2{%
11522   \@@gls@expand@field{##1}{plural}{##2}%
11523 }%
11524 \def\gls@assign@descplural##1##2{%
11525   \@@gls@expand@field{##1}{descplural}{##2}%
11526 }%
11527 \def\gls@assign@symbolplural##1##2{%
11528   \@@gls@expand@field{##1}{symbolplural}{##2}%
11529 }%
11530 \@do@newglossaryentry
11531 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11532 \let\gls@assign@plural\@org@gls@assign@plural
11533 \let\gls@assign@descplural\@org@gls@assign@descplural
11534 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11535 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11536 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11537 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11538 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11539 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```


5.5 Debugging Commands

owglonameaccess

```
11540 \newcommand*{\showglonameaccess}[1]{%
11541   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11542 }
```

owglotextaccess

```
11543 \newcommand*{\showglotextaccess}[1]{%
11544   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11545 }
```

glopluralaccess

```
11546 \newcommand*{\showglopluralaccess}[1]{%
11547   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11548 }
```

wglofirstaccess

```
11549 \newcommand*{\showglofirstaccess}[1]{%
11550   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11551 }
```

rstpluralaccess

```
11552 \newcommand*{\showglofirstpluralaccess}[1]{%
11553   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11554 }
```

glosymbolaccess

```
11555 \newcommand*{\showglosymbolaccess}[1]{%
11556   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11557 }
```

bolpluralaccess

```
11558 \newcommand*{\showglosymbolpluralaccess}[1]{%
11559   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11560 }
```

owglodescaccess

```
11561 \newcommand*{\showglodescaccess}[1]{%
11562   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11563 }
```

escpluralaccess

```
11564 \newcommand*{\showglodescpluralaccess}[1]{%
11565   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11566 }
```

wgloshortaccess

```
11567 \newcommand*{\showgloshortaccess}[1]{%  
11568   \expandafter\show\csname glo@glstetoklabel{#1}@shortaccess\endcsname  
11569 }
```

ortpluralaccess

```
11570 \newcommand*{\showgloshortpluralaccess}[1]{%  
11571   \expandafter\show\csname glo@glstetoklabel{#1}@shortpluralaccess\endcsname  
11572 }
```

owglolongaccess

```
11573 \newcommand*{\showglolongaccess}[1]{%  
11574   \expandafter\show\csname glo@glstetoklabel{#1}@longaccess\endcsname  
11575 }
```

ongpluralaccess

```
11576 \newcommand*{\showglolongpluralaccess}[1]{%  
11577   \expandafter\show\csname glo@glstetoklabel{#1}@longpluralaccess\endcsname  
11578 }
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex. Language support has now been split off into independent language modules.

```
11579 \NeedsTeXFormat{LaTeX2e}
11580 \ProvidesPackage{glossaries-babel}[2016/06/09 v4.25 (NLCT)]
```

Load tracklang to obtain language settings.

```
11581 \RequirePackage{tracklang}
11582 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11583 \AnyTrackedLanguages
11584 {%
11585   \ForEachTrackedDialect{\this@dialect}{%
11586     \IfTrackedLanguageFileExists{\this@dialect}%
11587       {glossaries-}% prefix
11588       {.ldf}%
11589       {%
11590         \RequireGlossariesLang{\CurrentTrackedTag}%
11591       }%
11592       {%
11593         \PackageWarningNoLine{glossaries}%
11594           {No language module detected for ‘\this@dialect’.\MessageBreak
11595             Language modules need to be installed separately.\MessageBreak
11596             Please check on CTAN for a bundle called\MessageBreak
11597             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11598       }%
11599     }%
11600 }%
11601 {}%
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11602 \NeedsTeXFormat{LaTeX2e}
11603 \ProvidesPackage{glossaries-polyglossia}[2016/06/09 v4.25 (NLCT)]
```

Load tracklang to obtain language settings.

```
11604 \RequirePackage{tracklang}
11605 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11606 \AnyTrackedLanguages
```

```

11607 {%
11608   \ForEachTrackedDialect{\this@dialect}{%
11609     \IfTrackedLanguageFileExists{\this@dialect}%
11610     {glossaries-}% prefix
11611     {.ldf}%
11612     {%
11613       \RequireGlossariesLang{\CurrentTrackedTag}%
11614     }%
11615     {%
11616       \PackageWarningNoLine{glossaries}%
11617       {No language module detected for ‘\this@dialect’.\MessageBreak
11618       Language modules need to be installed separately.\MessageBreak
11619       Please check on CTAN for a bundle called\MessageBreak
11620       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11621     }%
11622   }%
11623 }%
11624 {}%

```

Glossary

`makeindex` An indexing application. [10](#), [25](#), [26](#), [174](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [174](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added .. 6
General: Added range facility in format key 109	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 156	\@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol 123
1.04 (2007-08-03)	\@Glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol 122
General: Added \glstextformat 93	\@glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol 121
1.05 (2007-08-10)	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 117
\glossarysection: added \@mkboth to \glossarysection 37	descriptionplural: new 60
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 78	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 77
1.07 (2007-09-13)	descriptionplural support added 77
\@gls@link: fixed bug caused by \theglsentrycounter setting the page number too soon 107	symbolplural support added 77
\glsadd: fixed bug caused by \theglsentrycounter setting the page number too soon 153	\Glsentrydescplural: New 147
1.08 (2007-10-13)	\glsentrydescplural: New 147
General: Added babel support 31	\Glsentrysymbolplural: New 148
listgroup: changed listgroup style to use \glsgetgrouptitle 266	\glsentrysymbolplural: New 148
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 267	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 234
1.1 (2008-02-22)	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 240
\@glossarysection: numbered sections and auto label added 38	symbolplural: new 61
\@gls@tmpb: changed \toksdef to \newtoks 111	1.13 (2008-05-10)
\@gls@toc: numberline added 40	General: fixed bug that ignored 3rd parameter 125–132
\@p@glossarysection: numbered sections and auto label added 39	\ACRfullpl: new 215
General: msgen now loaded (\new@ifnextchar needed) 4	
translate: translate option added 22	
\setglossarysection: new 38	
numberedsection: numberedsection package option added 7	

<code>\Acrfullpl</code> : new	214	<code>\@gls@</code> : Test glossary type is <code>\acronymtype</code>	
<code>\acrfullpl</code> : new	214	in addition to checking if footnote option	
<code>\acrpluralsuffix</code> : New	212	has been used	119
<code>\gls@defglossaryentry</code> : Changed default		<code>\@glsdisp</code> : Test glossary type is	
first value	77	<code>\acronymtype</code> in addition to checking	
Changed default firstplural value	77	if footnote option has been used	124
Removed restriction on only using		<code>\@glspl@</code> : Test glossary type is	
<code>\newglossaryentry</code> in the preamble	83	<code>\acronymtype</code> in addition to checking	
<code>\newacronym</code> : Removed restriction on		if footnote option has been used	121
only using <code>\newacronym</code> in the preamble	212	<code>\@glstarget</code> : raised the hypertarget so	
1.14 (2008-06-17)		the target text doesn't scroll off the top	
<code>\@gls@hypergroup</code> : new	262	of the page	118
General: added nonnumberlist key to		<code>\gls@defglossaryentry</code> : Changed def	
<code>\printglossary</code>	198	to let	77
added numberedsection key to		1.17 (2008-12-26)	
<code>\printglossary</code>	196	<code>\@do@wrglossary</code> : new	177
<code>\firstacronymfont</code> : new	215	<code>\@do@seeglossary</code> : new	180
<code>\glsautoprefix</code> : new	7	<code>\@glo@storeentry</code> : new	84
<code>\glsnavhyperlink</code> : changed <code>\edef</code> to		<code>\@gls@glossary</code> : changed definition to	
<code>\protected@edef</code>	261	use <code>\index</code> instead of <code>\@index</code> ...	175
<code>\glsnavhypertarget</code> : added write to aux		<code>\@glsdefaultplural</code> : new	65
file	261	<code>\@glsdefaultsort</code> : new	65
<code>\glsnavigation</code> : changed to only use labels		<code>\@glshypernumber</code> : new	208
for groups that are present	262	<code>\@glsnoname</code> : new	64
1.15 (2008-08-15)		<code>\@glsnonextpages</code> : new	198
<code>\@gls@link</code> : added <code>\glslabel</code>	107	General: added xindy support	25
<code>\gls@defglossaryentry</code> : check for		parent: new	62
<code>\@glo@first</code> in description	81	see: new	62
check for <code>\@glo@text</code> in symbol	82	<code>\gls@defglossaryentry</code> : added non-	
<code>\gls@hypergroup</code> prerun: new	262	numberlist key	78
<code>\glsnavhypertarget</code> : added check if re-		added parent key	78
run required	261	added see key	78
<code>\glssettoctitle</code> : new	30	Stored main part of entry format when	
<code>\printglossary</code> : changed the way the		entry is defined	82
TOC title is set	182	<code>\gls@suffixF</code> : new	35
1.16 (2008-08-27)		<code>\gls@suffixFF</code> : new	36
<code>\@GLS@</code> : Test glossary type is <code>\acronymtype</code>		<code>\gls@wrglossary</code> : modified to allow for	
in addition to checking if footnote option		xindy support	175
has been used	121	<code>\glsnavhyperlink</code> : new	153
<code>\@GLSpl</code> : Test glossary type is		<code>\glshypernumber</code> : modified to allow ma-	
<code>\acronymtype</code> in addition to checking		terial to be attached to location	208
if footnote option has been used	123	<code>\glsnavhyperlink</code> : replaced <code>\hyperlink</code>	
<code>\@Gls@</code> : Test glossary type is <code>\acronymtype</code>		to <code>\@glslink</code>	261
in addition to checking if footnote option		<code>\glsnavhypertarget</code> : replaced	
has been used	120	<code>\hypertarget</code> to <code>\@glstarget</code> ...	261
<code>\@Glspl@</code> : Test glossary type is		<code>\glssee</code> : new	181
<code>\acronymtype</code> in addition to checking		<code>\glsseeformat</code> : new	181
if footnote option has been used	122	<code>\glsSetSuffixF</code> : new	35
		<code>\glsSetSuffixFF</code> : new	36

<code>\ifglxindy</code> : new	25	before term is displayed to prevent un-	
<code>\istfilename</code> : added xindy support ...	34	wanted whatsit	108
<code>\newglossarystyle</code> : made <code>\newglossarystyle</code>		<code>\forallglossaries</code> : replaced <code>\ifthenelse</code>	
<code>long</code>	207	with <code>\ifx</code>	49
<code>\nopostdesc</code> : new	33	<code>\forallglsentries</code> : replaced <code>\ifthenelse</code>	
<code>nonumberlist</code> : new	62	with <code>\ifx</code>	49
<code>\printglossary</code> : added check to deter-		<code>\glsdefmain</code> : new	13
mine if <code>\printglossary</code> is already		<code>\glsdescwidth</code> : changed <code>\linewidth</code> to	
defined	182	<code>\hsize</code>	269, 291
added print language to aux file	182	<code>\glslistdottedwidth</code> : changed	
<code>order</code> : order package option added	25	<code>\linewidth</code> to <code>\hsize</code>	268
<code>\writeist</code> : added xindy support	156	<code>\glspagelistwidth</code> : changed <code>\linewidth</code>	
1.18 (2009-01-14)		to <code>\hsize</code>	269, 291
<code>\@gls@loadlist</code> : new	9	<code>nomain</code> : added <code>nomain</code> package option .	13
<code>\@gls@loadlong</code> : new	8	<code>\writeist</code> : removed <code>item_02</code> - no such	
<code>\@gls@loadsuper</code> : new	8	<code>makeindex</code> key	160
<code>\@gls@loadtree</code> : new	9	2.02 (2007-07-13)	
<code>\gls@defglossaryentry</code> : Changed de-		<code>\@printglossary</code> : suppressed warning	
fault value of <code>sort</code> to <code>\@glsdefaultsort</code>		globally rather than locally	185
.....	78	2.02 (2009-07-13)	
moved <code>sort</code> sanitization to <code>\newglossaryentry</code>		<code>\glossarysection</code> : changed <code>\@mkboth</code>	
.....	82	to <code>\glossarymark</code>	37
<code>\glstarget</code> : new	201	<code>\gls glossarymark</code> : New	37
<code>\oldacronym</code> : new	211	2.03 (2009-09-23)	
<code>nolist</code> : new	9	<code>\@GLS@</code> : Added check for <code>hyperfirst</code>	121
<code>nolong</code> : new	8	<code>\@GLSpl</code> : Added check for <code>hyperfirst</code> ...	123
<code>sort</code> : moved sanitization to <code>\newglossaryentry</code>		<code>\@Gls@</code> : Added check for <code>hyperfirst</code>	120
.....	60	<code>\@Glspl@</code> : Added check for <code>hyperfirst</code> ..	122
<code>nostyles</code> : new	9	<code>\@gls@</code> : Added check for <code>hyperfirst</code>	119
<code>nosuper</code> : new	9	<code>\@gls@link</code> : new	106
<code>notree</code> : new	9	<code>\@gls@link</code> : added <code>\leavevmode</code>	107
1.19 (2009-03-02)		Moved entry existence check to avoid	
<code>\gls clearpage</code> : new	40	duplicate code	107
<code>\glsdisp</code> : new	123	<code>\@glsdisp</code> : Added check for <code>hyperfirst</code> .	124
<code>\SetDescriptionAcronymStyle</code> :		<code>\@glspl@</code> : Added check for <code>hyperfirst</code> ..	121
changed <code>\acronymfont</code> to use		<code>\gls glossarymark</code> : Added check to see if	
<code>\textsmaller</code> instead of <code>\smaller</code> 238		it's already defined	37
<code>\SetDescriptionFootnoteAcronymStyle</code> :		<code>hyperfirst</code> : new	24
changed <code>\acronymfont</code> to use		2.04 (2009-11-10)	
<code>\textsmaller</code> instead of <code>\smaller</code> 234		<code>\@GLS@</code> : Changed test to check if glossary	
<code>\SetFootnoteAcronymStyle</code> : changed		type has been identified as a list of	
<code>\acronymfont</code> to use <code>\textsmaller</code>		acronyms	121
instead of <code>\smaller</code>	240	<code>\@GLSpl</code> : Changed test to check if glos-	
<code>\SetSmallAcronymStyle</code> : changed		sary type has been identified as a list	
<code>\acronymfont</code> to use <code>\textsmaller</code>		of acronyms	123
instead of <code>\smaller</code>	243	<code>\@Gls@</code> : Changed test to check if glossary	
2.01 (2009 May 30)		type has been identified as a list of	
<code>\@gls@link</code> : moved <code>\@do@wrglossary</code>		acronyms	120

<code>\@Glspl@</code> : Changed test to check if glossary type has been identified as a list of acronyms	122	<code>\SetDUADisplayStyle</code> : new	243
<code>\@glossaryentryfield</code> : new	83	<code>\SetFootnoteAcronymDisplayStyle</code> : new	238
<code>\@glossarysubentryfield</code> : new	83	<code>\SetSmallAcronymDisplayStyle</code> : new	241
<code>\@gls@</code> : Changed test to check if glossary type has been identified as a list of acronyms	119	2.05 (2010-02-06)	
<code>\@glsacronymlists</code> : new	14	<code>\@glsdisp</code> : Added closing brace. Patch provided by Sergiu Dotenco	124
<code>\@glsdisp</code> : Changed test to check if glossary type has been identified as a list of acronyms	124	Removed spurious brace. Patch provided by Sergiu Dotenco	124
<code>\@glspl@</code> : Changed test to check if glossary type has been identified as a list of acronyms	121	<code>\writeist</code> : Added <code>\string</code> before opening and closing braces. Patch provided by Segiu Dotenco	161
<code>\@newglossaryentryposthook</code> : new ..	83	2.06 (2010-06-14)	
<code>\@newglossaryentryprehook</code> : new ..	83	<code>\altnewglossary</code> : new	58
<code>acronymlists</code> : new	16	<code>\CustomAcronymFields</code> : new	246
<code>\DeclareAcronymList</code> : new	15	<code>\CustomNewAcronymDef</code> : new	246
<code>\DefineAcronymSynonyms</code> : new	228	<code>\SetCustomDisplayStyle</code> : new	245
<code>\gls@defglossaryentry</code> : added user1-6 keys	78	<code>\SetCustomStyle</code> : new	246
<code>\glsadd</code> : fixed bug that ignored counter ..	153	2.07 (2010-07-10)	
<code>\Glsentryuseri</code> : new	149	General: <code>glsadd</code> format key stored in <code>\@glsnumberformat</code> (was mistakenly stored in <code>\@glo@format</code>)	153
<code>\Glsentryuserii</code> : new	149	3.0 (2010-07-12)	
<code>\Glsentryuseriii</code> : new	149	<code>\@makeglossary</code> : Added check for <code>savewrites</code>	165
<code>\Glsentryuseriiii</code> : new	149	<code>\gls@wrglossary</code> : modified to take into account <code>savewrites</code>	175
<code>\Glsentryuseriv</code> : new	150	3.0 (2010/03/31)	
<code>\Glsentryuseriv</code> : new	150	<code>\@set@glo@numformat</code> : added 4th argument	109
<code>\Glsentryuseriv</code> : new	150	3.0 (2011-04-02)	
<code>\Glsentryuseriv</code> : new	150	<code>\@do@wrglossary</code> : added check for hyper location prefix	178
<code>\Glsentryuseriv</code> : new	150	modified to use new format	177
<code>\Glsentryuseriv</code> : new	150	<code>\@glossarysec</code> : replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	6
<code>\ns@newglossary</code> : added check to determine if <code>\gls@{type}@display</code> and <code>\gls@{type}@displayfirst</code> have been defined.	57	<code>\@do@seeglossary</code> : Sanitize and escape cross-referencing information	180
<code>\SetAcronymLists</code> : new	16	<code>\@gls@counterwithin</code> : new	10
<code>\SetDefaultAcronymDisplayStyle</code> : new	230	<code>\@gls@ifinlist</code> : new	41
<code>\SetDefaultAcronymStyle</code> : new	231	<code>\@gls@link</code> : added <code>\@gls@saveentrycounter</code>	108
<code>\SetDescriptionAcronymDisplayStyle</code> : new	236	added <code>\@gls@setsort</code>	108
<code>\SetDescriptionDUAAcronymDisplayStyle</code> : new	234	<code>\@gls@saveentrycounter</code> : new	108
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> : new	232	<code>\@gls@setupsort@def</code> : new	11
		<code>\@gls@setupsort@standard</code> : new	11
		<code>\@gls@setupsort@use</code> : new	12
		<code>\@gls@xdy@locationlist</code> : new	44

<code>\@glslink:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	117	<code>\glsentryshort:</code> new	150
<code>\@glsnextpages:</code> new	198	<code>\Glsentryshortpl:</code> new	151
<code>\@print@glossary:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	185	<code>\glsentryshortpl:</code> new	150
<code>\@printglossary:</code> added <code>\currentglossary</code>	184	<code>\glsgetgrouptitle:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	205
added <code>\glsnextpages</code>	184	<code>\gls glossarymark:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	37
make toctitle default to title	184	<code>\gls hyperlink:</code> changed default from <code>\glsentryname</code> to <code>\glsentrytext</code>	153
<code>\@xdyattributelist:</code> new	41	<code>\gls hypernumber:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	208
General: added prefix to hyperlink	209	<code>\gls numberformat:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	36
etoolbox now loaded	4	<code>\gls reformat:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	36
replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	29, 32, 104, 196	<code>\gls reformat:</code> new	200
<code>\acrfootnote:</code> new	231	<code>\gls resetsubentrycounter:</code> new ...	199
<code>\ACRfull:</code> added starred version	213	<code>\gls seeitem:</code> hyperlink uses <code>\glsseeitemformat</code> instead of <code>\glsentryname</code>	181
<code>\Acrfull:</code> added starred version	213	<code>\gls seeitemformat:</code> new	181
<code>\acrfull:</code> added starred version	212	<code>\gls sortnumberfmt:</code> new	11
<code>\ACRfullpl:</code> added starred version ...	215	<code>\gls stepentry:</code> new	200
<code>\Acrfullpl:</code> added starred version ...	214	<code>\gls stepsubentry:</code> new	200
<code>\acrfullpl:</code> added starred version ...	214	<code>\gls subentrycounterlabel:</code> new ...	201
<code>\acrlinkfootnote:</code> new	231	<code>\gls subentryitem:</code> new	201
<code>\acrno linkfootnote:</code> new	232	<code>theglossary:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	201
<code>savewrites:</code> new	26	<code>short:</code> new	64
see: added <code>\@glo@seeautonumberlist</code>	62	<code>shortplural:</code> new	64
<code>seeautonumberlist:</code> new	8	<code>\ifglossaryexists:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	50
<code>\glossarysection:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	37	<code>\ifglsentryexists:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	51
<code>\glossarystyle:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	207	<code>\istfile:</code> deprecated	173
<code>\gls@codepage:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	25	<code>glossaryentry:</code> new	199
<code>\gls@def glossaryentry:</code> added <code>\@gls@defsort</code>	82	<code>glossarysubentry:</code> new	199
added short and long keys	78	<code>\newglossaryentry:</code> replaced <code>\DeclareRobustCommand</code> with <code>\newrobustcmd</code>	67
replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	78	<code>\newglossarystyle:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	207
<code>\gls@docclearpage:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	39	<code>\ns@newglossary:</code> added <code>\@gls@defsortcount</code>	58
<code>\glsadd:</code> added <code>\@gls@saveentrycounter</code>	154	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	57
<code>\GlsAddXdyCounters:</code> new	41	<code>entrycounter:</code> new	10
<code>\glsentrycounterlabel:</code> new	200	<code>entrycounterwithin:</code> new	10
<code>\glsentryitem:</code> new	201	<code>\oldacronym:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	211
<code>\Glsentrylong:</code> new	151	<code>compatible-2.07:</code> compatible-2.07 op- tion added	27
<code>\glsentrylong:</code> new	151	<code>long:</code> new	64
<code>\Glsentrylongpl:</code> new	151		
<code>\glsentrylongpl:</code> new	151		
<code>\Glsentryshort:</code> new	150		

longplural: new	64	\Acrfull: made robust	213
nonumberlist: now boolean	62	\acrfull: made robust	212
sort: new	10	\acrfullformat: removed \acronymfont as it should already be set in the sec- ond argument.	213
counter: replaced \@ifundefined with \ifcsundef	61	\ACRfullpl: made robust	215
\printglossary: replaced \@ifundefined with \ifcsundef	182	\Acrfullpl: made robust	214
\SetDescriptionFootnoteAcronymDisplayStyle expanded options link options	232	\acrfullpl: made robust	214
\setentrycounter: added optional ar- gument	206	\ACRlong: made robust	142
\showacronymlists: new	252	\Acrlong: made robust	141
\showglocounter: new	249	\acrlong: made robust	140
\showgloDESC: new	250	\ACRlongpl: made robust	144
\showgloDESCplural: new	250	\Acrlongpl: made robust	143
\showglofirst: new	248	\acrlongpl: made robust	142
\showglofirstpl: new	248	\ACRshort: made robust	138
\showgloflag: new	251	\Acrshort: made robust	137
\showgloindex: new	251	\acrshort: made robust	137
\showglolevel: new	248	\ACRshortpl: made robust	140
\showgloNAME: new	250	\Acrshortpl: made robust	139
\showgloparent: new	247	\acrshortpl: made robust	139
\showgloplural: new	248	\Gls: made robust	119
\showglosort: new	250	\glsadd: made robust	153
\showglossaries: new	252	\glsaddall: made robust	154
\showglossarycounter: new	253	\GLSdesc: made robust	129
\showglossaryentries: new	253	\Glsdesc: made robust	129
\showglossaryin: new	252	\glsdesc: made robust	129
\showglossaryout: new	252	\GLSdescplural: made robust	130
\showglossarytitle: new	252	\Glsdescplural: made robust	130
\showglosymbol: new	250	\glsdescplural: made robust	129
\showglosymbolplural: new	251	\glsfirst: made robust	125
\showglotext: new	248	\GLSfirstplural: made robust	128
\showglotype: new	248	\Glsfirstplural: made robust	127
\showglouserI: new	249	\glsfirstplural: made robust	127
\showglouserII: new	249	\glslink: made robust	106
\showglouserIII: new	249	\GLSname: made robust	128
\showglouserIV: new	249	\Glsname: made robust	128
\showglouserVI: new	249	\glsname: made robust	128
\showglouserVI: new	250	\GLSpl: made robust	123
subentrycounter: new	10	\Glspl: made robust	122
\writeist: added xindy-only macro defi- nitions to glossary open tag	158	\glspl: made robust	121
modified to support new format	156	\GLSplural: made robust	127
3.01 (2011-04-12)		\GLSsymbol: made robust	131
\@glswritefiles: added check for empty glossaries	173	\Glsymbol: made robust	131
General: made robust	120	\glsymbol: made robust	130
\ACRfull: made robust	213	\GLSsymbolplural: made robust	132
		\Glsymbolplural: made robust	131
		\glsymbolplural: made robust	131
		\Glstext: made robust	125
		\glstext: made robust	125

<code>\GLSuseri</code> : made robust	132	<code>\glsentryfullpl</code> : fixed bug (re-	
<code>\Glsuseri</code> : made robust	132	placed <code>\glsentryshort</code> with	
<code>\glsuseri</code> : made robust	132	<code>\glsentryshortpl</code>)	151
<code>\GLSuserii</code> : made robust	133	<code>\glsentrynumberlist</code> : new	152
<code>\Glsuserii</code> : made robust	133	<code>\glsmoveentry</code> : new	83
<code>\glsuserii</code> : made robust	133	<code>\glsresetsubentrycounter</code> : new ...	200
<code>\GLSuseriii</code> : made robust	134	<code>\ifglshaschildren</code> : new	52
<code>\Glsuseriii</code> : made robust	134	<code>\ifglshasparent</code> : new	53
<code>\glsuseriii</code> : made robust	134	<code>\makeglossaries</code> : added list parser ..	168
<code>\GLSuseriv</code> : made robust	135	<code>indexonlyfirst</code> : new	24
<code>\Glsuseriv</code> : made robust	135	<code>\renewglossarystyle</code> : new	207
<code>\glsuseriv</code> : made robust	134	<code>\showglossaryentries</code> : fixed misspelt	
<code>\GLSuserv</code> : made robust	136	command	253
<code>\Glsuserv</code> : made robust	135	<code>\SmallNewAcronymDef</code> : fixed broken	
<code>\glsuserv</code> : made robust	135	short and long plural	241
<code>\GLSuservi</code> : made robust	137	3.03 (2012/09/21)	
<code>\Glsuservi</code> : made robust	136	<code>\@gls@sanitizesort</code> : new	18
<code>\glsuservi</code> : made robust	136	<code>\@gls@setupsort@standard</code> : used	
3.02 (2012-05-19)		<code>\@gls@sanitizesort</code>	11
<code>\glsnumlistlastsep</code> : new	153	<code>\@printglossary</code> : allow title to override	
<code>\glsnumlistsep</code> : new	153	default toctitle	183
3.02 (2012-05-21)		General: allow title to set toctitle	195
<code>\@do@wrglossary</code> : changed <code>\@gls@locref</code>		<code>\glsinlinedescformat</code> : new	265
to <code>\theglentrycounter</code>	179	<code>\glsinlineemptydescformat</code> : new ..	265
<code>\@do@wrglossary</code> : changed <code>\@do@wr@glossary</code>		<code>\glsinlinenameformat</code> : new	265
to test for <code>indexonlyfirst</code> option; put		<code>\glsinlinepostchild</code> : new	265
old <code>\@do@wr@glossary</code> code into		<code>\glsinlinesubdescformat</code> : new	265
<code>\@do@wrglossary</code>	175	<code>\glsinlinesubnameformat</code> : new	265
<code>\@gls@missingnumberlist</code> : new	65	<code>\glspostinline</code> : replaced “.” with	
<code>\@gls@writefiles</code> : added check		<code>\glspostdescription</code>	265
for existence of token in case		<code>altlongragged4col</code> : added check for	
<code>\makeglossaries</code> has been		<code>glsnogroupskip</code>	284
omitted	173	<code>altsuperragged4col</code> : added check for	
<code>\@printglossary</code> : add a way to fetch		<code>glsnogroupskip</code>	302
current entry label	184	<code>alttree</code> : added check for <code>glsnogroupskip</code>	
<code>savenumberlist</code> : new	8	311
<code>ucmark</code> : new	9	index: added check for <code>glsnogroupskip</code>	305
<code>\gls@defglossaryentry</code> : added num-		<code>nogroupskip</code> : new	9
berlist element	81	long: added check for <code>glsnogroupskip</code> .	270
<code>\gls@save@numberlist</code> : new	182	<code>long3col</code> : added check for <code>glsnogroup-</code>	
<code>\gls@wrglossary</code> : added check for glos-		<code>skip</code>	271
sary file defined	175	<code>long4col</code> : added check for <code>glsnogroup-</code>	
<code>\glsdisplaynumberlist</code> : new	152	<code>skip</code>	273
<code>\glsentrycounter</code> : set default value ..	108	<code>longragged</code> : added check for <code>glsnogroup-</code>	
<code>\Glsentryfull</code> : fixed bug (re-		<code>skip</code>	280
placed <code>\glsentryshortpl</code> with		<code>longragged3col</code> : added check for	
<code>\glsentryshort</code>)	151	<code>glsnogroupskip</code>	282
		<code>nopostdot</code> : new	9
		<code>tree</code> : added check for <code>glsnogroupskip</code> .	306

treenoname: added check for glsnogroup-skip	308	mcolindexspannav: replaced '2' with \glsmcols	286
super: added check for glsnogroupskip	292	mcoltree: replaced '2' with \glsmcols	287
super3col: added check for glsnogroup-skip	293	mcoltreenoname: replaced '2' with \glsmcols	288
super4col: added check for glsnogroup-skip	295	mcoltreespannav: replaced '2' with \glsmcols	288
superragged: added check for glsnogroupskip	298	\gls@protected@pagefmts: added Roman to list	176
superragged3col: added check for glsnogroupskip	300	\gls@Romanpage: new	176
3.04 (2012-11-11)		\glsgetgrouplabel: fixed bug (typo in \equal)	206
altlist: replaced \newline with paragraph break	267	\nopostdesc: made robust	33
3.04 (2012-11-18)		3.05 (2013/04/21)	
\@do@wrglossary: changed \theglsentrycount back to \glslocref	179	\@gls@nohyperlist: new	16
\@do@wrglossary: modified to compensate for possible incorrect page number	177	\@GlsDeclareNoHyperList: new	16
\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage	110	nohypertypes: new	16
\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here.	185	3.06 (2013/06/17)	
General: Added check for doc package ...	4	\@xdy@main@language: Changed back to using \languagename	25
added datatool-base as a required package	4	\findrootlanguage: Obsoleted	47
added local key	104	3.07 (2013-07-05)	
\gls@Alphpage: new	176	\@gls@link: fixed bug that failed to find entry in list	107
\gls@alphpage: new	176	\glossarypreamble: modified to work with \setglossarypreamble	36
\gls@disablepagerefexpansion: new	176	\gls@docclearpage: added check for openright	39
\gls@numberpage: new	176	\glspostdescription: Added spacefactor code	9
\gls@protected@pagefmts: new	176	\GlsSetXdyCodePage: Added check for fontspec	48
\gls@romanpage: new	176	\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	236
\glsdefmain: added check for doc package	13	\setglossarypreamble: new	37
\glsorg@endtheglossary: new	5	3.08a (2013-08-30)	
\glsorg@theglossary: new	5	list: updated list style to use \glossentry and \subglossentry	266
\PrintChanges: new	5	listdotted: updated listdotted style to use \glossentry and \subglossentry	268
3.05 (2013-04-21)		altlist: updated altlist style to use \glossentry and \subglossentry	267
\@do@wrglossary: add Roman case. Fixed bugs in the else statements ..	177	inline: updated inline style to use \glossentry and \subglossentry	264
\@gls@link: added check for "nohypertypes"	107	3.08a (2013-09-28)	
mcolalttree: replaced '2' with \glsmcols	289	\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	84
mcolindex: replaced '2' with \glsmcols	285		

updated for \glossentry	84	tree: updated to use \glossentry and	
\@glossaryentryfield: switched to		\subglossentry	306
\glossentry	83	\setglossarystyle:new	206
\@glossarysubentryfield: switched to		\setglossentrycompatibility:new	203
\subglossentry	83	superragged: updated to use	
General: added nogroupskip key to		\glossentry and \subglossentry	298
\printglossary	196	3.09a (2013-10-09)	
removed definition of \@glossaryentryfield		\@gls@assign@symbolplural@field:	
.....	353	new	18
removed definition of \@glossarysubentryfield		\@gls@default@value: new	61
.....	353	\Glsentrydesc: made robust	146
\compatibleglossentry:new	202	\Glsentrydescplural: made robust ..	147
\compatiblesubglossentry:new ...	203	\Glsentryfirst: made robust	148
\glossaryentryfield: deprecated ...	204	\Glsentryfirstplural: made robust .	148
\Glossentrydesc: new	202	\Glsentryfull: made robust	151
\glossentrydesc: new	202	\Glsentryfullpl: made robust	151
\Glossentryname: new	202	\Glsentrylong: made robust	151
\glossentryname: new	202	\Glsentrylongpl: made robust	151
\Glossentrysymbol: new	203	\Glsentryname: made robust	145
\glossentrysymbol: new	203	\Glsentryplural: made robust	147
\gls@assign@desc@field: new	18	\Glsentryshort: made robust	150
\gls@assign@descplural@field: new	18	\Glsentryshortpl: made robust	151
\gls@assign@field: new	67	\Glsentrysymbol: made robust	147
\gls@ifnotmeasuring: new	85	\Glsentrysymbolplural: made robust	148
\glsaddallunused: new	154	\Glsentrytext: made robust	147
\glsexpandfields: new	67	\Glsentryuseri: made robust	149
\glsnoexpandfields: new	67	\Glsentryuserii: made robust	149
\glssee: made robust	181	\Glsentryuseriii: made robust	149
\glsseeformat: made robust	181	\Glsentryuseriv: made robust	150
\glsseeitem: made robust	181	\Glsentryuserv: made robust	150
\glsseelist: made robust	181	\Glsentryuservi: made robust	150
\ifglshdescsuppressed: new	53	\glstextup: new	212
\ifglshasdesc: new	53	\ifglshassymbol: changed test to check	
\ifglshassymbol: new	53	for \@gls@default@symbol	53
altlongragged4col: updated to use		3.10a (2013-09-28)	
\glossentry and \subglossentry	283	\gls@assign@type@field: new	18
alttree: updated to use \glossentry		3.10a (2013-10-13)	
and \subglossentry	309	\@gls@keymap: new	69
index: added paragraph break at end of		\@gls@provide@newglossary: new ...	56
environment	304	\@gls@writedef: new	68
updated to use \glossentry and		\@gls@defaultplural: Obsolete	65
\subglossentry	304	\@glsnodesc: new	64
long: updated to use \glossentry and		\@print@glossary: Added providecom-	
\subglossentry	269	mand code to aux file	185, 186
longragged: updated to use \glossentry		\gls@defglossaryentry: Changed to	
and \subglossentry	280	using \@gls@default@value	77, 78
longragged3col: updated to use		new	77
\glossentry and \subglossentry	282	\gls@writedefhook: new	76

<code>\makeglossaries:</code> Added providecommand code to aux file	167	<code>\@acrshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	351
<code>\new@glossaryentry:</code> new	68	<code>\@gls@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	119
<code>\ns@newglossary:</code> added <code>\@gls@provide@newglossary</code>	57	change to using <code>\glsentryfmt</code> style commands	119
3.11a (2013-10-15)		<code>\@gls@noexpand@fields:</code> Fixed bug expand replaced with noexpand	66
<code>\@ACRlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	353	<code>\@glsdisp:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	124
<code>\@ACRshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	351	change to using <code>\glsentryfmt</code> style commands	124
<code>\@Acrlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	352	<code>\@glspl@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	121
<code>\@Acrshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	351	change to using <code>\glsentryfmt</code> style commands	121
<code>\@GLS@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	120	General: added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	137-144
change to using <code>\glsentryfmt</code> style commands	121	changed to just use <code>\Glsentrydescplural</code>	130
removed <code>\MakeUppercase</code> (now moved to <code>\glsentryfmt</code>)	121	changed to just use <code>\glsentrydescplural</code>	130
<code>\@GLSpl:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	123	changed to just use <code>\Glsentrydesc</code> .	129
change to using <code>\glsentryfmt</code> style commands	123	changed to just use <code>\Glsentrydesc</code> .	129
removed <code>\MakeUppercase</code> as now dealt with in <code>\glsentryfmt</code>	123	changed to just use <code>\Glsentryfirstplural</code>	127
<code>\@Gls@:</code> add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	120	changed to just use <code>\glsentryfirstplural</code>	127, 128
change to using <code>\glsentryfmt</code> style commands	120	changed to just use <code>\Glsentryfirst</code>	126
removed <code>\makefirstuc</code> (now dealt with in <code>\glsentryfmt</code>)	120	changed to just use <code>\glsentryfirst</code>	126
<code>\@Glspl@:</code> add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	122	changed to just use <code>\Glsentryname</code> .	128
change to using <code>\glsentryfmt</code> style commands	122	128, 129
removed <code>\makefirstuc</code> (now dealt with in <code>\glsentryfmt</code>)	122	changed to just use <code>\Glsentryplural</code>	127
<code>\@acrlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	352	changed to just use <code>\glsentryplural</code>	126, 127
		changed to just use <code>\Glsentrysymbolplural</code>	132
		changed to just use <code>\glsentrysymbolplural</code>	131, 132
		changed to just use <code>\Glsentrysymbol</code>	131
		changed to just use <code>\glsentrysymbol</code>	130, 131
		Changed to just use <code>\Glsentrytext</code> .	125

changed to just use <code>\glentrytext</code>	125	<code>\ns@newglossary</code> : replaced <code>\glsdisplay</code>	
changed to just use <code>\Glsentryuseriii</code>		and <code>\glsdisplayfirst</code> with	
.....	134	<code>\glentryfmt</code>	57
changed to just use <code>\glentryuseriii</code>		compatible-3.07: <code>cnew</code>	27
.....	134	<code>\SetCustomDisplayStyle</code> : updated to	
changed to just use <code>\Glsentryuserii</code>	133	use <code>\defglentryfmt</code>	245
changed to just use <code>\glentryuserii</code>	133	<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to just use <code>\glentryuseriv</code>	135	changed to use <code>\defglentryfmt</code>	230
changed to just use <code>\glentryuseriv</code>	135	<code>\SetDescriptionAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuseri</code>	132	updated to use <code>\defglentryfmt</code>	236
changed to just use <code>\glentryuseri</code>		<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
.....	132, 133	updated to use <code>\defglentryfmt</code>	234
changed to just use <code>\Glsentryuservi</code>	136	<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
changed to just use <code>\glentryuservi</code>		updated to use <code>\defglentryfmt</code>	232
.....	136, 137	<code>\SetDUADisplayStyle</code> : updated to use	
changed to just use <code>\Glsentryuserv</code>	136	<code>\defglentryfmt</code>	243
changed to just use <code>\glentryuserv</code>		<code>\SetFootnoteAcronymDisplayStyle</code> :	
.....	135, 136	updated to use <code>\defglentryfmt</code>	238
Now requires <code>textcase</code>	4	<code>\SetSmallAcronymDisplayStyle</code> : up-	
acronymlists: replaced <code>\@addtoacronymlists</code>		dated to use <code>\defglentryfmt</code>	241
with <code>\DeclareAcronymList</code>	16	<code>\setupglossaries</code> : new	28
<code>\defglsdisplay</code> : obsoleted	103	<code>\showglong</code> : new	251
<code>\defglsdisplayfirst</code> : obsoleted	103	<code>\showgshort</code> : new	251
<code>\defglentryfmt</code> : new	56	numbers: new	27
<code>\forglentries</code> : replaced <code>\ifx</code> with		symbols: new	27
<code>\ifdefempty</code>	49	3.12a (2013-10-16)	
<code>\gls@assign@desc</code> : new	76	<code>\gls@defglossaryentry</code> : added	
<code>\gls@defglossaryentry</code> : Fixed default		<code>\glslabel</code>	77
counter if none supplied	81	<code>\glsaddkey</code> : new	71
<code>\gls@doentryfmt</code> : new	56	3.13a (2013-11-05)	
<code>\glsdisplay</code> : obsoleted	102	<code>\@gls@assign@symbol@field</code> : changed	
<code>\glsdisplayfirst</code> : obsoleted	102	to use <code>\glssetnoexpandfield</code>	18
<code>\glsentryfmt</code> : new	97	<code>\@gls@assign@symbolplural@field</code> :	
<code>\glsgetgrouptitle</code> : Added check in		changed to use <code>\glssetnoexpandfield</code>	18
case non-Latin alphabet in use	205	<code>\@gls@link</code> : removed <code>\relax</code>	108
<code>\gls glossarymark</code> : replaced <code>\MakeUppercase</code>		<code>\@gls@notranslatorhook</code> : new	22
with <code>\mfirstucMakeUppercase</code>	37	<code>\@gls@setupsort@standard</code> : moved	
<code>\glsnavigation</code> : switched to using		<code>\@gls@santizesort</code> to <code>\glsprestandardsort</code>	11
<code>\@gls@getgrouptitle</code>	263	ucmark: added check for memoir	10
<code>\ifglshasdesc</code> : replaced <code>\ifdefempty</code>		see: added <code>\gls@checkseeallowed</code>	62
with <code>\ifcempty</code>	53	<code>\glossarysection</code> : changed <code>\glossarymark</code>	
<code>\ifglshaslong</code> : new	53	to <code>\gls glossarymark</code>	37
<code>\ifglshasshort</code> : new	54	<code>\glossarystyle</code> : fixed bug caused by us-	
<code>\ifglshassymbol</code> : replaced <code>\ifdefempty</code>		ing <code>\ifdef</code> instead of <code>\ifcsdef</code>	207
with <code>\ifcempty</code>	53	<code>\gls@assign@desc@field</code> : changed to	
<code>\ifglused</code> : replaced <code>\ifthenelse</code> with		use <code>\glssetnoexpandfield</code>	18
<code>\ifbool</code>	51		
<code>\longnewglossaryentry</code> : new	76		

<code>\gls@assign@descplural@field:</code> changed to use <code>\glssetnoexpandfield</code> 18	<code>super3colheader:</code> switched to <code>\tabularnewline</code> 294
<code>\gls@assign@name@field:</code> changed to use <code>\glssetnoexpandfield</code> 18	<code>super4col:switched to \tabularnewline</code> 295
<code>\gls@assign@type@field:</code> changed to use <code>\glssetexpandfield</code> 18	<code>super4colheader:</code> switched to <code>\tabularnewline</code> 295
<code>\gls@checkseeallowed:</code> new 62	<code>super4colheaderborder:</code> switched to <code>\tabularnewline</code> 296
<code>\glsaddallunused:</code> set default to <code>\@glo@types</code> 154	<code>superheader:switched to \tabularnewline</code> 292
<code>\Glsentryfull:</code> changed to use <code>\acrfullformat</code> 151	<code>superheaderborder:</code> switched to <code>\tabularnewline</code> 292
<code>\glsentryfull:</code> changed to use <code>\acrfullformat</code> 151	3.14a (2013-11-12)
<code>\Glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 151	<code>\@glswritefiles:</code> renamed <code>\glswritefiles</code> to <code>\@glswritefiles</code> and used “savewrites” option to set <code>\glswritefiles</code> 173
<code>\glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 151	General: new 254
<code>\gls glossarymark:</code> renamed <code>\glossarymark</code> to <code>\gls glossarymark</code> to avoid con- flict with memoir 37	acronyms: new 14
<code>\glsprestandardsort:</code> new 10	<code>\gls@def glossaryentry:</code> added check for existence of default glossary 78
<code>\glssetexpandfield:</code> new 17	set the default for firstplural to be the value of plural 80
<code>\glssetnoexpandfield:</code> new 18	<code>xindygloss:</code> new 26
<code>altsuper4colheader:</code> switched to <code>\tabularnewline</code> 296	<code>\longprovideglossaryentry:</code> new ... 77
<code>altsuper4colheaderborder:</code> switched to <code>\tabularnewline</code> 297	<code>compatible-2.07:</code> added check for 2.07 before setting 3.07 compatibility 27
<code>long:</code> switched to <code>\tabularnewline</code> 269, 270	<code>notranslate:</code> new 22
<code>long3col:</code> switched to <code>\tabularnewline</code> 271	<code>\provideglossaryentry:</code> new 67
<code>long3colheader:</code> switched to <code>\tabularnewline</code> 272	4.0 (2013-11-14)
<code>long3colheaderborder:</code> switched to <code>\tabularnewline</code> 272	<code>\gls@def glossaryentry:</code> added check for first key 80
<code>long4col:</code> switched to <code>\tabularnewline</code> 272	<code>super:</code> fixed typo in <code>\subglossentry</code> (<code>\glossentrydesc</code>) 291
<code>long4colheader:</code> switched to <code>\tabularnewline</code> 273	4.01 (2013-11-16)
<code>longheader:</code> switched to <code>\tabularnewline</code> 270	General: fixed non-value options so that they can be passed to document class . 7
<code>longheaderborder:</code> switched to <code>\tabularnewline</code> 270	<code>\CustomAcronymFields:</code> inserted miss- ing comma 246
<code>\SetFootnoteAcronymDisplayStyle:</code> fixed missing argument bug 238	4.02 (2013-12-05)
<code>super:</code> switched to <code>\tabularnewline</code> . 291	<code>\@acrfull:</code> now using <code>\acrfullfmt</code> .. 212
<code>super3col:</code> switched to <code>\tabularnewline</code> 293	<code>\@gls@indexdef:</code> new 28
	<code>\@gls@numbersdef:</code> new 27
	<code>\@gls@symbolsdef:</code> new 27
	General: Removed <code>\acronymfont</code> . 141–144
	<code>\ACRfullfmt:</code> new 214
	<code>\Acrfullfmt:</code> new 213
	<code>\acrfullfmt:</code> new 213
	<code>\ACRfullplfmt:</code> new 215

<code>\Acrfullplfmt: new</code>	214	<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	
<code>\acrfullplfmt: new</code>	214	Moved check for empty custom text to	
<code>\acronymentry: new</code>	217	prevent unwanted parenthetical ma-	
<code>sanitize: fixed bug that caused an error</code>		terial	232
here	21	<code>\SetFootnoteAcronymDisplayStyle:</code>	
<code>sc-short-long: new</code>	221	Moved check for empty custom text to	
<code>sc-short-long-desc: new</code>	223	prevent unwanted parenthetical ma-	
<code>\Genacrfullformat: new</code>	102	terial	238
<code>\genacrfullformat: new</code>	102	<code>\SetGenericNewAcronym: new</code>	216
<code>\GenericAcronymFields: new</code>	217	<code>\SetSmallAcronymDisplayStyle:</code>	
<code>\Genplacrfullformat: new</code>	102	Moved check for empty custom text to	
<code>\genplacrfullformat: new</code>	102	prevent unwanted parenthetical ma-	
<code>\Glsentryfull: bug fix: added missing</code>		terial	241
<code>\acronymfont</code>	151	<code>dua: new</code>	223
<code>\glsentryfull: bug fix: added missing</code>		<code>dua-desc: new</code>	225
<code>\acronymfont</code>	151	<code>numberedsection: added nameref op-</code>	
<code>\Glsentryfullpl: bug fix: added miss-</code>		tion	7
ing <code>\acronymfont</code>	151	4.02 (2013-13-05)	
<code>\glsentryfullpl: bug fix: added miss-</code>		<code>\makeglossaries: made preamble only</code>	169
ing <code>\acronymfont</code>	151	4.03 (2014-01-17)	
<code>\glsgenacfmt: new</code>	100	General: changed default to <code>\@empty</code> in-	
<code>\GlsUseAcrEntryDispStyle: new</code> ...	218	stead of <code>\relax</code>	27
<code>\GlsUseAcrStyleDefs: new</code>	218	4.03 (2014-01-20)	
<code>short-long: new</code>	220	<code>\@do@wrglossary: added \glsdetoklabel</code>	
<code>short-long-desc: new</code>	222	178
<code>xindynoglsnumbers: new</code>	26	<code>\@ACRlong: removed \glslabel (defined</code>	
<code>sm-short-long: new</code>	221	in <code>\@gls@link</code>)	353
<code>sm-short-long-desc: new</code>	223	<code>\@ACRshort: removed \glslabel (de-</code>	
<code>index: new</code>	28	fined in <code>\@gls@link</code>)	351
<code>\newacronymstyle: new</code>	218	<code>\@Acrlong: removed \glslabel (defined</code>	
<code>long-sc-short: new</code>	220	in <code>\@gls@link</code>)	352
<code>long-sc-short-desc: new</code>	222	<code>\@Acrshort: removed \glslabel (de-</code>	
<code>long-short: new</code>	218	fined in <code>\@gls@link</code>)	351
<code>long-short-desc: new</code>	221	<code>\@GLS@: removed \glslabel (defined in</code>	
<code>long-sm-short: new</code>	221	<code>\@gls@link</code>)	120
<code>long-sm-short-desc: new</code>	222	<code>\@GLSpl: removed \glslabel (defined in</code>	
<code>long-sp-short-desc: new</code>	221	<code>\@gls@link</code>)	123
<code>footnote: new</code>	225	<code>\@Gls@: removed \glslabel (defined in</code>	
<code>footnote-desc: new</code>	227	<code>\@gls@link</code>)	120
<code>footnote-sc: new</code>	227	<code>\@Gls@entry@field: new</code>	145
<code>footnote-sc-desc: new</code>	228	<code>\@Glspl@: removed \glslabel (defined</code>	
<code>footnote-sm: new</code>	227	in <code>\@gls@link</code>)	122
<code>footnote-sm-desc: new</code>	228	<code>\@acrlong: removed \glslabel (defined</code>	
<code>\setacronymstyle: new</code>	217	in <code>\@gls@link</code>)	352
<code>\SetDescriptionAcronymDisplayStyle:</code>		<code>\@acrshort: removed \glslabel (de-</code>	
Moved check for empty custom text to		fined in <code>\@gls@link</code>)	351
prevent unwanted parenthetical ma-		<code>\@gls@: removed \glslabel (defined in</code>	
terial	236	<code>\@gls@link</code>)	119
		<code>\@gls@access@display: new</code>	339

<code>\@gls@entry@field:new</code>	144	<code>\glsentrydescpluralaccess:switched</code>	
<code>\@gls@fetchfield:new</code>	69	to using <code>\@gls@entry@field</code>	338
<code>\@gls@field@link:new</code>	124	<code>\glsentryfirstaccess:switched to us-</code>	
<code>\@gls@link:added \glsdetoklabel</code> .	107	ing <code>\@gls@entry@field</code>	338
moved <code>\@gls@link@opts</code> and		<code>\glsentryfirstplural: added</code>	
<code>\@gls@link@label</code> to <code>\@gls@link</code>	107	<code>\glsdetoklabel</code>	148
<code>\@gls@writedef:added \glsdetoklabel</code>		<code>\glsentrylongaccess:switched to us-</code>	
.....	68	ing <code>\@gls@entry@field</code>	339
<code>\@glsdisp:removed \glslabel (defined</code>		<code>\glsentrylongpluralaccess:switched</code>	
in <code>\@gls@link</code>)	124	to using <code>\@gls@entry@field</code>	339
<code>\@glspl@:removed \glslabel (defined</code>		<code>\glsentrypluralaccess:switched to</code>	
in <code>\@gls@link</code>)	121	using <code>\@gls@entry@field</code>	338
<code>\@printglossary:added \glsdetoklabel</code>		<code>\glsentryshortaccess:switched to us-</code>	
.....	184	ing <code>\@gls@entry@field</code>	338
General: removed <code>\glslabel (defined in</code>		<code>\glsentryshortpluralaccess:switched</code>	
<code>\@gls@link</code>)	137	to using <code>\@gls@entry@field</code>	338
sc-short-long-desc: redefined to use		<code>\glsentrysymbolaccess:switched to</code>	
accessibility information	357	using <code>\@gls@entry@field</code>	338
<code>\compatibleglossentry: added</code>		<code>\glsentrysymbolpluralaccess:</code>	
<code>\glsdetoklabel</code>	333	switched to using <code>\@gls@entry@field</code>	
<code>\compatiblesubglossentry: added</code>		338
<code>\glsdetoklabel</code>	334	<code>\glsentrytextaccess:switched to us-</code>	
<code>\Genacrfullformat:redefined to use ac-</code>		ing <code>\@gls@entry@field</code>	337
cessibility information	350	<code>\glsngenacfmt:redefined to use acce-</code>	
<code>\genacrfullformat:redefined to use ac-</code>		sibility information	348
cessibility information	350	<code>\glsgenentryfmt:redefined to use ac-</code>	
<code>\Genplacrfullformat:redefined to use</code>		cessibility information	345
accessibility information	350	<code>\gls hyperlink:added \glsdetoklabel</code>	
<code>\genplacrfullformat:redefined to use</code>		153
accessibility information	350	<code>\glslocalreset:added \glsdetoklabel</code>	
<code>\glossentryname:added \glsdetoklabel</code>		85
.....	202	<code>\glslocalunset:added \glsdetoklabel</code>	
<code>\gls@defglossaryentry: added</code>		86
<code>\glsdetoklabel</code>	77	<code>\glsmoveentry:added \glsdetoklabel</code>	
replaced #1 with <code>\@glo@label</code>	78	83
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>		replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	
.....	79	83
<code>\glsadd:added \glsdetoklabel</code>	153	<code>\glsrefentry:added \glsdetoklabel</code>	200
<code>\glsaddkey: switched to using</code>		<code>\glsreset:added \glsdetoklabel</code> ...	85
<code>\@gls@field@link</code>	72	<code>\glsseelist: added \expandafter</code>	
<code>\glsdetoklabel:new</code>	50	commands	181
<code>\glsdisplaynumberlist: added</code>		<code>\glsstepentry:added \glsdetoklabel</code>	
<code>\glsdetoklabel</code>	152	200
<code>\glsdoifexistsorwarn:new</code>	51	<code>\glsstepsubentry:added \glsdetoklabel</code>	
<code>\glsentryaccess:switched to using</code>		200
<code>\@gls@entry@field</code>	337	<code>\glsunset:added \glsdetoklabel</code> ...	86
<code>\glsentrydescaccess:switched to us-</code>		short-long: commented spurious EOL	220
ing <code>\@gls@entry@field</code>	338	redefined to use accessibility informa-	
		tion	355

short-long-desc: redefined to use accessibility information	357	\showglodescaccess:added\glsdetoklabel	369
\ifglsgdescsuppressed: added		\showglodescplural:added\glsdetoklabel	250
\glsdetoklabel	53	\showglodescpluralaccess: added	
fixed typo	53	\glsdetoklabel	369
\ifglsgentryexists:added\glsdetoklabel	51	\showglofirst:added\glsdetoklabel	248
\ifglsgshaschildren:added\glsdetoklabel	52	\showglofirstaccess:added\glsdetoklabel	369
\ifglsgshasdesc:added\glsdetoklabel	53	\showglofirsttpl:added\glsdetoklabel	248
\ifglsgshasfield:new	54	\showglofirstpluralaccess: added	
\ifglsgshaslong:added\glsdetoklabel	53	\glsdetoklabel	369
\ifglsgshasparent:added\glsdetoklabel	53	\showgloflag:added\glsdetoklabel	251
\ifglsgshasshort:added\glsdetoklabel	54	\showgloindex:added\glsdetoklabel	251
\ifglsgshassymbol:added\glsdetoklabel	53	\showglolevel:added\glsdetoklabel	248
replaced\ifcsempywith\ifdefempty		\showglolevel:added\glsdetoklabel	248
and replaced\ifxwith\ifdefequal	53	\showglolevel:added\glsdetoklabel	248
\ifglsgused:added\glsdetoklabel	51	\showglolevel:added\glsdetoklabel	248
sm-short-long-desc: redefined to use accessibility information	357	\showglolevel:added\glsdetoklabel	248
long-sc-short-desc: redefined to use accessibility information	356	\showglolevel:added\glsdetoklabel	248
long-short: redefined to use accessibility information	354	\showglolevel:added\glsdetoklabel	248
long-short-desc: redefined to use accessibility information	356	\showglolevel:added\glsdetoklabel	248
long-sm-short-desc: redefined to use accessibility information	356	\showglolevel:added\glsdetoklabel	248
footnote: redefined to use accessibility information	360	\showglolevel:added\glsdetoklabel	248
footnote-desc: redefined to use accessibility information	362	\showglolevel:added\glsdetoklabel	248
footnote-sc: redefined to use accessibility information	362	\showglolevel:added\glsdetoklabel	248
footnote-sc-desc: redefined to use accessibility information	363	\showglolevel:added\glsdetoklabel	248
footnote-sm: redefined to use accessibility information	362	\showglolevel:added\glsdetoklabel	248
footnote-sm-desc: redefined to use accessibility information	363	\showglolevel:added\glsdetoklabel	248
\renewacronymstyle:new	218	\showglolevel:added\glsdetoklabel	248
\showglocounter:added\glsdetoklabel	249	\showglolevel:added\glsdetoklabel	248
\showglodesc:added\glsdetoklabel	250	\showglolevel:added\glsdetoklabel	248

<code>\showglotextaccess</code> : added <code>\glsdetoklabel</code>	<code>\@gls@noidx@do</code> : new	193
..... 369	<code>\@gls@nofref@warn</code> : new	173
<code>\showglotype</code> : added <code>\glsdetoklabel</code>	<code>\@gls@reference</code> : new	195
..... 248	<code>\@gls@warnonglossdefined</code> : new	17
<code>\showglouseri</code> : added <code>\glsdetoklabel</code>	<code>\@gls@warnontheglossdefined</code> : new	17
..... 249	<code>\@no@makeglossaries</code> : new	172
<code>\showglouserii</code> : added <code>\glsdetoklabel</code>	<code>\@print@glossary</code> : new	185
..... 249	<code>\@print@noidx@glossary</code> : new	191
<code>\showglouseriii</code> : added <code>\glsdetoklabel</code>	<code>\@print@gloss@setsort</code> : new	183
..... 249	<code>\@print@glossary</code> : new	183
<code>\showglouseriv</code> : added <code>\glsdetoklabel</code>	General: added sort key to <code>printgloss</code>	
..... 249	group	198
<code>\showglouserv</code> : added <code>\glsdetoklabel</code>	<code>\compatibleglossentry</code> : changed	
..... 249	<code>\newcommand</code> to <code>\def</code> as is may or may	
<code>\showglouservi</code> : added <code>\glsdetoklabel</code>	not be defined	333
..... 250	<code>\compatiblesubglossentry</code> : changed	
<code>dua</code> : fixed bug in <code>\acrfullfmt</code>	<code>\newcommand</code> to <code>\def</code> as is may or may	
..... 224	not be defined	334
fixed bug in <code>\Acrrfullplfmt</code>	<code>\defglsdisplayfirst</code> : fixed unwanted	
..... 225	space	103
fixed bug in <code>\acrfullplfmt</code>	<code>\glo@grabfirst</code> : new	192
..... 225	<code>\gls@defglossaryentry</code> : replaced <code>\ifx</code>	
redefined to use accessibility informa-	with <code>\ifdefvoid</code>	82
tion	<code>\glsnoidxdisplayloc</code> : new	195
..... 358	<code>\glsnoidxdisplayloc</code> : list handler:	
<code>dua-desc</code> : commented spurious EOL	new	194
.. 225	<code>\glsnoidxloclist</code> : new	194
redefined to use accessibility informa-	<code>\glsnoidxloclist</code> : handler: new	194
tion	<code>\glsnoidxstripaccents</code> : new	19
..... 360	<code>alltree</code> : moved <code>hangindent</code> and	
4.04 (2014-03-04)	<code>parindent</code> assignments outside level	
<code>\@gls@getcounterprefix</code> : added warn-	test	309
ing if no prefix can be formed	<code>\makeglossaries</code> : Moved definition of	
..... 179	<code>\glswrite</code> to <code>\makeglossaries</code>	167
4.04 (2014-03-06)	<code>\makenoidxglossaries</code> : new	169
<code>\@@gls@noidx@nosanitizesort</code> : new	<code>\printglossary</code> : changed to use new	
. 19	<code>\@printglossary</code>	182
<code>\@@gls@noidx@sanitizesort</code> : new	<code>\printnoidxglossaries</code> : new	183
... 19	<code>\printnoidxglossary</code> : new	183
<code>\@@gls@nosanitizesort</code> : new	<code>\showgloclist</code> : new	251
..... 19	<code>\warn@noprintglossary</code> : Activate warn-	
<code>\@gls@sanitizesort</code> : new	ing in <code>\makeglossaries</code>	182
..... 18	<code>\writeist</code> : checked for definition of	
<code>\@glo@addchildren</code> : new	<code>\glswrite</code>	156, 160
..... 187	4.06 (2014-03-12)	
<code>\@glo@do@sortentries</code> : new	<code>\@GLS@</code> : added <code>\glsifhyper</code>	121
..... 187	<code>\@GLSpl</code> : added <code>\glsifhyper</code>	123
<code>\@glo@grabfirst</code> : new	<code>\@Gls@</code> : added <code>\glsifhyper</code>	120
..... 192	<code>\@Glspl</code> : added <code>\glsifhyper</code>	122
<code>\@glo@sortedinsert</code> : new		
..... 188		
<code>\@glo@sortentries</code> : new		
..... 186		
<code>\@glo@sorthandler@case</code> : new		
..... 189		
<code>\@glo@sorthandler@letter</code> : new		
... 188		
<code>\@glo@sorthandler@nocase</code> : new		
... 189		
<code>\@glo@sorthandler@word</code> : new		
..... 188		
<code>\@glo@sortmacro@case</code> : new		
..... 190		
<code>\@glo@sortmacro@def</code> : new		
..... 191		
<code>\@glo@sortmacro@def@do</code> : new		
..... 191		
<code>\@glo@sortmacro@letter</code> : new		
..... 189		
<code>\@glo@sortmacro@nocase</code> : new		
..... 190		
<code>\@glo@sortmacro@standard</code> : new		
... 190		
<code>\@glo@sortmacro@use</code> : new		
..... 191		
<code>\@glo@sortmacro@word</code> : new		
..... 189		
<code>\@gls@getothergrouptitle</code> : new		
... 205		

\@gls@: added \glsifhyper	119	\@gls@field@link: added assignment of	
\@gls@numbersdef: added hook to set		\do@gls@link@checkfirsthyper	124
toc title	28	\@gls@forbidtextext: new	56
\@gls@symbolsdef: added hook to set		\@gls@hyp@opt: new	105
toc title	27	\@gls@link: removed redundancy	107
\@glsdisp: added \glsifhyper	124	renamed \gls@type to \glstype ...	107
\@glspl@: added \glsifhyper	121	\@glsdisp: moved \glsifhyper	124
General: added \glsifhyper	137–144	moved check for first use to	
acronym: added hook to set toc title	14	\@gls@link	124
acronyms: added hook to set toc title ...	14	\@glspl@: moved \glsifhyper	121
\glsdefmain: added hook to set toc title	13	moved check for first use to	
4.07 (2014-04-04)		\@gls@link	121
\@glossarysection: added optional argu-		\@ignored@glossaries: new	59
ment when using unstarred version	38	General: added entrycounter option to	
\@gls@noidx@do: added \global in case		printgloss family	196
it's used in a tabular-like style	193	added nopostdot option to printgloss	
\Acrfullplfmt: fixed no case change		family	196
bug	214	added subentrycounter option to	
\glsletentryfield: new	145	printgloss family	197
4.08 (2014-07-30)		explicitly initialise hyper key	104
\@ACRlong: added \do@gls@link@checkfirsthyper		removed \glsifhyper	137–144
.....	352	removed \@sACRlongpl	144
\@ACRshort: added \do@gls@link@checkfirsthyper		removed \@sAcrlongpl	143
.....	351	removed \@sacrlongpl	142
\@Acrlong: added \do@gls@link@checkfirsthyper		removed \@sACRlong	142
.....	352	removed \@sAcrlong	141
\@Acrshort: added \do@gls@link@checkfirsthyper		removed \@sacrlong	140
.....	351	removed \@sACRshortpl	140
\@GLS@: moved \glsifhyper	121	removed \@sAcrshortpl	139
moved check for first use to		removed \@sacrshortpl	139
\@gls@link	121	removed \@sACRshort	138
\@GLSpl: moved \glsifhyper	123	removed \@sAcrshort	137
moved check for first use to		removed \@sacrshort	137
\@gls@link	123	removed \@sgls@link	106
\@Gls@: moved \glsifhyper	120	removed \@sGLSdescplural	130
moved check for first use to		removed \@sGlsdescplural	130
\@gls@link	120	removed \@sglsdescplural	130
\@Glspl@: moved \glsifhyper	122	removed \@sGLSdesc	129
moved check for first use to		removed \@sGlsdesc	129
\@gls@link	122	removed \@sglsdesc	129
\@acrlong: added \do@gls@link@checkfirsthyper		removed \@sglsdisp	124
.....	352	removed \@sGLSfirstplural	128
\@acrshort: added \do@gls@link@checkfirsthyper		removed \@sGlsfirstplural	127
.....	350	removed \@sglsfirstplural	127
\@closegls: new	166	removed \@sGLSfirst	126
\@gls@: moved \glsifhyper	119	removed \@sGlsfirst	126
moved check for first use to		removed \@sglsfirst	125
\@gls@link	119	removed \@sGLSname	128
\@gls@doautomake: new	26	removed \@sGlsname	128

removed \@sglsname	128	\acrfull: removed \@sacrfull	212
removed \@sGLSplural	127	switched to using \@gls@hyp@opt ..	212
removed \@sGlsplural	127	\ACRfullpl: removed \@sACRfullpl ..	215
removed \@sglsplural	126	switched to using \@gls@hyp@opt ..	215
removed \@sGLSpl	123	\Acrfullpl: removed \@sAcrfullpl ..	214
removed \@sGlspl	122	switched to using \@gls@hyp@opt ..	214
removed \@sglspl	121	\acrfullpl: removed \@sacrfullpl ..	214
removed \@sGLSsymbolplural	132	switched to using \@gls@hyp@opt ..	214
removed \@sGlsymbolplural	131	\ACRlong: switched to using \@gls@hyp@opt	
removed \@sglsymbolplural	131	142
removed \@sGLSsymbol	131	\Acrlong: switched to using \@gls@hyp@opt	
removed \@sGlsymbol	131	141
removed \@sglsymbol	130	\acrlong: switched to using \@gls@hyp@opt	
removed \@sGLStext	125	140
removed \@sGlstext	125	\ACRlongpl: switched to using	
removed \@sglstext	125	\@gls@hyp@opt	144
removed \@sGLSuseriii	134	\Acrlongpl: switched to using	
removed \@sGlsuseriii	134	\@gls@hyp@opt	143
removed \@sglsuseriii	134	\acrlongpl: switched to using	
removed \@sGLSuserii	133	\@gls@hyp@opt	142
removed \@sGlsuserii	133	\ACRshort: switched to using	
removed \@sglsuserii	133	\@gls@hyp@opt	138
removed \@sGLSuseriv	135	\Acrshort: switched to using	
removed \@sGlsuseriv	135	\@gls@hyp@opt	137
removed \@sglsuseriv	134	\acrshort: switched to using	
removed \@sGLSuseri	133	\@gls@hyp@opt	137
removed \@sGlsuseri	132	\ACRshortpl: switched to using	
removed \@sglsuseri	132	\@gls@hyp@opt	140
removed \@sGLSuservi	137	\Acrshortpl: switched to using	
removed \@sGlsuservi	136	\@gls@hyp@opt	139
removed \@sglsuservi	136	\acrshortpl: switched to using	
removed \@sGLSuserv	136	\@gls@hyp@opt	139
removed \@sGlsuserv	136	\forallacronyms: new	49
removed \@sglsuserv	135	\GLS: switched to using \@gls@hyp@opt	120
removed \@sGLS	120	\Gls: switched to using \@gls@hyp@opt	119
removed \@sGls	119	\gls: switched to using \@gls@hyp@opt	118
removed \@sgls	119	\gls@defglossaryentry: added check	
removed \@thirdofthree (defined in		for ignored glossary	79
kernel)	118	\gls@istfilebase: new	34
removed sPGLS	259	\glsaddkey: removed \@sGLS@user@<key>	
removed sPglS	257	73
removed spglS	256	removed \@sGls@user@<key>	72
removed sPGLSpl	259	removed \@sgls@user@<key>	72
removed sPglSpl	258	switched to using \@gls@hyp@opt	72, 73
removed spglSpl	257	\GLSdesc: switched to using \@gls@hyp@opt	
\ACRfull: removed \@sACRfull	213	129
switched to using \@gls@hyp@opt ..	213	\Glsdesc: switched to using \@gls@hyp@opt	
\Acrfull: removed \@sAcrfull	213	129
switched to using \@gls@hyp@opt ..	213		

\glsdesc: switched to using \@gls@hyp@opt	129	\glsplural: switched to using \@gls@hyp@opt	126
\GLSdescplural: switched to using \@gls@hyp@opt	130	\glsspace: new	213
\Glsdescplural: switched to using \@gls@hyp@opt	130	\GLSsymbol: switched to using \@gls@hyp@opt	131
\glsdescplural: switched to using \@gls@hyp@opt	129	\Glsymbol: switched to using \@gls@hyp@opt	131
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	118	\glssymbol: switched to using \@gls@hyp@opt	130
\glsdisp: switched to using \@gls@hyp@opt	123	\GLSsymbolplural: switched to using \@gls@hyp@opt	132
\glsdohyperlink: new	117	\Glsymbolplural: switched to using \@gls@hyp@opt	131
\glsdohypertarget: new	117	\glssymbolplural: switched to using \@gls@hyp@opt	131
\glsenablehyper: added \KV@glslink@hypertrue to definition	118	\GLStext: switched to using \@gls@hyp@opt	125
\GLSfirst: switched to using \@gls@hyp@opt	126	\Glstext: switched to using \@gls@hyp@opt	125
\Glsfirst: switched to using \@gls@hyp@opt	126	\glstext: switched to using \@gls@hyp@opt	125
\glsfirst: switched to using \@gls@hyp@opt	125	\glstreenamfmt: new	303
\GLSfirstplural: switched to using \@gls@hyp@opt	128	\GLSuseri: switched to using \@gls@hyp@opt	132
\Glsfirstplural: switched to using \@gls@hyp@opt	127	\Glsuseri: switched to using \@gls@hyp@opt	132
\glsfirstplural: switched to using \@gls@hyp@opt	127	\glsuseri: switched to using \@gls@hyp@opt	132
\glsifhyper: deprecated	105	\GLSuserii: switched to using \@gls@hyp@opt	133
\glslink: switched to using \@gls@hyp@opt	106	\Glsuserii: switched to using \@gls@hyp@opt	133
\glslinkcheckfirsthyperhook: new	107	\glsuserii: switched to using \@gls@hyp@opt	133
\glslinkvar: new	105	\GLSuseriii: switched to using \@gls@hyp@opt	134
\GLSname: switched to using \@gls@hyp@opt	128	\Glsuseriii: switched to using \@gls@hyp@opt	134
\Glsname: switched to using \@gls@hyp@opt	128	\glsuseriii: switched to using \@gls@hyp@opt	134
\glsname: switched to using \@gls@hyp@opt	128	\GLSuseriv: switched to using \@gls@hyp@opt	135
\GLSpl: switched to using \@gls@hyp@opt	123	\Glsuseriv: switched to using \@gls@hyp@opt	135
\Glspl: switched to using \@gls@hyp@opt	122	\glsuseriv: switched to using \@gls@hyp@opt	134
\glspl: switched to using \@gls@hyp@opt	121	\GLSuserv: switched to using \@gls@hyp@opt	136
\GLSplural: switched to using \@gls@hyp@opt	127		
\Glsplural: switched to using \@gls@hyp@opt	126		

\Glsuserv: switched to using	4.12 (2014-11-22)
\@gls@hyp@opt	135
\glsuserv: switched to using	
\@gls@hyp@opt	135
\GLSuservi: switched to using	
\@gls@hyp@opt	137
\Glsuservi: switched to using	
\@gls@hyp@opt	136
\glsuservi: switched to using	
\@gls@hyp@opt	136
\ifignoredglossary: new	59
altlongragged4col: fixed bug that displayed description instead of symbol	283
\newglossary: added starred version	57
\newignoredglossary: new	59
\ns@newglossary: added \@glotype@<name>@log	57
new	57
\p@gls@hyp@opt: new	105
\PGLS: changed to use \@gls@hyp@opt	259
\PglS: changed to use \@gls@hyp@opt	257
\pglS: changed to use \@gls@hyp@opt	256
\PGLSpl: changed to use \@gls@hyp@opt	259
\PglSpl: changed to use \@gls@hyp@opt	258
\pglSpl: changed to use \@gls@hyp@opt	257
\s@gls@hyp@opt: new	105
\s@newglossary: new	57
automake: new	26
4.09 (2014-08-12)	
\glsaddkey: fixed bug in user commands	72
4.10 (2014-08-27)	
\@Gls@acentryname: new	146
\@Gls@entryname: new	145
\@gls@glossary: Renamed \@glossary to \@gls@glossary	175
\glspercentchar: new	155
\glstildechar: new	155
almtree: moved space after symbol	310, 311
4.11 (2014-09-01)	
\@do@wrglossary: added hook	178
sanitize: none option	21
\gls@wrglossary: renamed from \@wrglossary to \@gls@wrglossary	175
\glsaddprotectedpagefmt: new	176
\glsbackslash: new	155
\@gls@addpredefinedattributes: Added glsignore attribute	43
\@gls@adjustmode: new	154
\@gls@notranslatorhook: removed	22
\@gls@toc: added \protect to \numberline	40
\@gls@usetranslator: new	22
\glsacrpluralsuffix: new	31
\glsadd: added check for vertical mode	153
\glsaddallunused: replaced @gobble with glsignore	154
\glsifusedtranslator: new	22
\glsignore: new	154
\glsupacrpluralsuffix: new	31
\ProvidesGlossariesLang: new	31
\RequireGlossariesLang: new	31
4.13 (2015-02-03)	
\indexspace: new	265, 285, 303
4.14 (2015-02-28)	
\@@glslocalreset: new	87
\@@glslocalunset: new	86
\@@glsreset: new	87
\@@glsunset: new	86
\@@newglossaryentry@defcounters: new	88
\cGls: new	91
\cGls@: new	91
\cGlspl@: new	92
\cglS: new	91
\cglS@: new	91
\cglSpl: new	92
\cglSpl@: new	92
\@gls@entry@count: new	91
\@gls@increment@currcount: new	90
\@gls@local@increment@currcount: new	90
\@gls@write@entrycounts: new	90
\@glslocalreset: new	87
\@glslocalunset: new	86
\@glsreset: new	87
\@glsunset: new	86
\@newglossaryentry@defcounters: new	83
\cGls: new	91
\cglS: new	91
\cGlsformat: new	92
\cglSformat: new	91
\cGlspl: new	92

<code>\cglspl:new</code>	92	<code>\glswriteentry:new</code>	176
<code>\cGlsplformat:new</code>	93	<code>\ifglsfieldcseq:new</code>	76
<code>\cglsplformat:new</code>	92	<code>\ifglsfielddefeq:new</code>	75
<code>\gls@defdocnewglossaryentry:new</code> .	67	<code>\ifglsfieldeq:new</code>	75
<code>\glsenableentrycount:new</code>	88	<code>long-sp-short:new</code>	219
<code>\glslocalreset:switched to \@glslocalreset</code>	85	<code>\showglofield:new</code>	252
<code>\glslocalunset:switched to \@glslocalunset</code>	86	4.18 (2015-09-09)	
<code>\glsreset:switched to \@glsreset</code> ...	85	General: split mfirstuc into separate bundle	4
<code>\glsunset:switched to \@glsunset</code> ...	86	4.19 (2015-10-31)	
4.15 (2015-03-16)		<code>\glstreenamibox:new</code>	309
General: bug fix replaced <code>\glo@type</code> with <code>\glstype</code>	144	4.19 (2015-11-22)	
4.16 (2015-06-18)		<code>\@gls@link@nocheckfirsthyper:new</code>	124
<code>\glsaddstoragekey:new</code>	70	<code>\@gls@preglossaryhook:new</code>	183
4.16 (2015-07-08)		<code>\@printglossary:added \@gls@preglossaryhook</code>	184
<code>\@ACRlong:added \glspostlinkhook</code>	353	<code>\do@glsdisablehyperinlist:new</code> ..	107
<code>\@ACRshort:added \glspostlinkhook</code>	352	<code>\doifglossarynoexistsordo:new</code> ...	52
<code>\@Acrlong:added \glspostlinkhook</code>	352	<code>\gls@gobbleopt:new</code>	56
<code>\@Acrshort:added \glspostlinkhook</code>	351	<code>\glsdoifexistsordo:new</code>	52
<code>\@GLS@:added \glspostlinkhook</code> ...	121	4.20 (2015-11-30)	
<code>\@GLSpl:added \glspostlinkhook</code> ..	123	<code>\@gls@link:added \@gls@setdefault@glslink@opts</code>	107
<code>\@Gls@:added \glspostlinkhook</code> ...	120	added <code>\glsdonohyperlink</code> when hyperlink is suppressed	108
<code>\@Glspl@:added \glspostlinkhook</code> .	123	<code>\@gls@setdefault@glslink@opts:</code> new	107
<code>\@acrlong:added \glspostlinkhook</code>	352	<code>\gls@checkseeallowed@preambleonly:</code> new	62
<code>\@acrshort:added \glspostlinkhook</code>	351	<code>\glsdonohyperlink:new</code>	117
<code>\@gls@:added \glspostlinkhook</code> ...	119	4.21 (2016-01-24)	
<code>\@gls@@link:added \glspostlinkhook</code>	106	<code>\@printglossary:warn if no style has been set</code>	183
<code>\@gls@field@link:added \glspostlinkhook</code>	124	General: changed <code>checkfirsthyper</code> assignment	137–144
<code>\@gls@link: moved definition of \glsifhyperon</code> outside of this macro	108	<code>\glossarystyle: set default style if not already set</code>	207
<code>\@glsdisp:added \glspostlinkhook</code>	124	<code>\glsLTpenaltycheck:new</code>	278
<code>\@glspl@:added \glspostlinkhook</code> .	122	<code>\glspatchLToutput:new</code>	279
General: added <code>\glspostlinkhook</code>	137–144	<code>\glspenaltygroupskip:new</code>	278
<code>\glsacspace:new</code>	220	<code>altlong4col-booktabs:new</code>	277
<code>\glsadd: changed \@do@wrglossary to \@do@wrglossary</code>	154	<code>altlongragged4col-booktabs:new</code> .	278
<code>\glsfielddef:new</code>	74	<code>long-booktabs:new</code>	275
<code>\glsfieldedef:new</code>	74	<code>long3col-booktabs:new</code>	276
<code>\glsfieldfetch:new</code>	75	<code>long4col-booktabs:new</code>	276
<code>\glsfieldgdef:new</code>	74	<code>longragged-booktabs:new</code>	277
<code>\glsfieldxdef:new</code>	73	<code>longragged3col-booktabs:new</code> ...	278
<code>\glsifhyperon: moved definition of \glsifhyperon</code>	107	<code>\setglossarystyle: set default style if not already set</code>	206
<code>\glslinkpostsetkeys:new</code>	107		
<code>\glspostlinkhook:new</code>	106		

4.22 (2016-04-19)	
\@do@wrglossary: added check for	
\@arabic	177
added test to allow temporary primitive	
modifications and added arabic case	177
mcolaltrreespannav: new	290
mcolindexspannav: new	286
mcoltreenonamespannav: new	289
mcoltreespannav: new	287
\gls@arabicpage: new	176
\gls@protected@pagefmts: added ara-	
bic to list	176
\glsentrytitlecase: new	148
\glsfindwidesttoplevelname: new .	308
\glslistgroupheaderfmt: new	266
\glslistnavigationitem: new	266
\glstreegroupheaderfmt: new	303
\glstreenavigationfmt: new	304
\ifglswrallowprimitivemods: new .	177
list: fixed missing space before descrip-	
tion	266
long: fixed typo in \glossentrydesc .	270
super4col: fixed bug in \glossentry .	295
4.23 (2016-04-30)	
\glscurrentfieldvalue: new	55
\ifglshasfield: added \glscurrentfieldvalue	
.....	54, 55
altlongragged4col: check for nogroup-	
skip changed	284
altsuperragged4col: check for	
nogroupskip changed	302
long: check for nogroupskip changed ..	270
long-booktabs: check for nogroupskip	
changed	275
long3col: check for nogroupskip	
changed	271
long3col-booktabs: check for nogroup-	
skip changed	276
long4col: check for nogroupskip	
changed	273
long4col-booktabs: check for nogroup-	
skip changed	277
longragged: check for nogroupskip	
changed	280
longragged3col: check for nogroupskip	
changed	282
super: check for nogroupskip changed .	292
super3col: check for nogroupskip	
changed	293
super4col: check for nogroupskip	
changed	295
superragged: check for nogroupskip	
changed	298
superragged3col: check for nogroup-	
skip changed	300
4.24 (2016-05-27)	
\@gls@extramakeindexopts: new ...	164
\@gls@glossary: added check for debug	
mode	175
\@gls@see@noindex: new	5
debug: new	5
seenindex: new	6
\glsnomakeindexwarning: new	40
\GlsSetQuote: new	162
\GlsSetWriteIstHook: new	162
4.25 (2016-06-09)	
\@gls@enablesavenonumberlist: new	63
\@gls@initnonumberlist: new	63
\@gls@savenonumberlist: new	63

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	
\!	114
\"	19, 111–114, 116
\#	158
\%	155, 160, 161, 317, 318
\&	31, 153
\'	19
\.	9, 19
\=	19
\?	111, 113, 163
\@@delimN	209
\@do@wrglossary	170, 178
\@do@wrglossary	154, 175
\@glo@assign@sortkey	170
\@glo@list	49, 50
\@glo@sort	19
\@glo@type	183
\@glossarysec	6, 38, 39
\@glossaryseclabel	7, 39, 196
\@glossarysecstar	7, 38, 39, 196
\@gls@checkactual	115, 116
\@gls@checkbar	114, 115
\@gls@checkescactual	113
\@gls@checkescbar	113, 114
\@gls@checkesclevel	114
\@gls@checkescquote	112, 164
\@gls@checklevel	115
\@gls@checkquote	112, 162, 163
\@gls@default@entryfmt	94, 103
\@gls@expand@field	17, 66, 70, 71, 231, 233, 235, 237, 239, 242, 244, 364–368
\@gls@extramakeindexopts	162, 167
\@gls@fixbraces	180
\@gls@noexpand@field	18, 65, 66
\@gls@noidx@no@sanitizesort	19
\@gls@noidx@nosanitizesort	172
\@gls@nosanitizesort	18, 172
\@@gls@sanitizesort	18, 172
\@@gls@xdycheckbackslash	116, 117
\@@gls@xdycheckquote	116
\@@gls@localreset	87, 89
\@@gls@localunset	86, 89
\@@gls@reset	87, 89
\@@gls@unset	86, 89
\@@newglossaryentry@defcounters	88
\@@this@glo@	50
\@ACRfull	213
\@ACRfullpl	215
\@ACRlong	142, 214
\@ACRlongpl	144, 215
\@ACRshort	138, 214
\@ACRshortpl	140, 215
\@Acrfull	213
\@Acrfullpl	214
\@Acrlong	141, 213
\@Acrlongpl	143, 214
\@Acrshort	137, 138
\@Acrshortpl	139
\@Alph	176–178
\@GLS	120
\@GLS@	120, 259
\@GLSdesc	129
\@GLSdesc@	129
\@GLSdescplural	130
\@GLSdescplural@	130
\@GLSfirst	126
\@GLSfirst@	126
\@GLSfirstplural	128
\@GLSfirstplural@	128
\@GLSname	128
\@GLSname@	128, 129
\@GLSpl	123
\@GLSpl@	123, 260
\@GLSplural	127

<code>\@GLSplural@</code>	127	<code>\@Glsuseriii@</code>	134
<code>\@GLSsymbol</code>	131	<code>\@Glsuseriv</code>	135
<code>\@GLSsymbol@</code>	131	<code>\@Glsuseriv@</code>	135
<code>\@GLSsymbolplural</code>	132	<code>\@Glsuseriv</code>	135, 136
<code>\@GLSsymbolplural@</code>	132	<code>\@Glsuseriv@</code>	136
<code>\@GLStext</code>	125	<code>\@Glsuserivi</code>	136
<code>\@GLStext@</code>	125	<code>\@Glsuserivi@</code>	136
<code>\@GLSuseri</code>	132, 133	<code>\@Mi</code>	279
<code>\@GLSuseri@</code>	133	<code>\@PGLS</code>	259
<code>\@GLSuserii</code>	133	<code>\@PGLS@</code>	259
<code>\@GLSuserii@</code>	133	<code>\@PGLSpl</code>	259
<code>\@GLSuseriii</code>	134	<code>\@PGLSpl@</code>	259
<code>\@GLSuseriii@</code>	134	<code>\@PglS</code>	257
<code>\@GLSuseriv</code>	135	<code>\@PglS@</code>	257
<code>\@GLSuseriv@</code>	135	<code>\@PglSpl</code>	258
<code>\@GLSuseriv</code>	136	<code>\@PglSpl@</code>	258
<code>\@GLSuseriv@</code>	136	<code>\@Roman</code>	176–178
<code>\@GLSuserivi</code>	137	<code>\@acrfull</code>	212
<code>\@GLSuserivi@</code>	137	<code>\@acrfullpl</code>	214
<code>\@Gls</code>	119	<code>\@acrlong</code>	140, 141, 213
<code>\@Gls@</code>	89, 91, 119, 258	<code>\@acrlongpl</code>	142, 143, 214
<code>\@Gls@acentryname</code>	216	<code>\@acrshort</code>	137, 213
<code>\@Gls@entry@field</code>	72, 146–151	<code>\@acrshortpl</code>	139, 214
<code>\@Gls@entryname</code>	145, 216	<code>\@addtoacronymlists</code>	15
<code>\@Glsdesc</code>	129	<code>\@after</code>	15
<code>\@Glsdesc@</code>	129	<code>\@afterheading</code>	267, 321
<code>\@Glsdescplural</code>	130	<code>\@alph</code>	176–178
<code>\@Glsdescplural@</code>	130	<code>\@arabic</code>	176–178
<code>\@Glsfirst</code>	126	<code>\@auxout</code>	56, 57, 90, 167, 170, 173, 182, 185, 186, 261
<code>\@Glsfirst@</code>	126	<code>\@backslashchar</code>	110, 116, 117
<code>\@Glsfirstplural</code>	127	<code>\@before</code>	15
<code>\@Glsfirstplural@</code>	127	<code>\@bsphack</code>	175
<code>\@Glsname</code>	128	<code>\@cGls</code>	91
<code>\@Glsname@</code>	128	<code>\@cGls@</code>	89, 91
<code>\@Glspl</code>	122	<code>\@cGlspl</code>	92
<code>\@Glspl@</code>	90, 92, 122, 258, 259	<code>\@cGlspl@</code>	90, 92
<code>\@Glsplural</code>	126, 127	<code>\@cclv</code>	279
<code>\@Glsplural@</code>	127	<code>\@cgls</code>	91
<code>\@Glsymbol</code>	131	<code>\@cgls@</code>	89, 91
<code>\@Glsymbol@</code>	131	<code>\@cglspl</code>	92
<code>\@Glsymbolplural</code>	131	<code>\@cglspl@</code>	89, 92
<code>\@Glsymbolplural@</code>	131, 132	<code>\@chapter</code>	29
<code>\@Glstext</code>	125	<code>\@classoptionslist</code>	28
<code>\@Glstext@</code>	125	<code>\@colht</code>	279
<code>\@Glsuseri</code>	132	<code>\@colroom</code>	279
<code>\@Glsuseri@</code>	132	<code>\@currentlabelname</code>	7, 196
<code>\@Glsuserii</code>	133	<code>\@curroptions</code>	28
<code>\@Glsuserii@</code>	133	<code>\@declaredoptions</code>	28
<code>\@Glsuseriii</code>	134		

<code>\@delimN</code>	209	<code>\@glo@desc</code>	60, 76, 77, 79, 81
<code>\@delimR</code>	208	<code>\@glo@descaccess</code>	335–337
<code>\@disable@onlypremakeg</code>	168	<code>\@glo@descplural</code>	60, 76, 77
<code>\@disable@premakecs</code>	30	<code>\@glo@descpluralaccess</code>	335–337
<code>\@disabled@gl\$addxdycounters</code>	43	<code>\@glo@do@sortentries</code>	186
<code>\@do@addcounter</code>	41	<code>\@glo@entry</code>	154
<code>\@do@auxoutstuff</code>	185, 186	<code>\@glo@entryprefix</code>	254
<code>\@do@glossentry</code>	202, 333	<code>\@glo@entryprefixfirst</code>	254
<code>\@do@gl\$@getcounterprefix</code>	178	<code>\@glo@entryprefixfirstplural</code>	254, 255
<code>\@do@gl\$@islistofacronyms</code>	15	<code>\@glo@entryprefixplural</code>	254
<code>\@do@gl\$see</code>	82	<code>\@glo@esclabel</code>	84, 85
<code>\@do@ifinlist</code>	41	<code>\@glo@etext</code>	94–96
<code>\@do@newglossaryentry</code>		<code>\@glo@first</code>	61, 77, 80, 81, 242, 368
	216, 230–237, 239, 241–244, 246, 364–368	<code>\@glo@firstaccess</code>	334, 336, 337
<code>\@do@seeglossary</code>	170, 181	<code>\@glo@firstplural</code>	61, 77, 80, 81, 368
<code>\@do@subglossentry</code>	203, 334	<code>\@glo@firstpluralaccess</code>	335–337
<code>\@do@wrglossary</code>	108	<code>\@glo@grabfirst</code>	192
<code>\@do@writeaux@info</code>	182	<code>\@glo@label</code>	64, 71–82,
<code>\@ehc</code>	279		88, 152, 153, 254, 255, 308, 309, 336, 337
<code>\@empty</code>	12, 15, 26–28, 30, 41, 42, 46,	<code>\@glo@list</code>	82
	48, 49, 79, 84, 109, 119–123, 137–144,	<code>\@glo@long</code>	53, 64, 78, 81
	156, 159, 161, 166, 167, 174, 175, 179,	<code>\@glo@longaccess</code>	335–337
	180, 197, 199, 206, 231, 233, 235–239,	<code>\@glo@longpl</code>	64,
	241, 242, 244, 246, 315, 317, 319, 351–353		78, 81, 230, 233, 234, 237, 239, 242–244, 364
<code>\@end@fixbraces</code>	180	<code>\@glo@longpluralaccess</code>	335–337
<code>\@end@fortrue</code>	23, 52, 70, 261	<code>\@glo@name</code>	11, 60, 65, 77, 80, 81
<code>\@esphack</code>	175	<code>\@glo@no@assign@sortkey</code>	169
<code>\@expandtwoargs</code>	28	<code>\@glo@nonumberlist</code>	63
<code>\@firstofone</code>	19, 20	<code>\@glo@numfmt</code>	179, 315
<code>\@firstofthree</code>	105, 118,	<code>\@glo@parent</code>	12, 62, 78–80, 84, 85, 187, 188
	119, 121, 124, 137, 139, 141, 143, 351–353	<code>\@glo@plural</code>	61, 77, 80, 366
<code>\@firstoftwo</code>	22,	<code>\@glo@pluralaccess</code>	334, 336, 337
	23, 69, 70, 105, 121–123, 139, 140, 143, 144	<code>\@glo@prefix</code>	
<code>\@for</code>	23, 28, 30,		8, 62, 78, 84, 85, 110, 178, 179, 314, 315
	41, 42, 49, 50, 68, 70, 110, 156–158, 168,	<code>\@glo@range</code>	178, 179, 314, 315
	169, 176, 181, 186, 187, 217, 231, 234,	<code>\@glo@see</code>	62, 78, 82
	235, 238, 240, 243, 245, 247, 261, 262, 315	<code>\@glo@seeautonumberlist</code>	8, 62
<code>\@glo@@desc</code>	81	<code>\@glo@short</code>	54, 64, 78, 81, 367
<code>\@glo@@symbol</code>	82	<code>\@glo@shortaccess</code>	335–337, 364–367
<code>\@glo@access</code>	334, 336, 337, 339	<code>\@glo@shortpl</code>	64, 78, 81,
<code>\@glo@addchildren</code>	187, 191		230, 232–234, 237, 239, 242, 244, 364, 367
<code>\@glo@assign@sortkey</code>	169, 170, 198	<code>\@glo@shortpluralaccess</code>	335–337
<code>\@glo@check@mkidxrangechar</code>		<code>\@glo@sort</code>	11, 18, 19, 60, 78, 80, 84, 85
	109, 110, 178, 314, 315	<code>\@glo@sortedinsert</code>	187, 188
<code>\@glo@childlist</code>	187	<code>\@glo@sortentries</code>	189, 190
<code>\@glo@counter</code>	62, 78, 81	<code>\@glo@sorthandler@case</code>	190
<code>\@glo@counterprefix</code>	173, 178–180, 206, 209	<code>\@glo@sorthandler@letter</code>	189
<code>\@glo@default@sorttype</code>	10, 170, 189, 190	<code>\@glo@sorthandler@nocase</code>	190
<code>\@glo@defaultcounter</code>	81	<code>\@glo@sorthandler@word</code>	189

<code>\@glo@sortinghandler</code>	186, 188	<code>\@gls@checkactual</code>	111, 163
<code>\@glo@sortinglist</code>	186, 188, 191	<code>\@gls@checkbar</code>	111, 163
<code>\@glo@sorttype</code>	170, 191–193, 198	<code>\@gls@checkedmkidx</code>	110–117, 162–164
<code>\@glo@storeentry</code>	11, 13	<code>\@gls@checkescactual</code>	111, 163
<code>\@glo@sufffix</code>	110, 179, 315	<code>\@gls@checkescbar</code>	111, 163
<code>\@glo@symbol</code>	53, 61, 77, 82, 236, 241, 336	<code>\@gls@checkescquote</code>	111, 163, 164
<code>\@glo@symbolaccess</code>	335–337, 367	<code>\@gls@checklevel</code>	111, 163
<code>\@glo@symbolplural</code>	61, 77, 82	<code>\@gls@checkmkidxchars</code>	84, 110, 163, 169, 178, 180, 314, 315
<code>\@glo@symbolpluralaccess</code>	335–337	<code>\@gls@checkquote</code>	111, 162, 163
<code>\@glo@text</code>	60, 77, 80, 82, 119–124, 145, 146, 237, 255, 366	<code>\@gls@classI</code>	157
<code>\@glo@textaccess</code>	334, 336, 337, 364–367	<code>\@gls@classII</code>	157
<code>\@glo@thislabel</code>	83	<code>\@gls@codepage</code>	186
<code>\@glo@thislettergrp</code>	192, 193	<code>\@gls@counter</code>	104, 107–109, 153, 154, 173, 179, 180, 315
<code>\@glo@thisvalue</code>	54, 55	<code>\@gls@counterwithin</code>	10, 197, 199
<code>\@glo@tmp</code>	71, 72, 179	<code>\@gls@ctr</code>	41
<code>\@glo@type</code>	7, 12, 61, 77–79, 81, 82, 153, 154, 168, 173, 174, 183–186, 188, 191, 192, 195, 196, 216, 231, 233, 235, 237, 239, 240, 242, 244, 246, 261, 262	<code>\@gls@currentlettergroup</code>	192, 193
<code>\@glo@types</code>	49, 50, 57, 87, 88, 154, 168, 169, 252, 308	<code>\@gls@declareoption</code>	8, 9, 13, 14, 17, 22, 25–28
<code>\@glo@useri</code>	63, 78, 81	<code>\@gls@default</code>	93
<code>\@glo@userii</code>	63, 78, 81	<code>\@gls@default@value</code>	53–55, 65, 66, 77, 78, 80, 81, 240, 254
<code>\@glo@useriii</code>	63, 78, 81	<code>\@gls@deffile</code>	68, 69
<code>\@glo@useriv</code>	64, 78, 81	<code>\@gls@defsort</code>	11, 12, 82
<code>\@glo@userv</code>	64, 78, 81	<code>\@gls@defsortcount</code>	11, 12, 58
<code>\@glo@uservi</code>	64, 78, 81	<code>\@gls@do@acronymsdef</code>	14, 29, 59
<code>\@glodesc</code>	81	<code>\@gls@do@indexdef</code>	28, 29, 59
<code>\@glolist@</code>	79	<code>\@gls@do@numbersdef</code>	27, 29, 59
<code>\@gloname</code>	81	<code>\@gls@do@symbolsdef</code>	27, 59
<code>\@glossary@default@style</code>	7, 9, 183, 206, 207, 247	<code>\@gls@do@symbolssdef</code>	29
<code>\@glossaryentryfield</code>	84	<code>\@gls@doautomake</code>	26, 169
<code>\@glossarysection</code>	37	<code>\@gls@docheckquotedef</code>	162–164
<code>\@glossarystyle</code>	183, 184, 196	<code>\@gls@docloadedfalse</code>	4
<code>\@glossarysubentryfield</code>	84, 85	<code>\@gls@docloadedtrue</code>	4
<code>\@gls</code>	118	<code>\@gls@dodedeflistparser</code>	168
<code>\@gls@</code>	89, 91, 119, 257, 258	<code>\@gls@doentrydef</code>	103
<code>\@gls@@link</code>	106	<code>\@gls@dolast</code>	181
<code>\@gls@Hcounter</code>	108, 109	<code>\@gls@donext</code>	181
<code>\@gls@ReturnAfterFi</code>	210	<code>\@gls@donext@def</code>	152
<code>\@gls@access@display</code>	339, 340	<code>\@gls@dothiswrite</code>	166, 167
<code>\@gls@actualchar</code>	84, 85, 113, 115, 160, 318	<code>\@gls@elem</code>	261
<code>\@gls@addpredefinedattributes</code>	156, 165	<code>\@gls@enablesavenonumberlist</code>	68
<code>\@gls@adjustmode</code>	153	<code>\@gls@encapchar</code>	113, 114, 160, 179, 180, 315, 318
<code>\@gls@automake</code>	166, 169	<code>\@gls@entry@count</code>	90
<code>\@gls@between</code>	262, 263	<code>\@gls@entry@field</code>	71, 72, 88, 89, 145–152, 337–339
<code>\@gls@body</code>	146	<code>\@gls@escbsdq</code>	111, 161, 319

<code>\@gls@expand@fields</code>	66, 67	<code>\@gls@noidx@do</code>	192
<code>\@gls@expandonce</code>	67	<code>\@gls@noidx@getgrouptitle</code>	170, 206
<code>\@gls@extramakeindexopts</code>	167	<code>\@gls@noidx@sanitizesort</code>	19, 172
<code>\@gls@fetchfield</code>	55	<code>\@gls@noidx@setsanitizesort</code>	21, 172
<code>\@gls@field@link</code>	72, 73, 125–137	<code>\@gls@noidx@loclist@finalsep</code>	171
<code>\@gls@firsttok</code>	192	<code>\@gls@noidx@loclist@prev</code>	171, 194, 195
<code>\@gls@fixbraces</code>	82	<code>\@gls@noidx@loclist@sep</code>	171, 194
<code>\@gls@forbidtexext</code>	57	<code>\@gls@noref@warn</code>	169, 192
<code>\@gls@get@counterprefix</code>	179	<code>\@gls@numberlink</code>	209
<code>\@gls@getbody</code>	146	<code>\@gls@numbersdef</code>	27
<code>\@gls@getcounterprefix</code>	178	<code>\@gls@numlist@lastsep</code>	152, 153
<code>\@gls@getgrouptitle</code>	170, 205, 263	<code>\@gls@numlist@nextsep</code>	152
<code>\@gls@glossary</code>	174, 175	<code>\@gls@numlist@sep</code>	152, 153
<code>\@gls@gobbleopt</code>	56	<code>\@gls@old@chapter</code>	29
<code>\@gls@grptitle</code>	205, 261, 263	<code>\@gls@oldnewglossaryentryposthook</code>	336
<code>\@gls@hyp@opt</code>	72, 73, 91, 92, 106, 118–123, 125–144, 212–215, 256–259	<code>\@gls@oldnewglossaryentryprehook</code>	336
<code>\@gls@hyp@opt@cs</code>	105	<code>\@gls@onlypremakeg</code>	30
<code>\@gls@hypergroup</code>	261	<code>\@gls@order</code>	166, 167
<code>\@gls@ifinlist</code>	41	<code>\@gls@org@LT@output</code>	278
<code>\@gls@ifnotmeasuring</code>	85	<code>\@gls@org@glsnoidx@displayloc</code>	172
<code>\@gls@igtype</code>	60	<code>\@gls@org@glsseeformat</code>	172
<code>\@gls@increment@currcount</code>	89	<code>\@gls@preglossaryhook</code>	184
<code>\@gls@indexdef</code>	28	<code>\@gls@prevlevel</code>	289, 290, 309–312, 327, 328
<code>\@gls@initnonumberlist</code>	63, 78	<code>\@gls@provide@newglossary</code>	57
<code>\@gls@islistofacronyms</code>	15	<code>\@gls@quotechar</code>	112–115, 160, 162, 164, 318
<code>\@gls@keylist</code>	363	<code>\@gls@reference</code>	170, 173
<code>\@gls@keymap</code>	63, 68, 70, 71, 254, 336	<code>\@gls@removespaces</code>	209, 210
<code>\@gls@label</code>	170, 173, 178, 179	<code>\@gls@renewglossary</code>	165
<code>\@gls@langmod</code>	166	<code>\@gls@replacementtext</code>	339
<code>\@gls@levelchar</code>	85, 114, 115, 160, 318	<code>\@gls@rest</code>	146
<code>\@gls@link</code>	106, 119–124, 137–144, 351–353	<code>\@gls@roman</code>	44, 315, 316
<code>\@gls@link@checkfirsthyper</code>	106, 119–124	<code>\@gls@sanitized@tmp</code>	110, 111
<code>\@gls@link@label</code>	107, 232, 238	<code>\@gls@sanitizedesc</code>	24
<code>\@gls@link@nocheckfirsthyper</code>	124, 137–144	<code>\@gls@sanitizesort</code>	11
<code>\@gls@link@opts</code>	107, 232, 238	<code>\@gls@sanitizesymbol</code>	24
<code>\@gls@list</code>	261, 262	<code>\@gls@saveentrycounter</code>	108, 154
<code>\@gls@listsuffix</code>	41	<code>\@gls@savenonumberlist</code>	62, 63
<code>\@gls@loadlist</code>	9, 247	<code>\@gls@see@noindex</code>	6, 62
<code>\@gls@loadlong</code>	8, 9, 247	<code>\@gls@setacrstyle</code>	24, 29
<code>\@gls@loadsuper</code>	9, 247	<code>\@gls@setcounter</code>	58
<code>\@gls@loadtree</code>	9, 247	<code>\@gls@setdefault@glslink@opts</code>	107
<code>\@gls@local@increment@currcount</code>	89	<code>\@gls@setsort</code>	11, 12, 108
<code>\@gls@loclist</code>	171, 172, 193, 194	<code>\@gls@setupshortcuts</code>	29
<code>\@gls@map</code>	68–70	<code>\@gls@sort</code>	193
<code>\@gls@missingnumberlist</code>	81	<code>\@gls@sort@A</code>	188, 189
<code>\@gls@noaccess</code>	339	<code>\@gls@sort@B</code>	188, 189
<code>\@gls@noexpand@fields</code>	67	<code>\@gls@startswithexpandonce</code>	66
<code>\@gls@nohyperlist</code>	16, 59, 107	<code>\@gls@storenonumberlist</code>	63, 81
		<code>\@gls@symbolsdef</code>	27

<code>\@gls@this</code>	176	<code>\@glslocalreset</code>	86, 89
<code>\@gls@thisHloc</code>	179	<code>\@glslocalunset</code>	86, 89
<code>\@gls@thisfield</code>	55	<code>\@glslocref</code>	173, 178, 179, 314, 315
<code>\@gls@thislabel</code>	52, 181, 191	<code>\@glsminrange</code>	156, 157, 316
<code>\@gls@thislist</code>	152, 153	<code>\@glsname</code>	128
<code>\@gls@thisloc</code>	179	<code>\@glsname@</code>	128
<code>\@gls@thisval</code>	70	<code>\@glsnextpages</code>	184
<code>\@gls@title</code>	37	<code>\@glsnodesc</code>	77, 79, 81
<code>\@gls@tmp</code>	12, 32, 46, 67, 110, 111, 175, 262, 263	<code>\@glsnoname</code>	77, 80, 81
<code>\@gls@tmpb</code>	112–117, 162, 164	<code>\@glsnonextpages</code>	184
<code>\@gls@toc</code>	39	<code>\@glsnumberformat</code>	104, 107, 153, 173, 178, 179, 314, 315
<code>\@gls@type</code>	169, 217, 231, 234, 235, 238, 240, 243, 245, 247, 308	<code>\@glsopenfile</code>	165, 174
<code>\@gls@updatechecked</code>	110, 111, 163	<code>\@glsorder</code>	167
<code>\@gls@usetranslator</code>	22, 23, 32	<code>\@glspl</code>	121
<code>\@gls@value</code>	65, 66, 148	<code>\@glspl@</code>	89, 92, 121, 257–259
<code>\@gls@warnonglossdefined</code>	17, 182	<code>\@glsplural</code>	126
<code>\@gls@warnontheGLOSSdefined</code>	17, 201	<code>\@glsplural@</code>	126
<code>\@gls@write@entrycounts</code>	90	<code>\@glsreset</code>	85, 89
<code>\@gls@writedef</code>	68	<code>\@glssee</code>	82, 181
<code>\@gls@writeisthook</code>	160–162	<code>\@glsymbol</code>	130
<code>\@gls@xdy@locationlist</code>	157	<code>\@glsymbol@</code>	130
<code>\@gls@xdycheckbackslash</code>	110	<code>\@glsymbolplural</code>	131
<code>\@gls@xdycheckquote</code>	110	<code>\@glsymbolplural@</code>	131
<code>\@gls@xref</code>	180	<code>\@glstarget</code>	118, 202, 261
<code>\@glsAlpha compositor</code>	35, 45, 316	<code>\@glstext</code>	125
<code>\@glsHlocref</code>	178	<code>\@glstext@</code>	125
<code>\@glsacronymlists</code>	15, 16, 49, 216, 217, 231, 233–235, 237–240, 242–247, 252	<code>\@glsunset</code>	86, 89
<code>\@glsaddkey</code>	71	<code>\@glsuseri</code>	132
<code>\@glsaddstoragekey</code>	70	<code>\@glsuseri@</code>	132
<code>\@glsaddxdyattribute</code>	42	<code>\@glsuserii</code>	133
<code>\@glsdefaultsort</code>	11	<code>\@glsuserii@</code>	133
<code>\@glsdesc</code>	129	<code>\@glsuseriii</code>	134
<code>\@glsdesc@</code>	129	<code>\@glsuseriii@</code>	134
<code>\@glsdescplural</code>	129, 130	<code>\@glsuseriv</code>	134
<code>\@glsdescplural@</code>	130	<code>\@glsuseriv@</code>	134, 135
<code>\@glsdisp</code>	123	<code>\@glsuserv</code>	135
<code>\@glsentry</code>	87, 88, 90	<code>\@glsuserv@</code>	135
<code>\@glsentrytitlecase</code>	148, 149	<code>\@glsuservi</code>	136
<code>\@glsfirst</code>	125	<code>\@glsuservi@</code>	136
<code>\@glsfirst@</code>	125, 126	<code>\@glswidestname</code>	309, 310, 327
<code>\@glsfirstletter</code>	49, 155	<code>\@glswritefiles</code>	26
<code>\@glsfirstplural</code>	127	<code>\@gobble</code>	12, 68, 69, 110, 155, 158, 169, 313, 317, 318
<code>\@glsfirstplural@</code>	127	<code>\@idxitem</code>	286, 304
<code>\@glsnumber</code>	208	<code>\@ifclassloaded</code>	4, 10, 38
<code>\@glsisacronymlistfalse</code>	15	<code>\@ifnextchar</code>	58, 105
<code>\@glsisacronymlisttrue</code>	15	<code>\@ifpackageloaded</code>	4, 22, 23, 32, 48, 85, 152, 162, 333
<code>\@glslink</code>	108, 118, 153, 261		

\@ifstar	57, 70, 71, 105, 211	\@secondofthree	105, 118, 120, 122, 138, 139, 141, 143, 351
\@ifundefined	31, 262, 269, 280, 291, 298, 310, 327, 341	\@secondoftwo	22, 23, 32, 68, 70, 118–120, 124, 137, 138, 141, 142, 351–353, 371
\@ignored@glossaries	59, 60	\@set@glo@numformat	179, 315
\@input@	185	\@sglsaddkey	71
\@istfilename	167	\@sglsaddstoragekey	70
\@makecol	279	\@thirdofthree	105, 120, 123, 138, 140, 142, 144, 351
\@makeglossary	168	\@this@attr	158
\@minus	265, 285, 303	\@this@childlabel	187
\@mkboth	38	\@this@counter	42
\@newglossary	56, 57	\@this@ctr	158
\@newglossaryentry@defcounters	82, 88	\@this@key	70
\@newglossaryentryposthook	71, 72, 83, 254, 336	\@this@label	186, 187
\@newglossaryentryprehook	71, 76, 78, 254, 336	\@thiscs	30
\@nil	15, 82, 109–111, 146, 163, 178, 180, 192, 193, 208–210, 314, 315	\@tmp	44, 316
\@nnil	15, 181	\@use@option	28
\@no@makeglossaries	168, 170	\@warn@nomakeglossaries	167, 186
\@no@post@desc	320	\@wrglossary@pageformat	177
\@nopostdesc	184	\@wrglossarynumberhook	177, 178
\@onelevel@sanitize	18, 19, 44, 68, 84, 110, 111, 159, 180, 182, 192, 316, 317	\@xdy@main@language	25, 166, 185
\@onlypreamble	58, 68, 77, 90, 93, 169, 172	\@xdy@attributelist	42, 158
\@onlypremakeg	34, 35, 41, 43, 46, 58, 162	\@xdy@attributes	42, 156, 313, 315
\@org@glossaryentrynumbers	183, 185	\@xdy@counters	41, 42, 158
\@org@gls@assign@descplural	231, 239, 242, 244, 364, 367, 368	\@xdy@language	185
\@org@gls@assign@firstpl	230, 231, 233, 235, 237, 239, 242, 244, 364–368	\@xdy@lettergroups	49, 160, 318
\@org@gls@assign@plural	231, 233, 235, 237, 239, 242, 244, 364–368	\@xdy@locationclassorder	47, 158, 317
\@org@gls@assign@symbolplural	231, 233, 235, 237, 242, 244, 365, 366, 368	\@xdy@locref	42, 159, 313, 317
\@org@gls@numberformat	152	\@xdy@requiredstyles	47, 156, 315
\@org@newglossaryentryprehook	76	\@xdy@sortrules	47, 160, 318
\@outputpage	279	\@xdystyle	156, 315
\@p@glossarysection	37	\@xdy@useralphabets	44, 157, 315
\@pgls	256	\@xdy@userlocationdefs	46, 157, 314, 316
\@pgls@	256	\@xfor@nextelement	181
\@pglspl	257	\	83, 110, 155, 160, 161, 208, 209, 318, 319, 321–323, 331, 332
\@pglspl@	257	\{	68, 69, 155, 161, 313, 318, 319
\@plus	265, 285, 303	\}	69, 155, 161, 313, 319
\@print@glossary	182	\^	19
\@print@noidx@glossary	183	\‘	19
\@printgloss@setsort	169, 170, 184	\	111, 113, 163
\@printglossary	182, 183	\~	19
\@roman	44, 315		
		A	
		\AA	20
		\aa	20
		accsupp package	333

<code>\accsuppglossaryentryfield</code>	333	<code>\appto</code>	16, 63, 70–72, 254, 336
<code>\accsuppglossarysubentryfield</code>	334	array package	275, 279, 297
<code>\acrfootnote</code>	232, 238	article class	179
<code>\Acrfull</code>	229	<code>\AtBeginDocument</code>	14, 48, 68, 85, 154, 170
<code>\acrfull</code>	229	<code>\AtEndDocument</code>	26, 68, 90, 169, 173, 185, 186, 262
<code>\ACRfullfmt</code>	214, 216, 225, 226, 359, 361	B	
<code>\Acrfullfmt</code>	213, 216, 224, 226, 359, 361	<code>\b</code>	20
<code>\acrfullfmt</code>	212, 216, 224, 226, 359, 361	babel package	22, 30, 32, 47
<code>\acrfullformat</code>	151, 213, 230, 246	<code>\begin</code>	106, 158, 166, 192, 266, 269–275, 277, 278, 280–303, 317
<code>\Acrfullpl</code>	229	<code>\BeginAccSupp</code>	339
<code>\acrfullpl</code>	229	<code>\begingroup</code>	5, 175, 177, 209
<code>\ACRfullplfmt</code>	215, 217, 225, 227, 359, 361	<code>\bfseries</code>	270, 272– 277, 281, 283–285, 292–297, 299, 301–303
<code>\Acrfullplfmt</code>	214, 216, 225, 226, 359, 361	<code>\bgroup</code>	19, 76, 152, 184, 187
<code>\acrfullplfmt</code>	214, 216, 225, 226, 359, 361	booktabs package	275–278
<code>\acrlinkfootnote</code>	231	<code>\boolean</code>	245
<code>\acrlinkfullformat</code>	213–215	<code>\boolfalse</code>	27
<code>\Acrlong</code>	229	<code>\booltrue</code>	27
<code>\acrlong</code>	229	<code>\bottomrule</code>	275–277
<code>\Acrlongpl</code>	229	<code>\box</code>	279
<code>\acrlongpl</code>	229	C	
<code>\acrnameformat</code>	237, 366	<code>\c</code>	19
<code>\acronymentry</code>	216, 219–223, 225–228, 355–357, 360–363	<code>\c@equation</code>	109
<code>\acronymfont</code>	100, 137–140, 146, 151, 215, 217, 219– 228, 232, 234, 236, 238–241, 243, 348, 349, 351–353, 355–357, 359–363, 365–367	<code>\c@glossarysubentry</code>	197
<code>\acronymname</code>	14, 33	<code>\c@page</code>	176–178
<code>\acronymsort</code>	216, 219– 223, 225, 226, 228, 355–357, 360, 361, 363	<code>\c@Gls</code>	91
<code>\acronymtype</code>	14, 216, 217, 230–237, 239, 241–244, 246, 364–367	<code>\c@Gls</code>	91
<code>\acrpluralsuffix</code>	216, 219–221, 225–227, 230, 231, 233, 234, 237–240, 242–244, 246, 355, 356, 360–362, 364–368	<code>\c@Glsformat</code>	89
<code>\Acrshort</code>	228	<code>\c@Glsformat</code>	89
<code>\acrshort</code>	228	<code>\c@Glspl</code>	92
<code>\Acrshortpl</code>	229	<code>\c@Glspl</code>	92
<code>\acrshortpl</code>	228	<code>\c@Glsplformat</code>	90
<code>\addcontentsline</code>	40	<code>\c@Glsplformat</code>	89
<code>\addglossarytocaptions</code>	32	<code>\changes</code>	106, 166, 266
<code>\addtolength</code>	310, 311, 327	<code>\char</code>	206
<code>\advance</code>	12, 79, 109, 279	<code>\cleardoublepage</code>	39
<code>\AE</code>	20	<code>\clearpage</code>	39
<code>\ae</code>	20	<code>\closeout</code>	68, 160, 161, 166, 174
amsgen package	4, 104	<code>\compatglossarystyle</code>	320–332
amsmath package	85	<code>\compatibleglossentry</code>	203
<code>\andname</code>	181	<code>\compatiblesubglossentry</code>	203
<code>\AnyTrackedLanguages</code>	32, 371	<code>\copy</code>	279
		<code>\count@</code>	192, 193
		<code>\cs</code>	106
		<code>\csdef</code>	17, 18, 70–73, 82, 83, 88, 89, 186, 187, 208, 218, 319

<code>\csedef</code>	90, 177	<code>\define@choickey</code>	6, 7, 10, 21, 22, 25, 62, 196, 197		
<code>\csgdef</code>	37, 56, 59, 89, 90, 182, 195	<code>\define@key</code>	7, 10, 16, 21, 25, 26, 60–64, 70, 71, 104, 153, 195, 196, 198, 254, 334, 335		
<code>\cslet</code>	63, 76, 77, 83, 191	<code>\DefineAcronymSynonyms</code>	29, 230		
<code>\csname</code>	10–12, 28, 30, 32, 33, 38, 39, 42, 44, 45, 48, 49, 52, 57–59, 65, 66, 70–74, 76, 79–87, 103, 107–110, 119–124, 137–145, 152, 154, 157, 158, 163–166, 170, 173–175, 177, 179, 180, 183–185, 187, 196, 202, 203, 206, 207, 211, 247–255, 261, 262, 309, 310, 313–315, 327, 333, 334, 336–338, 341, 351–353, 369, 370	<code>\delimN</code>	159, 168, 194, 209, 317		
<code>\csshow</code>	252	<code>\delimR</code>	159, 208, 209, 317		
<code>\csuse</code>	33, 36, 56, 65, 66, 72, 73, 103, 166, 167, 187, 190, 191, 193, 195–197, 207, 218, 255, 320–332	<code>\DescriptionDUANewAcronymDef</code>	235		
<code>\csxdef</code>	81, 90	<code>\DescriptionFootnoteNewAcronymDef</code>	233		
<code>\currentglossary</code>	36, 184, 197, 199	<code>\descriptionname</code>	33, 270, 272–276, 281, 283–285, 292–297, 299, 301–303		
<code>\currentglssubentry</code>	197–200	<code>\DescriptionNewAcronymDef</code>	237		
<code>\CurrentOption</code>	28, 254, 333	<code>\dimen@</code>	220, 279, 308, 309		
<code>\CurrentTrackedLanguage</code>	33, 371, 372	<code>\disable@keys</code>	28		
<code>\CurrentTrackedTag</code>	33, 371, 372	<code>\do</code>	23, 28, 30, 41, 42, 49, 50, 68, 70, 110, 152, 156–158, 168, 169, 176, 181, 186, 187, 217, 231, 234, 235, 238, 240, 243, 245, 247, 261, 262, 315		
<code>\CustomAcronymFields</code>	246	<code>\do@glo@storeentry</code>	11, 12, 82		
<code>\CustomNewAcronymDef</code>	247	<code>\do@gls@link@checkfirsthyper</code>	106, 108, 119–124, 137–144, 351, 352		
D				<code>\do@gls@xdycheckbackslash</code>	110
<code>\d</code>	19	<code>\do@gls@xdydisablehyperinlist</code>	108		
<code>datatool</code> package	188	<code>\do@glshaschildren</code>	52		
<code>\day</code>	156, 160, 315, 318	<code>doc</code> package	4, 5, 13		
<code>\DeclareAcronymList</code>	14, 16, 216, 217, 231, 233, 235, 237, 240, 242, 244, 246	<code>\doifglossarynoexistsordo</code>	57		
<code>\DeclareListParser</code>	168	<code>\dtl@ifsingle</code>	205		
<code>\DeclareOption</code>	8, 254, 333	<code>\dtl@insertinto</code>	188		
<code>\DeclareOptionX</code>	8	<code>\dtl@sortresult</code>	188, 189		
<code>\DeclareRobustCommand</code>	33, 181, 240, 339–341	<code>\dtlcompare</code>	189		
<code>\def</code>	8, 11, 12, 15, 19, 20, 25, 29, 30, 33, 34, 37, 41, 43, 45–49, 52, 56–58, 60–64, 67, 74, 76, 78–83, 89–93, 103, 104, 107–117, 119–144, 152, 153, 156, 160, 162–164, 166, 168, 170, 171, 173, 177–180, 183, 186, 191, 192, 194–199, 205–216, 231, 233, 235–237, 239, 241, 242, 244, 246, 254, 256–260, 262–265, 289, 290, 309–312, 314, 315, 318, 320, 327, 328, 333–336, 350–353, 364–368	<code>\dtlicompare</code>	189		
<code>\def@gls@xdycheckbackslash</code>	116, 117	<code>\DTLifinlist</code>	60, 107		
<code>\DefaultNewAcronymDef</code>	231	<code>\DTLifint</code>	206		
<code>\defglstryfmt</code>	58, 59, 103, 217, 230, 232, 234, 236, 238, 241, 243, 246	<code>\dtlletterindexcompare</code>	188		
<code>\define@boolkey</code>	5, 6, 8–10, 13, 20, 21, 24–27, 104, 198	<code>\DTLsubstituteall</code>	111		
		<code>\dtlwordindexcompare</code>	188		
		<code>\DUANewAcronymDef</code>	245		
		E			
		<code>\eappto</code>	59, 83, 177		
		<code>\edef</code>	12, 15, 30, 32, 40–42, 44, 46, 47, 49, 52, 57, 59, 60, 65, 66, 70, 73–78, 83, 84, 103, 107–117, 152, 154, 155, 160, 162–164, 166–168, 170, 174, 178, 179, 182, 185–189, 193, 197, 200, 206, 209, 211, 230, 232, 234, 236, 239, 241, 243, 261, 313, 314, 316, 318, 363–367		
		<code>\egroup</code>	19, 77, 153, 185, 187		
		<code>\else</code>	5, 9, 12–15, 17, 18, 20, 21, 26, 28–30, 34, 35, 38–44, 46–49, 62,		

65, 79, 80, 83–85, 89, 90, 107–117, 119–124, 146, 155, 156, 159–166, 174–181, 184, 193, 197–201, 206, 208–210, 220, 234, 235, 238, 240, 243, 245, 247, 262, 266, 270, 271, 273, 276, 277, 279, 280, 282, 284, 292, 293, 295, 298, 300, 302, 305, 306, 308–311, 314–320, 325–328, 339	233–235, 237, 238, 240, 243–247, 253, 262, 266, 270, 271, 273, 276–279, 281, 282, 284, 292, 293, 295, 299, 300, 302, 305–311, 313–317, 319, 320, 325–328, 339
<code>\emph</code>	181, 210
<code>\empty</code>	209
<code>\end</code>	106, 159, 166, 192, 266, 269–275, 277, 278, 280–303, 317
<code>\end@doifinlist</code>	41
<code>\end@getprefix</code>	179
<code>\end@gl@islistofacronyms</code>	15
<code>\EndAccSupp</code>	339
<code>\endcsname</code>	10–12, 28, 30, 32, 33, 38, 39, 42, 44, 45, 48, 49, 52, 57–59, 65, 66, 70–74, 76, 79–87, 103, 107–110, 119–124, 137–145, 152, 154, 157, 158, 163–166, 170, 173–175, 177, 179, 180, 183–187, 196, 202, 203, 206, 207, 211, 247–255, 261, 262, 309, 310, 313–315, 327, 333, 334, 336–338, 341, 351–353, 369, 370
<code>\endfoot</code>	270–277, 281–285
<code>\endgroup</code>	5, 175, 178, 209
<code>\endhead</code>	270–277, 281–285
<code>\endtheglossary</code>	5
<code>\entryname</code>	33, 270, 272–276, 281, 283–285, 292–297, 299, 301–303
<code>\equal</code>	21, 29, 39, 108, 168, 206, 261
equation (counter)	108, 109
etoolbox package	4
<code>\expandafter</code>	11, 12, 19, 28, 30, 32, 42, 44, 45, 47–50, 52, 57, 58, 60, 65, 66, 68–74, 76, 79, 80, 82, 84–87, 103, 107–116, 146, 153, 155, 158, 162–165, 174–176, 178, 179, 181, 184, 187, 188, 192, 193, 202, 203, 207, 209, 211, 232, 238, 247–255, 261, 262, 309, 313–315, 317, 318, 333, 334, 336, 337, 339, 363, 369, 370
<code>\expandonce</code>	65–67, 110, 111, 163, 164, 177, 188, 189, 202, 203, 216, 230, 232–234, 237, 239, 242–244, 333, 334
F	
<code>\fi</code>	5–7, 9, 11–15, 17, 18, 20, 21, 23, 26, 28–30, 33–35, 38–49, 58, 62, 65, 79–85, 89, 90, 106–117, 119–124, 146, 154–156, 159, 161–165, 167–169, 174–184, 186, 193, 196–201, 206–210, 220, 230, 231,
file types	
<code>.aux</code>	185
<code>.glo</code>	84
<code>.ist</code>	155, 165
<code>.toc</code>	40
<code>.xdy</code>	34
<code>glo</code>	253
<code>\firstacronymfont</code> ...	102, 218–220, 226, 232, 236, 238, 241, 350, 354–356, 360, 361
<code>\footnote</code>	226, 232, 361
<code>\FootnoteNewAcronymDef</code>	240
<code>\forallglossaries</code>	50, 173, 183, 308
<code>\forallgl@entries</code>	87, 88, 90, 154
<code>\ForEachTrackedDialect</code>	33, 371, 372
<code>\forgl@entries</code>	50, 52, 83, 191, 308
<code>\forlist@csloop</code>	186, 192
<code>\forlistloop</code>	171, 172, 194
G	
garamondx package	212
<code>\gdef</code> ..	12, 42, 57, 74, 79, 80, 175, 198, 199, 262
<code>\Genacr@fullformat</code>	101, 216–220, 226, 350, 354, 355, 361
<code>\genacr@fullformat</code>	101, 102, 216–220, 226, 349, 350, 354, 355, 360
<code>\GenericAcronymFields</code>	216, 218–228, 354–357, 359, 360, 363
<code>\Genplacr@fullformat</code>	101, 217, 219, 220, 226, 349, 355, 361
<code>\genplacr@fullformat</code>	101, 102, 216, 217, 219, 220, 226, 349, 355, 361
<code>\glo@desc</code>	320
<code>\glo@do@compare</code>	188, 189
<code>\glo@grabfirst</code>	193
<code>\glo@label</code>	52, 83
<code>\glo@list</code>	83
<code>\glo@name</code>	202
<code>\glo@parent</code>	52
<code>\glo@type</code>	83
<code>\glo@value</code>	68, 69
<code>\global</code>	11, 12, 65, 68, 76, 77, 82, 86, 87, 175, 185, 193, 199, 279
<code>\glo@linkprefix</code>	108, 153, 202
<code>\gloskey</code>	106
glossentry (counter)	200

glossaries package
..... [28](#), [48](#), [156](#), [247](#), [254](#), [266](#), [313](#), [333](#)
glossaries-accsupp package [83](#), [333](#)
glossaries-extra package [333](#)
\GlossariesWarning
..... [5](#), [6](#), [17](#), [20](#), [21](#), [37](#), [40](#), [51](#), [55](#),
[62](#), [65](#), [91](#), [92](#), [103](#), [105](#), [152](#), [166](#), [167](#),
[169](#), [171–173](#), [175](#), [180](#), [183](#), [204](#), [207](#), [313](#)
\GlossariesWarningNoLine
..... [5](#), [17](#), [168](#), [170](#), [174](#), [186](#), [262](#)
\glossary [314](#), [315](#)
glossary package [1](#), [211](#)
glossary styles:
 altlist [267](#), [268](#), [321](#)
 altlistgroup [267](#), [268](#), [321](#)
 altlisthypergroup [268](#), [321](#)
 altlong4col [274](#), [283](#), [323](#)
 altlong4col-booktabs [277](#), [278](#)
 altlong4colborder [274](#), [323](#)
 altlong4colheader [274](#), [277](#), [323](#)
 altlong4colheaderborder ... [275](#), [323](#)
 altlongragged4col ... [278](#), [283](#), [284](#), [324](#)
 altlongragged4col-booktabs [278](#)
 altlongragged4colborder [284](#), [324](#)
 altlongragged4colheader [284](#), [324](#)
 altlongragged4colheaderborder [284](#), [325](#)
 altsuper4col [296](#), [297](#), [301](#), [332](#)
 altsuper4colborder [297](#), [332](#)
 altsuper4colheader [296](#), [332](#)
 altsuper4colheaderborder ... [297](#), [332](#)
 altsuperragged4col [301–303](#), [330](#)
 altsuperragged4colborder ... [302](#), [330](#)
 altsuperragged4colheader ... [302](#), [330](#)
 altsuperragged4colheaderborder .
 [303](#), [330](#)
 almtree [289](#), [309](#), [311](#), [327](#)
 almtreegroup [311](#), [328](#)
 almtreehypergroup [311](#), [328](#)
 index [285](#), [304](#), [305](#), [325](#)
 indexgroup [305](#), [325](#)
 indexhypergroup [305](#), [325](#)
 inline [320](#)
 list [7](#), [266–268](#), [320](#)
 listdotted [268](#), [269](#), [321](#)
 listgroup [266](#), [267](#), [320](#)
 listhypergroup [267](#), [321](#)
 long [269–271](#), [275](#), [280](#), [321](#), [323](#)
 long-booktabs [275](#), [277](#)
 long3col [271](#), [272](#), [276](#), [322](#)
 long3col-booktabs [276](#), [278](#)
 long3colborder [271](#), [322](#)
 long3colheader [272](#), [276](#), [322](#)
 long3colheaderborder [272](#), [322](#)
 long4col [272–274](#), [276](#), [322](#)
 long4col-booktabs [276](#), [277](#)
 long4colborder [273](#), [323](#)
 long4colheader [273](#), [276](#), [323](#)
 long4colheaderborder [273](#), [323](#)
 longborder [270](#), [322](#)
 longheader [270](#), [275](#), [322](#)
 longheaderborder [270](#), [322](#)
 longragged [277](#), [280](#), [281](#)
 longragged-booktabs [277](#)
 longragged3col [278](#), [281](#), [282](#), [324](#)
 longragged3col-booktabs [278](#)
 longragged3colborder [282](#), [324](#)
 longragged3colheader [282](#), [324](#)
 longragged3colheaderborder . [283](#), [324](#)
 longraggedborder [281](#), [323](#)
 longraggedheader [281](#), [324](#)
 longraggedheaderborder [281](#), [324](#)
 mcolalmtree [289](#), [329](#)
 mcolalmtreegroup [290](#), [329](#)
 mcolalmtreehypergroup [290](#), [329](#)
 mcolindex [286](#), [328](#)
 mcolindexgroup [286](#), [328](#)
 mcolindexhypergroup [286](#), [328](#)
 mcoltree [287](#), [328](#)
 mcoltreegroup [328](#)
 mcoltreehypergroup [287](#), [328](#)
 mcoltreename [288](#), [329](#)
 mcoltreenamegroup [288](#), [329](#)
 mcoltreenamehypergroup [288](#), [289](#), [329](#)
 sublistdotted [321](#)
 super [291–293](#), [299](#), [331](#)
 super3col [293](#), [294](#), [331](#)
 super3colborder [293](#), [331](#)
 super3colheader [294](#), [331](#)
 super3colheaderborder [294](#), [331](#)
 super4col [294–296](#), [332](#)
 super4colborder [295](#), [332](#)
 super4colheader [295](#), [332](#)
 super4colheaderborder [296](#), [332](#)
 superborder [292](#), [331](#)
 superheader [292](#), [331](#)
 superheaderborder [292](#), [331](#)
 superragged [298](#), [299](#), [329](#)

superragged3col 299–301, 330
superragged3colborder 300, 330
superragged3colheader 301, 330
superragged3colheaderborder 301, 330
superraggedborder 299, 329
superraggedheader 299, 329
superraggedheaderborder 299, 330
tree 287, 305–307, 309, 325
treegroup 287, 306, 326
treehypergroup 306, 326
treenoname 288, 307, 308, 326
treenonamegroup 308, 327
treenonamehypergroup 308, 327
glossary-hypernav package 155
glossary-list package 7, 9, 265
glossary-long package 8, 269, 283, 291
glossary-longragged package 279
glossary-mcols package 285
glossary-super package .. 8, 9, 269, 291, 297, 301
glossary-superragged package 297
glossary-tree package 9, 303
\glossaryentry 179, 180, 315
glossaryentry (counter) 10, 200, 201
\glossaryentryfield
..... 202, 320–327, 329–332, 354
\glossaryentrynumbers
.. 8, 159, 183–185, 193, 194, 198, 199, 317
\glossaryheader 158,
192, 263, 266–276, 280–288, 290, 291,
293, 294, 298, 300, 301, 304–309, 311, 317
\glossarymark 37
\glossaryname 13, 32, 33
\glossarypostamble 159, 192, 317
\glossarypreamble 158, 192, 317
\glossarysection 158, 192, 317
glossarysubentry (counter) 10, 199–201
\glossarysubentryfield
..... 203, 320–327, 329–332, 354
\glossarytitle .. 158, 183, 184, 192, 195, 317
\glossarytoctitle 7, 13,
14, 27, 28, 30, 33, 37, 158, 183, 192, 196, 317
\glossentry .. 83, 184, 194, 203, 204, 264,
266–269, 271, 272, 280, 282, 283, 291,
293, 295, 298, 300, 302, 304, 306, 307, 309
\Glossentrydesc 353
\glossentrydesc
. 264, 266–273, 280, 282, 283, 291–293,
295, 298, 300, 302, 304–308, 310, 311, 353
\glossentryname 264, 266–
269, 271, 272, 280, 282, 283, 291, 293,
295, 298, 300, 302, 304–307, 310, 311, 353
\Glossentrysymbol 354
\glossentrysymbol 264, 272, 273,
283, 284, 295, 302, 304–307, 310, 311, 354
\Gls 91, 211, 229
\gls 91, 169, 200, 211, 229
\gls@Alphpage 176, 178
\gls@alphpage 176, 178
\gls@arabicpage 176, 178
\gls@assign@desc 76, 81
\gls@assign@descplural
..... 231, 239, 242, 244, 364, 367, 368
\gls@assign@field
..... 67, 71, 72, 76, 78, 80–82, 254, 255
\gls@assign@firstpl 230,
231, 233, 235, 237, 239, 242, 244, 364–368
\gls@assign@plural
231, 233, 235, 237, 239, 242, 244, 364–368
\gls@assign@symbolplural
231, 233, 235, 237, 242, 244, 365, 366, 368
\gls@checkisacronymlist 106
\gls@checkseeallowed 62, 67, 168, 170
\gls@checkseeallowed@preambleonly .. 67
\gls@codepage 48, 166, 186
\gls@defdocnewglossaryentry 68, 88
\gls@defdocglossaryentry 67, 68, 77
\gls@disablepagerefexpansion .. 175, 178
\gls@do@addxdyattribute 42
\gls@doclearpage 40
\gls@dostub 111
\gls@dotocitle 183, 184, 195, 196
\gls@end@sanitizesort 19
\gls@endcheck 66, 67
\gls@glossary 174, 179, 180
\gls@gobbleopt 58
\gls@grplabel 261
\gls@hypergrouprrun 262
\gls@ifnotmeasuring 85, 86
\gls@inlinepostchild 263–265, 320
\gls@inlinesep 263, 264, 320
\gls@inlinesubsep 263, 264, 320
\gls@islistofacronyms 15
\gls@istfilebase 34, 166
\gls@label 211
\gls@level 79, 80, 193
\gls@noidxglossary 170
\gls@nosetquote 77, 160, 162, 164

<code>\gls@numberpage</code>	176, 177	<code>\glscustomtext</code>	...
<code>\gls@org@glossaryentryfield</code>	184		94, 97, 99, 100, 102, 119–124, 137–
<code>\gls@org@glossarysubentryfield</code>	184		144, 223, 224, 232, 236, 238, 239, 241,
<code>\gls@org@insert</code>	236, 238, 241		341, 344, 345, 347, 348, 350–353, 358, 359
<code>\gls@protected@pagefmts</code>	110, 176, 177	<code>\GlsDeclareNoHyperList</code>	16
<code>\gls@Romanpage</code>	176, 178	<code>\glsdefaulttype</code>	13,
<code>\gls@romanpage</code>	176, 178		37, 48, 49, 56, 57, 78, 93, 103, 174, 182, 183
<code>\gls@save@numberlist</code>	8	<code>\glsdefmain</code>	13, 59
<code>\gls@suffiF</code>	35, 159, 161, 317, 319	<code>\glsdescriptionaccessdisplay</code>	...
<code>\gls@suffiFF</code>	36, 159, 161, 317, 319		343–345, 353, 354
<code>\gls@text</code>	102	<code>\glsdescriptionpluralaccessdisplay</code>	...
<code>\gls@thissty</code>	23		341, 342
<code>\gls@tmp</code>	173, 174, 240	<code>\glsdescwidth</code>	269–271, 274,
<code>\gls@tmplen</code>	117, 308–311, 327, 328		275, 277, 278, 280–285, 291–294, 296–303
<code>\gls@tr@set@acronym@toctitle</code>	14	<code>\glsdetoklabel</code>	51–55, 63, 68, 73–77,
<code>\gls@tr@set@main@toctitle</code>	13		83, 86–90, 107, 145, 146, 152–154, 170–
<code>\gls@tr@set@numbers@toctitle</code>	28		172, 178, 184, 187–189, 193, 195, 197,
<code>\gls@tr@set@symbols@toctitle</code>	27		200, 202, 247–252, 309, 333, 334, 369, 370
<code>\gls@wrglossary</code>	175	<code>\glsdisplay</code>	94, 103
<code>\gls@xdystring</code>	110, 111	<code>\glsdisplayfirst</code>	94, 103
<code>\gls@xindy@glsnumbersfalse</code>	26	<code>\glsdisplaynumberlist</code>	171
<code>\gls@xindy@glsnumberstrue</code>	25	<code>\glsdohyperlink</code>	117, 118
<code>\glsaccsupp</code>	339	<code>\glsdohypertarget</code>	118
<code>\glsacronymtrue</code>	14	<code>\glsdoifexists</code>	...
<code>\glsacrpluralsuffix</code>	...		52–54, 73–76, 85, 86, 119–124, 137–
	31, 212, 221, 225–227, 231		144, 152, 153, 171, 172, 256–260, 350–354
<code>\glsacrshortcutsfalse</code>	29	<code>\glsdoifexistsordo</code>	106, 145
<code>\glsacrshortcutstrue</code>	29	<code>\glsdoifexistsorwarn</code>	195, 202, 203
<code>\glsacspace</code>	219, 222	<code>\glsdoifnoexists</code>	67, 76
<code>\glsadd</code>	154	<code>\glsdonohyperlink</code>	108, 117, 118
<code>\glsadd options</code>		<code>\glsdosanitizesort</code>	10, 11
<code>counter</code>	153	<code>\glsentryaccess</code>	339
<code>format</code>	153, 208	<code>\glsentrycounter</code>	206, 209
<code>\glsaddall options</code>		<code>\glsentrycounterfalse</code>	10
<code>types</code>	153, 154	<code>\glsentrycounterlabel</code>	197, 201
<code>\GlsAddXdyAttribute</code>	41–43, 313, 314	<code>\glsentrycountertrue</code>	10
<code>\GlsAddXdyCounters</code>	42, 43, 58	<code>\glsentrycurrcount</code>	88, 90
<code>\glsautomakefalse</code>	26	<code>\Glsentrydesc</code>	129, 203, 353
<code>\glsautoprefix</code>	7, 196	<code>\glsentrydesc</code>	96, 97, 129, 202, 343–345, 353
<code>\glscapscase</code>	94, 95, 98–101,	<code>\glsentrydescaccess</code>	340
	119–124, 137–144, 223, 224, 236, 241,	<code>\Glsentrydescplural</code>	130
	341, 343, 345, 346, 348, 349, 351–353, 358	<code>\glsentrydescplural</code>	94, 95, 130, 341, 342
<code>\glsclearpage</code>	39	<code>\glsentrydescpluralaccess</code>	340
<code>\glsclosebrace</code>	46, 159, 160, 317, 318	<code>\Glsentryfirst</code>	92, 96, 99, 126, 344, 347
<code>\glscompositor</code>	35, 45, 161, 316, 319	<code>\glsentryfirst</code>	91, 96, 97, 99, 126, 343, 344, 347
<code>\glscounter</code>	16, 29, 41, 58, 81, 108, 313	<code>\glsentryfirstaccess</code>	340
<code>\glscurrententrylabel</code>	182, 184	<code>\Glsentryfirstplural</code>	93, 95, 98, 127, 342, 346
<code>\glscurrentfieldvalue</code>	54, 55	<code>\glsentryfirstplural</code>	92,
			94, 95, 98, 127, 128, 341, 342, 345, 346

<code>\glsentryfirstpluralaccess</code>	340	<code>\glsentrysymbol</code>	96, 97, 130, 131, 203, 232, 236, 241, 343–345, 354
<code>\glsentryfmt</code>	58, 59	<code>\glsentrysymbolaccess</code>	340
<code>\Glsentryfull</code>	217, 225, 227, 360, 362	<code>\Glsentrysymbolplural</code>	132
<code>\glsentryfull</code>	217, 225, 227, 359, 362	<code>\glsentrysymbolplural</code>	.. 94, 95, 131, 132, 232, 236, 241, 341, 342
<code>\Glsentryfullpl</code>	217, 225, 227, 360, 362	<code>\glsentrysymbolpluralaccess</code>	340
<code>\glsentryfullpl</code>	217, 225, 227, 360, 362	<code>\Glsentrytext</code>	96, 99, 125, 343, 347
<code>\glsentryitem</code>	197, 264, 266–269, 271, 272, 280, 282, 283, 291, 293, 295, 298, 300, 302, 304, 306, 307, 310, 320–327, 329–332	<code>\glsentrytext</code>	96, 97, 99, 125, 153, 181, 343, 344, 346, 347
<code>\Glsentrylong</code>	92, 141, 146, 151, 219, 224, 225, 350, 352, 355, 358–360	<code>\glsentrytextaccess</code>	339
<code>\glsentrylong</code>	91, 102, 141, 142, 146, 151, 218–228, 238, 350, 352–363	<code>\glsentrytype</code>	78
<code>\glsentrylongaccess</code>	340	<code>\Glsentryuseri</code>	132
<code>\Glsentrylongpl</code>	93, 143, 151, 219, 224, 225, 350, 355, 358–360	<code>\glsentryuseri</code>	132, 133
<code>\glsentrylongpl</code>	.. 92, 102, 143, 144, 151, 219, 220, 224–227, 238, 246, 350, 355, 356, 358–362	<code>\Glsentryuserii</code>	133
<code>\glsentrylongpluralaccess</code>	340	<code>\glsentryuserii</code>	133
<code>\Glsentryname</code>	128, 202, 353	<code>\Glsentryuseriii</code>	134
<code>\glsentryname</code>	128, 129, 308, 353	<code>\glsentryuseriii</code>	134
<code>\glsentrynumberlist</code>	152, 171	<code>\Glsentryuseriv</code>	135
<code>\Glsentryplural</code>	95, 98, 127, 342, 346	<code>\glsentryuseriv</code>	135
<code>\glsentryplural</code>	.. 94, 95, 98, 126, 127, 341, 342, 345, 346	<code>\Glsentryuserv</code>	136
<code>\glsentrypluralaccess</code>	339	<code>\glsentryuserv</code>	135, 136
<code>\Glsentryprefix</code>	258	<code>\Glsentryuservi</code>	136
<code>\glsentryprefix</code>	256, 259	<code>\glsentryuservi</code>	136, 137
<code>\Glsentryprefixfirst</code>	258	<code>\glsfieldfetch</code>	148
<code>\glsentryprefixfirst</code>	257, 259	<code>\glsfirstaccessdisplay</code>	343, 344, 347
<code>\Glsentryprefixfirstplural</code>	259	<code>\glsfirstpluralaccessdisplay</code>	.. 341, 342, 345, 346
<code>\glsentryprefixfirstplural</code>	257, 260	<code>\glsfirstpluralacesdisplay</code>	346
<code>\Glsentryprefixplural</code>	258	<code>\glsngenacfmt</code>	218–220, 226, 354, 355, 360
<code>\glsentryprefixplural</code>	257, 260	<code>\glsngenentryfmt</code>	.. 218–220, 224, 226, 230, 232, 234, 236, 238, 241, 243, 246, 354, 355, 359, 360
<code>\glsentryprevcount</code>	88–90	<code>\glsgetgrouptitle</code>	263, 267, 268, 286–290, 305–308, 311, 312
<code>\Glsentryshort</code>	100, 138, 146, 220, 226, 227, 349–351, 355, 361, 362	<code>\gls glossarymark</code>	37
<code>\glsentryshort</code>	.. 100, 102, 137, 138, 146, 151, 217–228, 348–351, 354–357, 359–363	<code>\gls groupheading</code>	160, 193, 263, 266–269, 271, 272, 280, 282, 283, 286–291, 293, 295, 298, 300, 301, 304–309, 311, 312, 318
<code>\glsentryshortaccess</code>	340	<code>\gls groupskip</code>	159, 193, 264, 266, 270, 271, 273, 276, 277, 280, 282, 284, 292, 293, 295, 298–300, 302, 305, 306, 308, 311, 317
<code>\Glsentryshortpl</code>	100, 140, 220, 226, 227, 348, 355, 361, 362	<code>\glshyperfirstfalse</code>	226, 360
<code>\glsentryshortpl</code>	.. 100, 102, 139, 140, 151, 219, 220, 225–227, 246, 348, 350, 355, 359–362	<code>\glshyperfirsttrue</code>	24
<code>\glsentryshortpluralaccess</code>	340	<code>\glshyperlink</code>	181
<code>\Glsentrysymbol</code>	131, 203, 354	<code>\glshypernavsep</code>	263
		<code>\glshypernumber</code>	36, 210
		<code>\glsifhyperon</code>	105
		<code>\glsIfListOfAcronyms</code>	15

<code>\glsifplural</code>	94, 97, 100, 101, 119–124, 137–144, 223, 232, 236, 238, 241, 341, 345, 348, 349, 351–353, 358	<code>\glsnamefont</code>	202, 203, 333, 334, 353
<code>\glsifusetranslator</code>	22, 23, 32, 33, 371	<code>\glsnavhyperlink</code>	263
<code>\glsindexonlyfirstfalse</code>	24	<code>\glsnavhypertarget</code> 267, 268, 286–290, 305, 307, 308, 312
<code>\glsinlinedescformat</code>	264, 320	<code>\glsnavigation</code> 267, 268, 286–290, 305, 307, 308, 312
<code>\glsinlinedopostchild</code>	264, 320	<code>\glsnextpages</code>	8, 62, 184
<code>\glsinlinedemptydescformat</code>	264, 320	<code>\glsnogroupskipfalse</code>	9
<code>\glsinlinenameformat</code>	264, 320	<code>\glsnoidxdisplayloc</code>	172, 173
<code>\glsinlineparentchildseparator</code>	264, 320	<code>\glsnoidxdisplayloclisthandler</code>	171
<code>\glsinlinepostchild</code>	264, 320	<code>\glsnoidxloclist</code>	171, 193, 194
<code>\glsinlineseparator</code>	264, 320	<code>\glsnoidxloclisthandler</code>	194
<code>\glsinlinesubdescformat</code>	264, 320	<code>\glsnoidxnumberlistloophandler</code>	172
<code>\glsinlinesubnameformat</code>	264, 320	<code>\glsnoidxstripaccents</code>	19
<code>\glsinlinesubseparator</code>	264, 320	<code>\glsnomakeindexwarning</code>	162
<code>\glsinsert</code> 94–101, 119–124, 137–144, 224, 232, 236, 238, 239, 241, 341–353, 358, 359		<code>\glsnonextpages</code>	62, 184
<code>\glskeylisttok</code>	216, 230, 231, 233– 235, 237, 239, 240, 242–244, 246, 363–368	<code>\glsnopostdotfalse</code>	9
<code>\glslabel</code>	77, 94–101, 106–108, 138–144, 218–220, 223, 224, 226, 232, 236, 238, 241, 341–350, 354, 355, 358–360	<code>\glsnoxindywarning</code> 35, 41, 43, 44, 46–48, 156	
<code>\glslabeltok</code>	216, 230–237, 239–241, 243, 244, 246, 364–367	<code>\glsnumberformat</code>	152
<code>\glslink</code>	216, 217, 224–227, 232, 359, 361	<code>\glsnumberlistloop</code>	172
<code>\glslink options</code>		<code>\glsnumbersgroupname</code>	27, 33, 206
counter	104, 118, 253	<code>\glsnumlistlastsep</code>	152, 171
format	104, 118, 208	<code>\glsnumlistparser</code>	153, 168
hyper	104, 107, 118	<code>\glsnumlistsep</code>	152, 171
local	104	<code>\glsopenbrace</code>	46, 159, 160, 317, 318
<code>\glslinkcheckfirsthyperhook</code>	107	<code>\glsorder</code>	25, 166, 167, 190
<code>\glslinkpostsetkeys</code>	108	<code>\glsorg@endtheglossary</code>	5
<code>\glslinkvar</code>	105	<code>\glsorg@PrintChanges</code>	5
<code>\glslistdottedwidth</code>	268, 321	<code>\glsorg@theglossary</code>	5
<code>\glslistgroupheaderfmt</code>	267, 268	<code>\glspagelistwidth</code> ...	271, 274, 275, 277, 278, 281–285, 293, 294, 296, 297, 300–303
<code>\glslistnavigationitem</code>	267, 268	<code>\glspatchLToutput</code>	275–278
<code>\glslocalreset</code>	87	<code>\glspenaltygroupskip</code>	276, 277
<code>\glslocalunset</code>	88, 119–124	<code>\glspercentchar</code>	68, 69, 158, 160
<code>\glslongaccessdisplay</code>	350, 352–364	<code>\Glspl</code>	92, 230
<code>\glslongkey</code>	368	<code>\glspl</code>	92, 230
<code>\glslongpluralaccessdisplay</code> 350, 355, 356, 358–362, 364	<code>\glspluralaccessdisplay</code>	341, 342, 345, 346
<code>\glslongpluralkey</code>	368	<code>\glspluralsuffix</code> 31, 80, 81, 219, 220, 355, 356, 360–362
<code>\glslongtok</code>	216–220, 224, 226, 230, 231, 233–235, 237, 239, 240, 242– 244, 246, 247, 354, 355, 359, 360, 363–368	<code>\glspostdescription</code>	34, 265–267, 269, 270, 280, 291, 292, 298, 304–308, 310, 311, 320–323, 325–329, 331
<code>\glsLTpenaltycheck</code>	279	<code>\glspostinline</code>	263
<code>\glsmcols</code>	285–290	<code>\glspostlinkhook</code> 106, 119–124, 137–144, 351–353
<code>\glsnameaccessdisplay</code>	353, 354	<code>\glsprestandardsort</code>	11
		<code>\glsreset</code>	87
		<code>\glsresetentrycounter</code>	197, 200
		<code>\glsresetentrylist</code> .	159, 192, 198, 199, 317

<code>\glsresetsubentrycounter</code>	<code>\glstreenamebox</code>	310, 311
..... 197, 198, 201, 264, 320	<code>\glstreenamefmt</code>	304–311
<code>\glssanitizesortfalse</code>	<code>\glstreenavigationfmt</code>	286–290, 305, 307, 308, 312
..... 21	<code>\glstype</code> .	106, 107, 119–124, 137–144, 351–353
<code>\glssanitizesorttrue</code>	<code>\glsucmarkfalse</code>	10
..... 21	<code>\glsucmarktrue</code>	10
<code>\glsavenumberlistfalse</code>	<code>\glsunset</code>	87, 89, 90, 119–124
..... 8	<code>\glsupacrpluralsuffix</code>	220, 221, 227, 234, 238, 240, 243
<code>\glsavewritesfalse</code>	<code>\GlsUseAcrEntryDispStyle</code> ...	217, 220–223, 225, 227, 228, 356, 357, 360, 362, 363
..... 27	<code>\GlsUseAcrStyleDefs</code>	217, 220–223, 225, 227, 228, 356, 357, 360, 362, 363
<code>\glsseeformat</code>	<code>\glswrallowprimitivemodstrue</code>	177
..... 158, 170, 172, 317	<code>\glswrite</code> ...	156–161, 167, 173, 174, 315–319
<code>\glsseeitem</code>	<code>\glswritedefhook</code>	69
..... 181	<code>\glswriteentry</code>	175
<code>\glsseeitemformat</code>	<code>\glswritefiles</code>	26, 27, 173
..... 181	<code>\glsxindyfalse</code>	25
<code>\glsseeleastsep</code>	<code>\glsxindytrue</code>	26
..... 181		
<code>\glsseeesep</code>		
..... 181		
<code>\glssetexpandfield</code>		
..... 18, 20, 21		
<code>\glssetnoexpandfield</code>		
..... 18, 20, 21		
<code>\GlsSetQuote</code>		
..... 77, 160		
<code>\glssettoctitle</code>		
..... 33, 183		
<code>\glsshortaccessdisplay</code>		
..... 348–351, 354–357, 359–364		
<code>\glsshortkey</code>		
..... 368		
<code>\glsshortpluralaccessdisplay</code>		
..... 348, 350, 355, 359–362, 364		
<code>\glsshortpluralkey</code>		
..... 368		
<code>\glsshorttok</code>		
..... 216, 230–235, 237, 239–244, 246, 364–368		
<code>\glssortnumberfmt</code>		
..... 12		
<code>\glsspace</code>		
..... 213		
<code>\glsstepentry</code>		
..... 197, 201		
<code>\glsstepsubentry</code>		
..... 197, 198, 201		
<code>\glssubentrycounterfalse</code>		
..... 10		
<code>\glssubentrycounterlabel</code> ...		
..... 197, 198, 201		
<code>\glssubentryitem</code>		
..... 197, 198, 264, 266–268, 270, 271, 273, 280, 282, 283, 291, 293, 295, 298, 300, 302, 305–307, 310, 320–327, 329–332		
<code>\glssymbolaccessdisplay</code> ...		
..... 343–345, 354		
<code>\glssymbolpluralaccessdisplay</code> .		
..... 341, 342		
<code>\glssymbolsgroupname</code>		
..... 27, 33, 206		
<code>\glstarget</code>		
..... 204, 265–273, 280, 282, 283, 291–293, 295, 298, 300, 302, 304–307, 310, 311, 320–332		
<code>\glstextaccessdisplay</code> ..		
..... 343, 344, 346, 347		
<code>\glstextformat</code>		
..... 106, 108		
<code>\glstextup</code>		
..... 31, 362		
<code>\glstildechar</code>		
..... 42, 158, 159		
<code>\glstranslatefalse</code>		
..... 22, 23		
<code>\glstranslatetrue</code>		
..... 23		
<code>\glstreegroupheaderfmt</code>		
..... 286–290, 305–308, 311, 312		
<code>\glstreeindent</code> ..		
..... 306, 307, 309–311, 326–328		
	H	
	<code>\H</code>	19
	<code>\hangindent</code>	289, 290, 306, 307, 309, 311, 312, 325–328
	<code>\hbox</code>	268, 321
	<code>\hfill</code>	268, 321
	<code>\hline</code>	270–274, 281–285, 292–294, 296, 297, 299–301, 303
	<code>\hsize</code>	268, 269, 280, 291, 298
	<code>\hss</code>	268, 321
	<code>\ht</code>	279
	<code>\hyperdef</code>	29
	<code>\hyperlink</code>	104, 117, 209
	<code>hyperref package</code>	179, 182, 208, 253
	<code>\hypertarget</code>	117
	I	
	<code>\IeC</code>	19
	<code>\if</code>	110, 178, 179, 314
	<code>\if@endfor</code>	262
	<code>\if@gl@debug</code>	5, 17, 175
	<code>\if@gl@docloaded</code>	4, 13, 174
	<code>\if@gl@isacronymlist</code>	106
	<code>\if@openright</code>	39
	<code>\ifbool</code>	14, 24, 27, 51, 94–96
	<code>\ifboolexpr</code>	32, 56, 205
	<code>\ifcase</code>	6, 7, 23, 62, 196, 304, 325

<code>\ifcsdef</code>	22, 33, 39, 65, 66, 72–76, 103, 175, 186, 187, 190, 191, 207, 218	<code>\ifglindexonlyfirst</code>	176
<code>\ifcsequal</code>	53	<code>\ifglfnogroupskip</code>	266, 270, 271, 273, 275–277, 280, 282, 284, 292, 293, 295, 298, 300, 302, 305, 306, 308, 311
<code>\ifcsstring</code>	75	<code>\ifglfnonumberlist</code>	198
<code>\ifcsundef</code>	6, 25, 29, 32, 36, 37, 39, 50, 51, 58, 59, 61, 78, 81, 88, 89, 104, 108, 109, 117, 118, 166, 173, 182, 185, 187, 195, 196, 201, 205–208, 211, 217, 262, 319	<code>\ifglfnopostdot</code>	9
<code>\ifdef</code>	54, 55, 63, 68, 104, 145, 148, 171, 172, 212	<code>\ifglfnnumberline</code>	40
<code>\ifdefempty</code>	16, 38, 39, 50, 53–55, 59, 94, 97, 100, 169, 192, 193, 216, 217, 223, 232, 236, 238, 240, 241, 341, 345, 348, 358	<code>\ifglssanitizesort</code>	18, 21
<code>\ifdefequal</code>	52–55, 65–67, 70, 79, 83, 193	<code>\ifglssavenumberlist</code>	65, 168, 182
<code>\ifdefstrequal</code>	76	<code>\ifglssavewrites</code>	26, 165, 175
<code>\ifdefstring</code>	32, 56, 166, 168, 189, 190, 193, 194	<code>\ifglssubentrycounter</code>	197, 199–201
<code>\ifdefvoid</code>	19, 82, 193, 194	<code>\ifglstoc</code>	40
<code>\ifdim</code>	220, 279, 309	<code>\ifglstranslate</code>	32
<code>\iffalse</code>	82, 87	<code>\ifglsucmark</code>	38
<code>\IfFileExists</code>	8, 22, 23, 185	<code>\ifglused</code>	90, 94–100, 106, 154, 176, 232, 236, 238, 241, 256–260, 341–348
<code>\ifglossaryexists</code>	37, 48, 52, 165, 166	<code>\ifglswrallowprimitivemods</code>	177
<code>\ifgl@sanitize@description</code>	20	<code>\ifglxindy</code> .	34, 35, 40–48, 58, 83, 84, 111, 155, 156, 162, 166, 178, 180, 185, 313–315
<code>\ifgl@sanitize@name</code>	20	<code>\ifignoredglossary</code>	79, 82, 175
<code>\ifgl@sanitize@symbol</code>	20	<code>\ifin@</code>	28
<code>\ifgl@xindy@glnumbers</code>	49	<code>\ifinlistcs</code>	191, 195
<code>\ifgl@sacrdescription</code>	245	<code>\ifKV@glslink@hyper</code>	107, 108
<code>\ifgl@sacrdua</code>	234, 240, 243, 245	<code>\ifKV@glslink@local</code>	119–124
<code>\ifgl@sacrfootnote</code>	106, 245	<code>\ifmeasuring@</code>	85
<code>\ifgl@sacr synonym</code>	13, 14	<code>\ifnum</code>	11, 89, 90, 192, 278, 279, 306, 307, 309, 310, 326, 327
<code>\ifgl@sacrshortcuts</code>	29, 230	<code>\ifstrempty</code>	320
<code>\ifgl@sacrsmalldcaps</code> .	234, 235, 238, 240, 243	<code>\ifstrequal</code>	205
<code>\ifgl@sacrsmalld</code>	234, 235, 238, 240	<code>\ifthenelse</code>	21, 29, 39, 108, 168, 206, 245, 261
<code>\ifgl@sacrautomaake</code>	26, 169	<code>\IfTrackedLanguage</code>	162
<code>\ifgl@sacrdescsuppressed</code>	264	<code>\IfTrackedLanguageFileExists</code>	33, 371, 372
<code>\ifgl@sacrsentrycounter</code>	197, 199–201	<code>\iftrue</code>	82, 86
<code>\ifgl@sacrsentryexists</code>	51, 52, 68, 77, 79	<code>\ifundef</code>	57, 68, 78, 156, 160, 167, 197
<code>\ifgl@shaschildren</code>	264, 320	<code>\ifvmode</code>	154
<code>\ifgl@shasdesc</code>	264	<code>\ifvoid</code>	279
<code>\ifgl@shaslong</code>	91–93, 146, 218–220, 223, 226, 238, 354, 355, 358, 360	<code>\ifx</code>	11, 12, 15, 28, 30, 41, 42, 44, 46, 48, 49, 79–82, 84, 109, 112– 117, 146, 156, 159, 161, 162, 164, 174, 177–181, 183, 184, 197, 199, 206–209, 231, 233, 235, 237, 239, 240, 242, 244, 246, 247, 315–317, 319, 320, 325–328, 339
<code>\ifgl@shasparent</code>	187, 193, 308	<code>\immediate</code>	68, 69, 90, 166, 174, 185, 186
<code>\ifgl@shasprefix</code>	258	<code>\in@</code>	28
<code>\ifgl@shasprefixfirst</code>	258	<code>\index</code>	175
<code>\ifgl@shasprefixfirstplural</code>	258	<code>\indexname</code>	28
<code>\ifgl@shasprefixplural</code>	258	<code>\indexspace</code> .	266, 286–290, 305–308, 311, 312
<code>\ifgl@shassymbol</code>		<code>\input</code>	31, 93
<code>\ifgl@shashyperfirst</code>	232, 236, 241, 304–307, 310, 311	<code>\inputencodingname</code>	25

<code>\InputIfFileExists</code>	68	<code>makeglossaries</code> ..	25, 34, 48, 57, 162, 167, 185
<code>\istfilename</code>	34, 156, 160, 167, 315, 318	<code>\makeglossaries</code>	6, 26, 30, 62, 169–171, 173, 186
<code>\item</code>	266–269, 286, 304, 305, 320, 321, 325	<code>\makeglossary</code>	165, 168
J			
<code>\jobname</code>	34, 68, 156, 160, 166, 167, 185, 315, 318	<code>makeindex</code>	373
K			
<code>\key@ifundefined</code>	70, 71	<code>makeindex</code>	10, 25, 26, 31, 34–36, 40, 56–58, 60, 84, 109, 113, 155, 158, 160, 162, 165, 174, 178, 179, 204, 205, 314, 315
<code>\KV@glslink@hyperfalse</code> .	104, 106, 107, 118	<code>delim_n</code>	36
<code>\KV@glslink@hypertrue</code>	104, 118	<code>delim_r</code>	36
L			
<code>\L</code>	20	<code>page_compositor</code>	35
<code>\l</code>	20	<code>special characters</code>	111, 112, 155
<code>\label</code>	7, 196, 197, 200	<code>\makenoidxglossaries</code> ..	6, 62, 168, 172, 173
<code>\language</code>	25	<code>\MakeTextUppercase</code>	4
<code>\leaders</code>	268, 321	<code>\MakeUppercase</code>	342, 344, 351, 353
<code>\leavevmode</code>	76, 107	<code>\markboth</code>	38
<code>\let</code> ..	5, 9, 11–14, 19, 20, 22, 23, 26–29, 32, 34, 43, 54–56, 65, 67, 76–82, 85–88, 90, 93, 105–108, 110, 111, 117–124, 137– 144, 146, 152, 153, 160, 161, 165–170, 172, 175–177, 181, 183–185, 193, 195, 196, 199, 203, 216, 228–231, 233, 235– 239, 241, 242, 244, 254, 261–263, 278, 286, 304, 319, 336, 351–353, 364–368, 371	<code>\mbox</code>	154, 267, 289, 290, 309, 321
<code>\letcs</code>	52– 55, 68, 71, 72, 75, 80, 81, 145, 146, 166, 171, 172, 187–189, 193, 202, 205, 206, 309	<code>memoir class</code>	174
<code>link text</code>	<u>93</u>	<code>\memUchead</code>	38
<code>\listcsadd</code>	191	<code>\MessageBreak</code>	33, 56, 183, 371, 372
<code>\listcsgadd</code>	195	<code>mfirstuc package</code>	1
<code>\listcsxadd</code>	186, 187	<code>\mfirstucMakeUppercase</code> 4, 38, 73, 95, 97–101, 125–138, 140, 142, 144, 216, 217, 224–227, 236, 241, 259, 260, 346–350, 358, 359, 361, 362
<code>\listeadd</code>	191	<code>\midrule</code>	275–277
<code>\loadglsentries</code>	93	<code>\month</code>	156, 160, 315, 318
<code>\long</code>	76, 210	<code>multicol package</code>	285
<code>\longnewglossaryentry</code>	77	N	
<code>longtable package</code>	269, 275, 279	<code>\n</code>	160, 161, 318
<code>\LT@end@open</code>	279	<code>\NeedsTeXFormat</code> ..	4, 254, 313, 319, 333, 371
<code>\LT@err</code>	279	<code>\new@glossaryentry</code>	67, 171
<code>\LT@foot</code>	279	<code>\new@ifnextchar</code>	56, 72, 73, 91, 92, 119–123, 125–144, 212–215, 256–259
<code>\LT@head</code>	279	<code>\newacronym</code>	211, 216, 231, 233, 235, 237, 239, 242, 244, 246
<code>\LT@lastfoot</code>	279	<code>\newacronymhook</code>	216, 231, 233, 235, 237, 240, 243, 244, 247, 363
<code>\LT@output</code>	278, 279	<code>\newacronymstyle</code>	218–223, 225–228
M			
<code>\makeatletter</code>	68, 185	<code>\newcommand</code> 5–19, 21, 22, 24–32, 34–44, 46–77, 83–88, 90–94, 97, 100, 102, 103, 105– 108, 110, 111, 117–156, 162, 164–167, 169, 172–183, 185–191, 193–195, 198– 208, 210–218, 220, 228, 230–239, 241– 253, 255–259, 261–266, 278, 279, 285, 303, 304, 309, 319, 337–339, 354, 368–370
<code>\makeatother</code>	68	<code>\newcount</code>	11, 12, 65
<code>\makebox</code>	268, 309–311, 321, 327, 328	<code>\newcounter</code>	197, 199

<code>\newenvironment</code>	201	<code>user6</code>	136, 150, 335
<code>\newglossary</code>	13, 14, 27, 28, 58, 168	<code>\newglossarystyle</code>	263, 266–278, 280–309, 311
<code>\newglossaryentry</code>	28, 64, 67, 88, 216, 230, 232, 234, 236, 239, 241, 243, 246, 364–367	<code>\newif</code>	4, 15, 22, 25, 177
<code>\newglossaryentry options</code>		<code>\newlength</code> ..	117, 268, 269, 280, 291, 298, 307
<code>access</code>	336, 337	<code>\newrobustcmd</code>	
<code>counter</code>	61	...	67, 68, 72, 73, 91, 92, 106, 118–123, 125–151, 153, 154, 212–215, 255–259, 308
<code>description</code>		<code>\newterm</code>	28
. 24, 60, 64, 67, 77, 129, 146, 212, 239, 335		<code>\newtoks</code>	111, 165, 215
<code>descriptionaccess</code>	338, 340	<code>\newwrite</code>	68, 156, 160, 165, 167
<code>descriptionplural</code>	129, 335	<code>ngerman package</code>	162
<code>descriptionpluralaccess</code>	338, 340	<code>\noalign</code>	278
<code>first</code> . 61, 80, 118, 125, 148, 237, 242, 243, 334		<code>\nobreak</code>	267, 279, 321
<code>firstaccess</code>	338, 340	<code>\noexpand</code> . 15, 30, 41, 42, 81, 82, 103, 108–	
<code>firstplural</code>	61, 127, 148, 335	111, 116, 117, 152, 162–164, 166–168,	
<code>firstpluralaccess</code>	338, 340	177, 178, 182, 185, 186, 188, 189, 202,	
<code>format</code>	156	203, 211, 216, 230, 232–234, 236, 237,	
<code>long</code>	100, 151, 335	239, 241–244, 246, 313, 333, 334, 364–368	
<code>longaccess</code>	339, 340	<code>\nohyppage</code>	208
<code>longplural</code>	151, 335	<code>\noindent</code>	204, 286–290, 306–308
<code>longpluralaccess</code>	339, 340	<code>\noist</code>	318, 319
<code>name</code>	60, 64, 67, 77, 128, 145, 181, 334	<code>\nopostdesc</code>	28, 34, 76, 184, 320
<code>nonumberlist</code>	62, 63	<code>\normalbaselineskip</code>	278
<code>parent</code>	62, 67	<code>\nr</code>	6, 7, 22, 23, 62, 196
<code>plural</code>	61, 80, 126, 334	<code>\ns@ACRfull</code>	213
<code>pluralaccess</code>	338, 339	<code>\ns@Acrfull</code>	213
<code>prefix</code>	254	<code>\ns@acrfull</code>	212
<code>prefixfirst</code>	254	<code>\ns@ACRfullpl</code>	215
<code>prefixfirstplural</code>	255	<code>\ns@Acrfullpl</code>	214
<code>prefixplural</code>	255	<code>\ns@acrfullpl</code>	214
<code>see</code>	5, 8, 62, 67, 168, 170	<code>\ns@ACRlong</code>	142
<code>short</code>	100, 150, 335	<code>\ns@Acrlong</code>	141
<code>shortaccess</code>	338, 340	<code>\ns@acrlong</code>	140
<code>shortplural</code>	150, 335	<code>\ns@ACRlongpl</code>	144
<code>shortpluralaccess</code>	338, 340	<code>\ns@Acrlongpl</code>	143
<code>sort</code>	60, 149, 204, 205	<code>\ns@acrlongpl</code>	142
<code>symbol</code>	60, 61, 130, 233–235, 237, 242, 272, 294, 334–336	<code>\ns@ACRshort</code>	138
<code>symbolaccess</code>	338, 340	<code>\ns@Acrshort</code>	137
<code>symbolplural</code>	131, 335	<code>\ns@acrshort</code>	137
<code>symbolpluralaccess</code>	338, 340	<code>\ns@ACRshortpl</code>	140
<code>text</code> 60, 61, 118, 124, 147, 233, 237, 334		<code>\ns@Acrshortpl</code>	139
<code>textaccess</code>	337, 339	<code>\ns@acrshortpl</code>	139
<code>type</code>	13, 61, 93, 149	<code>\ns@newglossary</code>	57
<code>user1</code>	132, 149, 335	<code>\null</code>	110–117, 162–164, 185
<code>user2</code>	133, 149	<code>\number</code>	11, 80, 90, 176, 177, 203, 334
<code>user3</code>	133, 149	<code>\numberline</code>	40
<code>user4</code>	134, 150	<code>\numexpr</code>	90
<code>user5</code>	135, 150		
		O	
		<code>\O</code>	20

<code>\o</code>	20	<code>true</code>	167, 173
<code>\OE</code>	20	<code>section</code>	6, 38
<code>\oe</code>	20	<code>sort</code>	
<code>\openout</code>	68, 156, 160, 166, 315, 318	<code>def</code>	10, 11
<code>\OR</code>	245	<code>standard</code>	10
<code>\or</code>	6, 7, 23, 196, 304, 325	<code>use</code>	10, 11
<code>\org@glossaryentrynumbers</code>	184, 199	<code>style</code>	7, 247
<code>\org@glossarytitle</code>	183, 184	<code>subentrycounter</code>	197, 199
<code>\org@glspostdescription</code>	34	<code>toc</code>	6
<code>\org@ifKV@glslink@hyper</code>	107, 108	<code>true</code>	6
<code>\orgAlph</code>	177, 178	<code>translate</code>	22
<code>\orgalph</code>	177, 178	<code>false</code>	22
<code>\orgarabic</code>	177, 178	<code>translator</code>	22
<code>\orgnumber</code>	177	<code>xindy</code>	25, 26, 158, 253
<code>\orgRoman</code>	177, 178	<code>\PackageError</code>	5, 6, 26, 30, 42, 48, 51, 52, 56, 62, 64, 65, 71– 76, 78–80, 88, 104, 145, 164, 165, 168, 171, 172, 189, 190, 192, 196, 198, 206– 208, 217, 218, 234, 235, 240, 243, 319, 341
<code>\orgromannumeral</code>	177, 178	<code>\PackageInfo</code>	5, 166, 175
<code>\orgthe</code>	177	<code>\PackageWarning</code>	5, 16, 333
<code>\outputpenalty</code>	278, 279	<code>\PackageWarningNoLine</code>	5, 17, 33, 371, 372
P			
<code>\p@</code>	265, 285, 303	<code>\pagegoal</code>	279
<code>\p@glshyp@opt</code>	105	<code>\pagelistname</code>	33, 272– 274, 276, 277, 283–285, 294–297, 301–303
package options:		<code>\par</code>	34, 204, 265, 267, 285, 287–290, 303, 304, 306–312, 321, 326–328
<code>acronym</code>	13, 14, 30, 182, 211	<code>\parindent</code>	285–290, 304–307, 309–312, 326–328
<code>true</code>	14	<code>\parskip</code>	285–289, 304, 305, 307
<code>counter</code>	16	<code>\PassOptionsToPackage</code>	254, 333
<code>description</code>	237, 238	<code>\penalty</code>	278
<code>dua</code>	236–238	<code>\phantomsection</code>	39
<code>entrycounter</code>	197, 199	<code>polyglossia package</code>	22, 32
<code>true</code>	10	<code>\printglossaries</code>	168
<code>footnote</code>	119–124, 233, 236, 237, 239	<code>\printglossary</code>	14, 17, 27, 28, 168, 183, 198
<code>hyperfirst</code>		<code>\printglossary options</code>	
<code>false</code>	119–124	<code>entrycounter</code>	196
<code>indexonlyfirst</code>	380	<code>nogroupskip</code>	196
<code>makeindex</code>	158, 253	<code>nonumberlist</code>	198
<code>nogroupskip</code>	270, 271, 273, 275–277, 280, 282, 284, 292, 293, 295, 298, 300, 302	<code>nopostdot</code>	196
<code>nolist</code>	247	<code>numberedsection</code>	196
<code>nolong</code>	247, 269	<code>style</code>	196
<code>nomain</code>	13	<code>subentrycounter</code>	197
<code>nonumberlist</code>	8	<code>title</code>	195
<code>nosuper</code>	247	<code>toctitle</code>	195
<code>notree</code>	247	<code>type</code>	13, 182, 195
<code>nowarn</code>	5	<code>\printindex</code>	28
<code>numberline</code>	6	<code>\printnoidxglossaries</code>	170
<code>sanitize</code>	20, 60, 145, 146		
<code>sanitizesort</code>	17		
<code>savewrites</code>	26, 377		
<code>false</code>	165		

<code>\printnoidxglossary</code>	<code>\renewenvironment</code>
..... 169, 170, 173, 183, 189, 190, 198	201, 263, 266, 269–275, 277, 278, 280–305, 307, 309
<code>\printnoidxglossary options</code>	<code>\RequireGlossariesLang</code>
<code>sort</code>	33, 371, 372
..... 198	<code>\RequirePackage</code>
<code>\printnumbers</code> 4, 8, 9, 22, 23, 28, 32, 247, 253,
27	254, 269, 275, 279, 285, 291, 297, 334, 371
<code>\printsymbols</code>	<code>\restorecounters@</code>
27	109
<code>\ProcessOptions</code>	<code>\romannumeral</code>
254, 333	176–178, 309, 310, 327
<code>\ProcessOptionsX</code>	
28	S
<code>\protect</code>	<code>\s@glshyp@opt</code>
40, 102, 218–220, 226, 232, 236, 238, 350, 354, 355, 360, 361	105
<code>\protected@edef</code>	<code>\s@newglossary</code>
7, 42, 44, 47, 49, 79, 82, 84, 94–96, 102, 109, 152, 175, 178, 196, 202, 203, 206, 207, 216, 240, 246, 255, 261, 314, 315, 333, 334, 339	57
<code>\protected@write</code>	<code>\savecounters@</code>
56, 57, 156–158, 167, 170, 173, 175, 182, 261, 315	109
<code>\protected@xdef</code>	<code>\seename</code>
... 11, 12, 15, 19, 66, 84, 85, 178, 336, 337	181
<code>\providecommand</code>	<code>\SetAcronymStyle</code>
..... 14, 30, 31, 38, 56, 90, 118, 158, 167, 170, 185, 186, 202, 203, 265, 285, 303	24
<code>\ProvidesFile</code>	<code>\setbool</code>
31	21
<code>\ProvidesPackage</code>	<code>\setbox</code>
..... 4, 254, 261, 263, 265, 269, 275, 279, 285, 291, 297, 303, 313, 319, 333, 371	279
	<code>\setcounter</code>
R	197, 199, 200
<code>\r</code>	<code>\SetCustomDisplayStyle</code>
19	246, 247
<code>\raggedright</code>	<code>\SetDefaultAcronymDisplayStyle</code> ...
277, 278, 280–285, 298–303	231
<code>\raisebox</code>	<code>\SetDefaultAcronymStyle</code>
117	245
<code>\ref</code>	<code>\SetDescriptionAcronymDisplayStyle</code>
200 237, 238
<code>\refstepcounter</code>	<code>\SetDescriptionAcronymStyle</code>
197, 200	245
<code>\relax</code>	<code>\SetDescriptionDUAAcronymDisplayStyle</code>
7, 9, 12–14, 22, 23, 29, 43, 56, 61, 62, 66, 79, 82, 89, 90, 105, 106, 109, 110, 112–116, 146, 159–162, 164, 165, 168, 170, 172, 176, 178–181, 183, 185, 192, 193, 195, 196, 206, 207, 247, 262, 265, 278, 285, 289, 290, 303, 304, 306–312, 314, 317–319, 325–328, 336, 351, 352, 364–366, 368 235
<code>\renewacronymstyle</code> .	<code>\SetDescriptionDUAAcronymStyle</code> ...
354–358, 360, 362, 363	245
<code>\renewcommand</code>	<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>
4–10, 13, 14, 16, 17, 21, 23, 24, 26, 29, 32–36, 48, 59, 63, 76, 88–90, 152, 154–156, 162, 163, 168–172, 174, 184, 186, 196–198, 216–228, 231, 233–235, 237–240, 242–244, 246, 263, 264, 266–277, 279–295, 298–302, 304–314, 319–327, 329–332, 336, 341, 345, 348, 350, 353–357, 359–367 233, 234
	<code>\SetDescriptionFootnoteAcronymStyle</code> 245
	<code>\SetDUADisplayStyle</code>
	244, 245
	<code>\SetDUAStyle</code>
	245
	<code>\setentrycounter</code>
	42, 158, 195, 313
	<code>\SetFootnoteAcronymDisplayStyle</code> ...
	240
	<code>\SetFootnoteAcronymStyle</code>
	245
	<code>\SetGenericNewAcronym</code>
	217
	<code>\setglossarystyle</code>
 183, 207, 247, 266–278, 281–290, 292–297, 299–303, 305, 306, 308, 311
	<code>\setglossentrycompatibility</code> ...
	196, 207
	<code>\setkeys</code> ..
	21, 26, 29, 38, 78, 108, 154, 184, 216, 231, 233, 235, 237, 240, 242, 244, 246
	<code>\setlength</code>
	268, 269, 280, 285–289, 291, 298, 304, 305, 307, 310, 311, 327, 328
	<code>\SetSmallAcronymDisplayStyle</code> ..
	242, 243
	<code>\SetSmallAcronymStyle</code>
	245
	<code>\settoheight</code>
	117
	<code>\settowidth</code>
	220, 308–310, 327
	<code>\sfcode</code>
	9

\show	247–253, 369, 370	\textsmaller	221, 227, 234, 238, 240, 243, 362
\SmallNewAcronymDef	243	\textttt	210
\space	6, 26, 30, 42, 46, 49, 62, 64, 88, 91, 92, 102, 103, 105, 152, 157–160, 164, 166–168, 170, 172, 173, 181, 183, 186, 197, 198, 200, 201, 207, 213, 218–228, 236, 240, 241, 263, 265–267, 269, 270, 280, 291, 292, 298, 304–311, 313, 314, 316–318, 320– 323, 325–329, 331, 350, 354–357, 359–364	\textulc	212
\spacefactor	9	\textup	210, 212
\SS	20	\th	20
\ss	20	\the	30, 32, 42, 47, 49, 57, 112–117, 156, 160, 162, 164, 174, 175, 177, 182, 193, 202, 203, 209, 216–220, 224, 226, 230, 232–234, 236, 237, 239, 241–244, 246, 313, 315, 318, 333, 334, 354, 355, 359, 360, 363–368
\string	6, 17, 26, 30, 40, 42, 44–47, 49, 56, 57, 62, 64, 68, 69, 72, 73, 83, 84, 88, 90–92, 103, 105, 110, 112–114, 116, 152, 155–162, 164, 165, 167–173, 179, 180, 183, 185, 186, 189, 190, 198, 204, 207, 261, 313–319	\the@numberlist	152, 153
\strut	204, 266–268, 270, 271, 273, 280, 282, 283, 292, 293, 295, 298, 300, 302, 307, 320–324, 326, 329–332	\theglossary	5
\subglossentry	83, 184, 193, 203, 204, 264, 266– 268, 270, 271, 273, 280, 282, 283, 291, 293, 295, 298, 300, 302, 304, 306, 307, 310	\theglossaryentry	197, 199, 200
\subitem	304, 325	\theglossarysubentry	198, 199, 201
\subsubitem	305, 325	\theglentrycounter	108, 109, 178, 314, 315
supertabular package	8, 247, 291, 297	\theH	180
\symbolname	33, 273, 274, 276, 284, 285, 295–297, 302, 303	\theHglossaryentry	197, 199
		\theHglossarysubentry	198, 199
		\theHglentrycounter	109, 178
		\thesection	29
		\this@dialect	33, 371, 372
		\toks@	30, 32, 42, 47, 49, 57, 112–117, 162, 164, 182, 202, 203, 209, 313, 333, 334
		\toprule	275, 276
		tracklang package	32, 371
		\trans@languages	32
		\translate	32, 33
		\translatelet	13, 14, 27, 28
		translator package	13, 14, 22, 27, 28, 32, 33, 182
		U	
		\u	19
		\uccode	192
		\undef	63, 182
		\unskip	76, 268, 321
		\unvbox	279
		\usedictionary	32
		\usepackage	189, 190
		V	
		\v	19
		\val	6, 7, 22, 62, 196
		\vbox	279
		\vsize	279
		\vskip	265, 278, 285, 303
		\vss	279
		W	
		\warn@nomakeglossaries	168–170
T			
\t	19		
\thead	291–303		
\tfoot	291–303		
\tabularnewline	269– 277, 280–285, 291–303, 323, 324, 329, 330		
\texorpdfstring	148		
\textbar	263		
\textbf	204, 210, 303, 325–328		
textcase package	4		
\textit	210		
\textmd	210		
\textrm	210		
\textsc	210, 220, 221, 227, 234, 238, 240, 243, 362		
\textsf	210		
\textsl	210		

<code>\warn@noprintglossary</code>	168–170, 185	<code>xindy</code>	10,
<code>\write</code>	68, 69, 90, 156–		25, 26, 34, 35, 40, 43, 45, 47–49, 84, 116,
	161, 166, 167, 170, 174, 185, 186, 315–319		155–157, 174, 178, 179, 185, 204, 253, 314
<code>\writeist</code>	165, 319	<code>\xmakefirstuc</code>	94–96, 102, 145, 146, 255
	X	<code>\xspace</code>	211
<code>\x</code>	209	<code>xspace package</code>	4, 211
<code>\xatlevel@</code>	109		Y
<code>\xcapitalisewords</code>	148	<code>\year</code>	156, 160, 315, 318
<code>\xdef</code>	73, 79, 80, 82, 184, 262		Z
<code>\xglsaccsupp</code>	339	<code>\z@</code>	279
<code>\xifinlistcs</code>	186–188, 191		
<code>xindy</code>	373		