Documented Code For glossaries v4.40

Nicola L.C. Talbot

Dickimaw Books
http://www.dickimaw-books.com/

2018-06-01

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 v4.40: LaTeX2e 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 . . . 34
- 1.4 Xindy . . . 44
- 1.5 Loops and conditionals . . . 53
- 1.6 Defining new glossaries . . . 59
- 1.7 Defining new entries . . . 64
- 1.8 Resetting and unsetting entry flags . . . 90
- 1.9 Keeping Track of How Many Times an Entry Has Been Unset . . . 93
- 1.10 Loading files containing glossary entries . . . 98
- 1.11 Using glossary entries in the text . . . 98
- 1.12 Adding an entry to the glossary without generating text . . . 157
- 1.13 Creating associated files . . . 159
- 1.14 Writing information to associated files . . . 178
- 1.15 Glossary Entry Cross-References . . . 187
- 1.16 Displaying the glossary . . . 189
- 1.17 Acronyms . . . 217
- 1.18 Predefined acronym styles . . . 222
- 1.19 Predefined Glossary Styles . . . 253
- 1.20 Debugging Commands . . . 254
- 1.21 Compatibility with version 2.07 and below . . . 259

2 Prefix Support (glossaries-prefix Code) 261

3 Glossary Styles 268
- 3.1 Glossary hyper-navigation definitions (glossary-hypernav package) . . . 268
- 3.2 In-line Style (glossary-inline.sty) . . . 270
- 3.3 List Style (glossary-list.sty) . . . 273
- 3.4 Glossary Styles using longtable (the glossary-long package) . . . 276
- 3.5 Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package . . . 282
- 3.6 Glossary Styles using longtable (the glossary-longragged package) . . . 287
- 3.7 Glossary Styles using multicol (glossary-mcols.sty) . . . 292
- 3.8 Glossary Styles using supertabular environment (glossary-super package) . . . 298
- 3.9 Glossary Styles using supertabular environment (glossary-superragged package) 305
- 3.10 Tree Styles (glossary-tree.sty) . . . 311

4 Backwards Compatibility 321
4.1 glossaries-compatible-207 321
4.2 glossaries-compatible-307 327

5 Accessibility Support (glossaries-accsupp Code) 341
5.1 Defining Replacement Text 342
5.2 Accessing Replacement Text 345
5.3 Displaying the Glossary 361
5.4 Acronyms 362
5.5 Debugging Commands 377

6 Multi-Lingual Support 379
6.1 Polyglossia Captions 379

Glossary 381

Change History 382

Index 406

1 Main Package Code

1.1 Package Definition

This package requires $\LaTeX 2_{\varepsilon}$.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries}[2018/06/01 v4.40 (NLCT)]
```

Required packages:

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

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

```
\RequirePackage{textcase}
\renewcommand*{\mfirstucMakeUppercase}{\MakeTextUppercase}%
```

```
\RequirePackage{xfor}
```

```
\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.)

```
\RequirePackage{amsgen}
```

As from v3.0, now loading etoolbox:

```
\RequirePackage{etoolbox}
```

Check if doc has been loaded.

`f@gls@docloaded`

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

```
\@ifpackageloaded{doc}%
{%
  \@gls@docloadedtrue
}%
{%
  \@ifclassloaded{nlctdoc}{\@gls@docloadedtrue}{\@gls@docloadedfalse}%
}
\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

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

@endtheglossary

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

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

```
\let\glsorg@PrintChanges\PrintChanges
\renewcommand{\PrintChanges}{%
  \begingroup
    \let\theglossary\glsorg@theglossary
    \let\endtheglossary\glsorg@endtheglossary
    \glsorg@PrintChanges
  \endgroup
}
```

End of doc stuff.

```
\fi
```

1.2 Package Options

debug Switch on debug mode. This will also cancel the nowarn option. This is now a choice key.

```
\newif\if@gls@debug
\define@choicekey{glossaries.sty}{debug}[\val\nr]{true,false,showtargets}[true]{%
  \ifcase\nr\relax
    \@gls@debugtrue
    \renewcommand*{\GlossariesWarning}[1]{%
      \PackageWarning{glossaries}{##1}%
    }%
    \renewcommand*{\GlossariesWarningNoLine}[1]{%
      \PackageWarningNoLine{glossaries}{##1}%
    }%
    \let\@glsshowtarget\@gobble
    \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
  \or
    \@gls@debugfalse
    \let\@glsshowtarget\@gobble
    \PackageInfo{glossaries}{debug mode OFF}%
  \or
    \@gls@debugtrue
    \renewcommand*{\GlossariesWarning}[1]{%
      \PackageWarning{glossaries}{##1}%
    }%
    \renewcommand*{\GlossariesWarningNoLine}[1]{%
      \PackageWarningNoLine{glossaries}{##1}%
    }%
    \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
    \renewcommand{\@glsshowtarget}{\glsshowtarget}%
  \fi
}
```

`\glsshowtarget` If debug=showtargets, show the hyperlink target name in the margin.

```
\newcommand*{\glsshowtarget}[1]{%
 \ifmmode
   \nfss@text{\ttfamily\small [#1]}%
 \else
   \ifinner
     \texttt{\small [#1]}%
   \else
     \marginpar{\texttt{\small #1}}%
   \fi
 \fi
}
```

`\@glsshowtarget` debug=showtargets will redefine this.

```
\newcommand*{\@glsshowtarget}[1]{}
```

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

`gls@see@noindex`

```
\newcommand*{\@gls@see@noindex}{%
  \PackageError{glossaries}%
  {`\gls@xr@key' key may only be used after \string\makeglossaries\space
   or \string\makenoidxglossaries\space (or move
   \string\newglossaryentry\space
   definitions into the preamble)}%
  {You must use \string\makeglossaries\space
   or \string\makenoidxglossaries\space before defining
   any entries that have a `\gls@xr@key' key. It may
   be that the `see' key has been written to the .glsdefs
   file from the previous run, in which case you need to
   move your definitions
   to the preamble if you don't want to use
   \string\makeglossaries\space
   or \string\makenoidxglossaries}%
}
```

`seenoindex`

```
\define@choicekey{glossaries.sty}{seenoindex}[\val\nr]{error,warn,ignore}{%
  \ifcase\nr
```

```
    \renewcommand*{\@gls@see@noindex}{%
      \PackageError{glossaries}%
      {`\gls@xr@key' key may only be used after \string\makeglossaries\space
       or \string\makenoidxglossaries}%
      {You must use \string\makeglossaries\space
       or \string\makenoidxglossaries\space before defining
       any entries that have a `\gls@xr@key' key}%
    }%
  \or
    \renewcommand*{\@gls@see@noindex}{%
      \GlossariesWarning{`\gls@xr@key' key ignored}%
    }%
  \or
    \renewcommand*{\@gls@see@noindex}{}%
  \fi
}
```

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.

```
\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.

```
\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`.

```
\ifcsundef{chapter}%
  {\newcommand*{\@@glossarysec}{section}}%
  {\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`.

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

Determine whether or not to use numbered sections.

glossarysecstar

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

lossaryseclabel

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

\glsautoprefix Prefix to add before label if automatically generated:

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

numberedsection

```
\define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
false,nolabel,autolabel,nameref}[nolabel]{%
  \ifcase\nr\relax
    \renewcommand*{\@@glossarysecstar}{*}%
    \renewcommand*{\@@glossaryseclabel}{}%
  \or
    \renewcommand*{\@@glossarysecstar}{}%
    \renewcommand*{\@@glossaryseclabel}{}%
  \or
    \renewcommand*{\@@glossarysecstar}{}%
    \renewcommand*{\@@glossaryseclabel}{%
      \label{\glsautoprefix\@glo@type}}%
  \or
    \renewcommand*{\@@glossarysecstar}{*}%
    \renewcommand*{\@@glossaryseclabel}{%
      \protected@edef\@currentlabelname{\glossarytoctitle}%
      \label{\glsautoprefix\@glo@type}}%
  \fi
}
```

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.) Note that the list style is incompatible with classicthesis so change the default to index if that package has been loaded.

y@default@style

```
\@ifpackageloaded{classicthesis}
{\newcommand*{\@glossary@default@style}{index}}
{\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. This now uses `\def` instead of `\renewcommand` as `\@glossary@default@style` may have been set to `\relax`.

```
\define@key{glossaries.sty}{style}{%
  \def\@glossary@default@style{#1}%
}
```

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

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

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`

```
\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 ignores its argument).

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

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

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

`eautonumberlist`

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

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

```
\@gls@declareoption{seeautonumberlist}{%
   \renewcommand*{\@glo@seeautonumberlist}{%
      \def\@glo@prefix{\glsnextpages}%
   }%
}
```

`esclocations` When using `makeindex` or `xindy`, the locations may need to be adjusted to ensure they're in a format that's allowed by the indexing application. This involves a bit of hackery and isn't needed if the locations are all guaranteed to be in the correct form (or if the user is prepared to post-process the glossary file before calling the relevant indexing application) so esclocations=false will switch off this mechanism allowing for a faster and more stable approach.

```
\define@boolkey{glossaries.sty}[gls]{esclocations}[true]{}
\glsesclocationstrue
```

`\@gls@loadlong`

```
\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.

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

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

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

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

\@gls@loadlist

```
\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. If the style is still set to list, the default must be set to \relax.

```
\@gls@declareoption{nolist}{%
 \renewcommand*{\@gls@loadlist}{%
  \ifdefstring{\@glossary@default@style}{list}%
  {\let\@glossary@default@style\relax}%
  {}%
 }%
}
```

\@gls@loadtree

```
\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.

```
\@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).

```
\@gls@declareoption{nostyles}{%
  \renewcommand*{\@gls@loadlong}{}%
  \renewcommand*{\@gls@loadsuper}{}%
  \renewcommand*{\@gls@loadlist}{}%
  \renewcommand*{\@gls@loadtree}{}%
  \let\@glossary@default@style\relax
}
```

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.)

```
\newcommand*{\glspostdescription}{%
  \ifglsnopostdot\else.\spacefactor\sfcode`\. \fi
}
```

nopostdot Boolean option to suppress post description dot

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

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

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

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

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

```
\@ifclassloaded{memoir}
{%
  \glsucmarktrue
}%
{%
  \glsucmarkfalse
}
```

glossaryentry If the entrycounter package option has been used, define a counter to number each level 0 entry. This is now defined by an internal command for consistency.

aryentrycounter

```
\newcommand*{\@gls@define@glossaryentrycounter}{%
  \ifglsentrycounter
```

Define the glossaryentry counter if it doesn't already exist.

```
    \ifundef\c@glossaryentry
    {%
      \ifx\@gls@counterwithin\@empty
        \newcounter{glossaryentry}%
      \else
        \newcounter{glossaryentry}[\@gls@counterwithin]%
      \fi
      \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
    }%
    {}%
  \fi
}
```

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.

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

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

```
\define@key{glossaries.sty}{counterwithin}{%
  \renewcommand*{\@gls@counterwithin}{#1}%
  \glsentrycountertrue
  \@gls@define@glossaryentrycounter
}
```

s@counterwithin The default value is no parent counter:

```
\newcommand*{\@gls@counterwithin}{}
```

lossarysubentry If the subentrycounter package option has been used, define a counter to number each level 1 entry. This is now defined by an internal command for consistency.

subentrycounter

```
\newcommand{\@gls@define@glossarysubentrycounter}{%
```

Check if counter already defined.

```
  \ifundef\c@glossarysubentry
  {%
    \ifglssubentrycounter
      \ifglsentrycounter
        \newcounter{glossarysubentry}[glossaryentry]%
      \else
        \newcounter{glossarysubentry}%
      \fi
```

As with `\theHglossaryentry`, this starts with `\currentglossary.` to help avoid duplicate hyper targets.

```
      \def\theHglossarysubentry{\currentglossary.\currentglssubentry.\theglossarysubentry}%
    \fi
  }%
  {}%
}
```

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.

```
\define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
\glssubentrycounterfalse
```

efault@sorttype Initialise default sort for `\printnoidxglossary`

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

sort Define the sort method: sort=standard (default), sort=def (order of definition) or sort=use (order of use). If no indexing required, use sort=none.

```
\define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
  \renewcommand*{\@glo@default@sorttype}{#1}%
  \csname @gls@setupsort@#1\endcsname
}
```

sprestandardsort

```
\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.

```
\newcommand*{\glsprestandardsort}[3]{%
  \glsdosanitizesort
}
```

eck@sortallowed

```
\newcommand*{\@glo@check@sortallowed}[1]{}
```

upsort@standard Set up the macros for default sorting.

```
\newcommand*{\@gls@setupsort@standard}{%
```

Store entry information when it's defined.

```
  \def\do@glo@storeentry{\@glo@storeentry}%
```

No count register required for standard sort.

```
  \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.)

```
  \def\@gls@defsort##1##2{%
    \ifx\@glo@sort\@glsdefaultsort
      \let\@glo@sort\@glo@name
    \fi

    \let\glsdosanitizesort\@gls@sanitizesort
    \glsprestandardsort{\@glo@sort}{##1}{##2}%
    \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
  }%
```

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

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

This sort option is allowed with `\makeglossaries` and `\makenoidxglossaries`.

```
  \let\@glo@check@sortallowed\@gobble
}
```

Set standard sort as the default:

```
\@gls@setupsort@standard
```

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

```
\newcommand*\glssortnumberfmt[1]{%
  \ifnum#1<100000 0\fi
  \ifnum#1<10000 0\fi
  \ifnum#1<1000 0\fi
  \ifnum#1<100 0\fi
  \ifnum#1<10 0\fi
  \number#1%
}
```

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

```
\newcommand*{\@gls@setupsort@def}{%
```

Store entry information when it's defined.

```
  \def\do@glo@storeentry{\@glo@storeentry}%
```

Defined count register associated with the glossary.

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

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

```
  \def\@gls@defsort##1##2{%
```

It may be that the sort order was changed after the glossary was defined, so check if the count register has been defined.

```
    \ifcsundef{glossary@##1@sortcount}%
    {\@gls@defsortcount{##1}}%
    {}%
    \expandafter\global\expandafter
    \advance\csname glossary@##1@sortcount\endcsname by 1\relax
    \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
       \expandafter\glssortnumberfmt
         {\csname glossary@##1@sortcount\endcsname}}%
  }%
```

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

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

This sort option is allowed with `\makeglossaries` and `\makenoidxglossaries`.

```
  \let\@glo@check@sortallowed\@gobble
}
```

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

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

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

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

Defined count register associated with the glossary.

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

Initialise the sort key to empty.

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

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

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

Get the parent, if one exists

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

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

```
    \ifx\@glo@parent\@empty
    \else
      \expandafter\@gls@setsort\expandafter{\@glo@parent}%
    \fi
```

Set index information for this entry

```
    \edef\@glo@type{\csname glo@##1@type\endcsname}%
    \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
    \ifx\@gls@tmp\@empty
      \expandafter\global\expandafter
      \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
      \expandafter\protected@xdef\csname glo@##1@sort\endcsname{%
         \expandafter\glssortnumberfmt
           {\csname glossary@\@glo@type @sortcount\endcsname}}%
      \@glo@storeentry{##1}%
    \fi
  }%
```

This sort option is allowed with `\makeglossaries` and `\makenoidxglossaries`.

```
  \let\@glo@check@sortallowed\@gobble
}
```

`@setupsort@none` Slightly improves efficiency in the event that no indexing is required.

```
\newcommand*{\@gls@setupsort@none}{%
```

Don't store entry index information.

```
  \def\do@glo@storeentry##1{}%
```

No count register required for standard sort.

```
  \def\@gls@defsortcount##1{}%
```

Don't modify sort value.

```
  \def\@gls@defsort##1##2{%
    \expandafter\global\expandafter\let\csname glo@##2@sort\endcsname\@glo@sort
  }%
```

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

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

This sort option isn't allowed with `\makeglossaries` or `\makenoidxglossaries`.

```
  \renewcommand\@glo@check@sortallowed[1]{\PackageError{glossaries}
   {Option sort=none not allowed with \string##1}%
   {(Use sort=def instead)}}%
}
```

\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.)

```
\newcommand*{\glsdefmain}{%
  \if@gls@docloaded
    \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
  \else
    \newglossary{main}{gls}{glo}{\glossaryname}%
  \fi
```

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

```
  \newcommand*{\gls@tr@set@main@toctitle}{%
    \translatelet{\glossarytoctitle}{Glossary}%
  }%
}
```

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

```
\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

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

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

```
\@gls@declareoption{nomain}{%
  \let\glsdefaulttype\relax
  \renewcommand*{\glsdefmain}{}%
}
```

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.

```
\define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
  \ifglsacronym
    \renewcommand{\@gls@do@acronymsdef}{%
      \DeclareAcronymList{acronym}%
      \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
      \renewcommand*{\acronymtype}{acronym}%
```

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

```
      \newcommand*{\gls@tr@set@acronym@toctitle}{%
        \translatelet{\glossarytoctitle}{Acronyms}%
      }%
    }%
  \else
    \let\@gls@do@acronymsdef\relax
  \fi
}
```

\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.

```
\AtBeginDocument{%
  \ifglsacronym
    \ifbool{glscompatible-3.07}%
    {}%
    {%
      \providecommand*{\printacronyms}[1][]{%
        \printglossary[type=\acronymtype,#1]}%
    }%
  \fi
}
```

@do@acronymsdef Set default value

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

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

```
\@gls@declareoption{acronyms}{%
  \glsacronymtrue
  \renewcommand{\@gls@do@acronymsdef}{%
     \DeclareAcronymList{acronym}%
     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
     \renewcommand*{\acronymtype}{acronym}%
```

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

```
      \newcommand*{\gls@tr@set@acronym@toctitle}{%
        \translatelet{\glossarytoctitle}{Acronyms}%
      }%
   }%
}
```

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

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

dtoacronynlists

```
\newcommand*{\@addtoacronymlists}[1]{%
  \ifx\@glsacronymlists\@empty
```

```
    \protected@xdef\@glsacronymlists{#1}%
  \else
    \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
  \fi
}
```

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.)

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

IfListOfAcronyms `\glsIfListOfAcronyms{⟨label⟩}{⟨true part⟩}{⟨false part⟩}`

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

```
\newcommand{\glsIfListOfAcronyms}[1]{%
  \edef\@do@gls@islistofacronyms{%
    \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
  \@do@gls@islistofacronyms
}
```

Internal command requires label and list to be expanded:

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

Not found

```
    #4%
  \else
```

Found

```
    #3%
  \fi
}
```

lsisacronymlist Convenient boolean.

```
\newif\if@glsisacronymlist
```

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

```
\newcommand*{\gls@checkisacronymlist}[1]{%
   \glsIfListOfAcronyms{#1}%
     {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
}
```

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.)

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

acronymlists

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

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

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

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

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

gls@nohyperlist

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

lareNoHyperList

```
\newcommand*{\GlsDeclareNoHyperList}[1]{%
  \ifdefempty\@gls@nohyperlist
  {%
     \renewcommand*{\@gls@nohyperlist}{#1}%
  }%
  {%
     \appto\@gls@nohyperlist{,#1}%
  }%
}
```

nohypertypes

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

ossariesWarning Prints a warning message.

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

esWarningNoLine Prints a warning message without the line number.

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

tentrieswarning Warn user that sorting may take a long time. This is actually an informational message rather than a warning so just use `\typeout`.

```
\newcommand{\glosortentrieswarning}{%
 \typeout{Using TeX to sort glossary entries---this may
 take a while}%
}
```

nowarn Define package option to suppress warnings

```
\@gls@declareoption{nowarn}{%
  \if@gls@debug
    \GlossariesWarning{Warnings can't be suppressed in debug mode}%
  \else
    \renewcommand*{\GlossariesWarning}[1]{}%
    \renewcommand*{\GlossariesWarningNoLine}[1]{}%
    \renewcommand*{\glosortentrieswarning}{}%
    \renewcommand*{\@gls@missinglang@warn}[2]{}%
  \fi
}
```

issinglang@warn Missing language warning.

```
\newcommand*{\@gls@missinglang@warn}[2]{%
  \PackageWarningNoLine{glossaries}%
  {No language module detected for `#1'.\MessageBreak
  Language modules need to be installed separately.\MessageBreak
  Please check on CTAN for a bundle called\MessageBreak
  `glossaries-#2' or similar}%
}
```

nolangwarn Suppress warning if language support not found.

```
\@gls@declareoption{nolangwarn}{%
  \renewcommand*{\@gls@missinglang@warn}[2]{}%
}
```

nonglossdefined Issue a warning if overriding `\printglossary`

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

theglossdefined Issue a warning if overriding theglossary

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

noredefwarn

Suppress warning on redefinition of `\printglossary`

```
\@gls@declareoption{noredefwarn}{%
  \renewcommand*{\@gls@warnonglossdefined}{}%
  \renewcommand*{\@gls@warnontheglossdefined}{}%
}
```

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

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

lssetexpandfield

`\glssetexpandfield{⟨field⟩}`

Sets field to always expand.

```
\newcommand*{\glssetexpandfield}[1]{%
  \csdef{gls@assign@#1@field}##1##2{%
    \@@gls@expand@field{##1}{#1}{##2}%
  }%
}
```

setnoexpandfield

`\glssetnoexpandfield{⟨field⟩}`

Sets field to never expand.

```
\newcommand*{\glssetnoexpandfield}[1]{%
  \csdef{gls@assign@#1@field}##1##2{%
    \@@gls@noexpand@field{##1}{#1}{##2}%
  }%
}
```

sign@type@field

The type must always be expandable.

```
\glssetexpandfield{type}
```

sign@desc@field

The description is not expanded by default:

```
\glssetnoexpandfield{desc}
```

escplural@field

```
\glssetnoexpandfield{descplural}
```

ls@sanitizename

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

sign@name@field

Don’t expand name by default.

```
\glssetnoexpandfield{name}
```

@sanitizesymbol

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

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

```
\glssetnoexpandfield{symbol}
```

bolplural@field

```
\glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```
\newcommand*{\@gls@sanitizesort}{%
  \ifglssanitizesort
    \@@gls@sanitizesort
  \else
    \@@gls@nosanitizesort
  \fi
}
```

ls@sanitizesort

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

@nosanitizesort

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

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

```
\newcommand*\@gls@noidx@sanitizesort{%
  \ifdefvoid\@glo@sort
  {}%
  {%
    \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
  }%
}
\def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
  \def\@glo@sort{#1#2}%
  \@onelevel@sanitize\@glo@sort
}
```

@nosanitizesort

```
\newcommand*{\@@gls@noidx@nosanitizesort}{%
  \ifdefvoid\@glo@sort
  {}%
  {%
    \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
  }%
}
\def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
  \bgroup
    \glsnoidxstripaccents
    \protected@xdef\@@glo@sort{#1#2}%
  \egroup
  \let\@glo@sort\@@glo@sort
}
```

idxstripaccents This strips accents by redefining the standard accent commands to just do their argument. (This will be localised since `\glsnoidxstripaccents` is used within a group.) Anything outside this standard set really shouldn't be using `\makenoidxglossaries`.

```
\newcommand*\glsnoidxstripaccents{%
  \let\IeC\@firstofone
  \let\'\@firstofone
  \let\`\@firstofone
  \let\^\@firstofone
  \let\"\@firstofone
  \let\u\@firstofone
  \let\t\@firstofone
  \let\d\@firstofone
  \let\r\@firstofone
  \let\=\@firstofone
  \let\.\@firstofone
  \let\~\@firstofone
  \let\v\@firstofone
  \let\H\@firstofone
  \let\c\@firstofone
  \let\b\@firstofone

  \let\a\@secondoftwo
  \def\AE{AE}%
  \def\ae{ae}%
  \def\OE{OE}%
  \def\oe{oe}%
  \def\AA{AA}%
  \def\aa{aa}%
  \def\L{L}%
  \def\l{l}%
  \def\O{O}%
  \def\o{o}%
  \def\SS{SS}%
  \def\ss{ss}%
  \def\th{th}%

  \def\TH{TH}%
  \def\dh{dh}%
  \def\DH{DH}%
}
```

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`.

```
\define@boolkey[gls]{sanitize}{description}[true]{%
  \GlossariesWarning{sanitize={description} package option deprecated}%
  \ifgls@sanitize@description
    \glssetnoexpandfield{desc}%
    \glssetnoexpandfield{descplural}%
  \else
    \glssetexpandfield{desc}%
    \glssetexpandfield{descplural}%
  \fi
}
```

```
\define@boolkey[gls]{sanitize}{name}[true]{%
  \GlossariesWarning{sanitize={name} package option deprecated}%
  \ifgls@sanitize@name
    \glssetnoexpandfield{name}%
  \else
    \glssetexpandfield{name}%
  \fi
}
```

```
\define@boolkey[gls]{sanitize}{symbol}[true]{%
  \GlossariesWarning{sanitize={symbol} package option deprecated}%
  \ifgls@sanitize@symbol
    \glssetnoexpandfield{symbol}%
    \glssetnoexpandfield{symbolplural}%
  \else
    \glssetexpandfield{symbol}%
    \glssetexpandfield{symbolplural}%
  \fi
}
```

`sanitizesort`

```
\define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
  \ifglssanitizesort
    \glssetnoexpandfield{sortvalue}%
    \renewcommand*{\@gls@noidx@setsanitizesort}{%
      \glssanitizesorttrue
      \glssetnoexpandfield{sortvalue}%
    }%
  \else
    \glssetexpandfield{sortvalue}%
    \renewcommand*{\@gls@noidx@setsanitizesort}{%
      \glssanitizesortfalse
      \glssetexpandfield{sortvalue}%
    }%
  \fi
}
```

Default setting:

```
\glssanitizesorttrue
\glssetnoexpandfield{sortvalue}%
```

setsanitizesort Default behaviour for `\makenoidxglossaries` is sanitizesort=false.

```
\newcommand*{\@gls@noidx@setsanitizesort}{%
  \glssanitizesortfalse
  \glssetexpandfield{sortvalue}%
}
```

```
\define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
  \setbool{glssanitizesort}{#1}%
  \ifglssanitizesort
    \glssetnoexpandfield{sortvalue}%
  \else
    \glssetexpandfield{sortvalue}%
  \fi
  \GlossariesWarning{sanitize={sort} package option
    deprecated. Use sanitizesort instead}%
}
```

sanitize

```
\define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
  \ifthenelse{\equal{#1}{none}}%
  {%
    \GlossariesWarning{sanitize package option deprecated}%
    \glssetexpandfield{name}%
    \glssetexpandfield{symbol}%
    \glssetexpandfield{symbolplural}%
    \glssetexpandfield{desc}%
    \glssetexpandfield{descplural}%
  }%
  {%
    \setkeys[gls]{sanitize}{#1}%
  }%
}
```

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

```
\newif\ifglstranslate
```

otranslatorhook `\@gls@notranslatorhook` has been removed.

s@usetranslator

```
\newcommand*\@gls@usetranslator{%
```

polyglossia tricks `\@ifpackageloaded` into thinking that babel has been loaded, so check for polyglossia as well.

```
  \@ifpackageloaded{polyglossia}%
  {%
     \let\glsifusetranslator\@secondoftwo
  }%
  {%
    \@ifpackageloaded{babel}%
    {%
        \IfFileExists{translator.sty}%
        {%
           \RequirePackage{translator}%
           \let\glsifusetranslator\@firstoftwo
        }%
        {}%
    }%
    {}%
  }%
}
```

dtranslatordict Checks if given translator dictionary has been loaded.

```
\newcommand{\glsifusedtranslatordict}[3]{%
  \glsifusetranslator
  {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
  {#3}%
}
```

notranslate Provide a synonym for translate=false that can be passed via the document class.

```
\@gls@declareoption{notranslate}{%
  \glstranslatefalse
  \let\@gls@usetranslator\relax
  \let\glsifusetranslator\@secondoftwo
}
```

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

```
\define@choicekey{glossaries.sty}{translate}[\val\nr]%
  {true,false,babel}[true]%
  {%
    \ifcase\nr\relax
      \glstranslatetrue
      \renewcommand*\@gls@usetranslator{%
        \@ifpackageloaded{polyglossia}%
        {%
           \let\glsifusetranslator\@secondoftwo
        }%
        {%
          \@ifpackageloaded{babel}%
          {%
            \IfFileExists{translator.sty}%
            {%
               \RequirePackage{translator}%
               \let\glsifusetranslator\@firstoftwo
            }%
            {}%
          }%
          {}%
        }%
      }%
    \or
      \glstranslatefalse
      \let\@gls@usetranslator\relax
      \let\glsifusetranslator\@secondoftwo
    \or
      \glstranslatetrue
      \let\@gls@usetranslator\relax
      \let\glsifusetranslator\@secondoftwo
    \fi
  }
```

Set the default value:

```
\glstranslatefalse
\let\glsifusetranslator\@secondoftwo
\@ifpackageloaded{translator}%
{%
  \glstranslatetrue
  \let\glsifusetranslator\@firstoftwo
}%
{%
  \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
  {
    \@ifpackageloaded{\gls@thissty}%
    {%
      \glstranslatetrue
      \@endfortrue
    }%
    {}%
  }
}
```

`indexonlyfirst` Set whether to only index on first use.

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

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

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

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

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

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

```
\define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
  \ifbool{glsacrdescription}%
  {}%
  {%
    \renewcommand*{\@gls@sanitizedesc}{}%
  }%
  \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
}
```

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

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

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

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

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

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

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

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

`shotcuts` Define acronym shortcuts.

```
\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.

```
\newcommand*{\glsorder}{word}
```

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

```
\newcommand*{\@glsorder}[1]{}
```

`order`

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

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

```
\newif\ifglsxindy
```

The default is makeindex:

```
\glsxindyfalse
```

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

```
\@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.

```
\define@boolkey[gls]{xindy}{glsnumbers}[true]{}
\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.)

```
\def\@xdy@main@language{\languagename}%
```

Define key to set the language

```
\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.

```
\ifcsundef{inputencodingname}{%
  \def\gls@codepage{}}{%
  \def\gls@codepage{\inputencodingname}
}
```

Define a key to set the code page.

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

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

```
\define@key{glossaries.sty}{xindy}[]{%
  \glsxindytrue
  \setkeys[gls]{xindy}{#1}%
}
```

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

```
\@gls@declareoption{xindygloss}{%
  \glsxindytrue
}
```

ndynoglsnumbers Provide a synonym for xindy=glsnumbers=false that can be passed via the document class options.

```
\@gls@declareoption{xindynoglsnumbers}{%
  \glsxindytrue
  \gls@xindy@glsnumbersfalse
}
```

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

```
\define@boolkey{glossaries.sty}[gls]{automake}[true]{%
  \ifglsautomake
    \renewcommand*{\@gls@doautomake}{%
      \PackageError{glossaries}{You must use
      \string\makeglossaries\space with automake=true}
      {%
         Either remove the automake=true setting or
         add \string\makeglossaries\space to your document preamble.%
      }%
    }%
  \else
    \renewcommand*{\@gls@doautomake}{}%
  \fi
}
\glsautomakefalse
```

@gls@doautomake

```
\newcommand*{\@gls@doautomake}{}
\AtEndDocument{\@gls@doautomake}
```

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

```
\define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
  \ifglssavewrites
    \renewcommand*{\glswritefiles}{\@glswritefiles}%
  \else
    \let\glswritefiles\@empty
  \fi
}
```

Set default:

```
\glssavewritesfalse
\let\glswritefiles\@empty
```

compatible-3.07

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

compatible-2.07

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

Also set 3.07 compatibility if this option is set.

```
  \ifbool{glscompatible-2.07}%
  {%
    \booltrue{glscompatible-3.07}%
  }%
  {}%
}
\boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

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

Default is not to define the symbols glossary:

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

@gls@symbolsdef

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

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

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

numbers Create a “symbols” glossary type

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

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

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

index Create an “index” glossary type

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

Default is not to define index glossary:

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

\@gls@indexdef `\indexname` isn’t set by glossaries.

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

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

```
\@for\CurrentOption :=\@declaredoptions\do{%
  \ifx\CurrentOption\@empty
  \else
    \@expandtwoargs
      \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
    \ifin@
      \@use@ption
      \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
    \fi
  \fi
}
```

Now process options passed to the package:

```
\ProcessOptionsX
```

Load backward compatibility stuff:

```
\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:

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

Now define `\setupglossaries`:

```
\newcommand*{\setupglossaries}[1]{%
  \renewcommand*{\@gls@setacrstyle}{}%
  \ifglsacrshortcuts
    \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
  \else
    \def\@gls@setupshortcuts{%
      \ifglsacrshortcuts
        \DefineAcronymSynonyms
      \fi
    }%
  \fi
  \glsacrshortcutsfalse
  \let\@gls@do@numbersdef\relax
  \let\@gls@do@symbolssdef\relax
  \let\@gls@do@indexdef\relax
  \let\@gls@do@acronymsdef\relax
  \ifglsentrycounter
    \let\@gls@doentrycounterdef\relax
  \else
    \let\@gls@doentrycounterdef\@gls@define@glossaryentrycounter
  \fi
  \ifglssubentrycounter
    \let\@gls@dosubentrycounterdef\relax
  \else
    \let\@gls@dosubentrycounterdef\@gls@define@glossarysubentrycounter
  \fi
  \setkeys{glossaries.sty}{#1}%
  \@gls@setacrstyle
  \@gls@setupshortcuts
  \@gls@do@acronymsdef
  \@gls@do@numbersdef
  \@gls@do@symbolssdef
  \@gls@do@indexdef
  \@gls@doentrycounterdef
  \@gls@dosubentrycounterdef
}
```

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}`).

```
\ifthenelse{\equal{\glscounter}{section}}%
{%
  \ifcsundef{chapter}{}%
  {%
    \let\@gls@old@chapter\@chapter
    \def\@chapter[#1]#2{\@gls@old@chapter[{#1}]{#2}%
    \ifcsundef{hyperdef}{}{\hyperdef{section}{\thesection}{}}}%
  }%
}%
{}
```

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

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

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

```
\newcommand*{\@onlypremakeg}[1]{%
  \ifx\@gls@onlypremakeg\@empty
     \def\@gls@onlypremakeg{#1}%
  \else
     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
```

```
      \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
  \fi
}
```

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

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

sable@premakecs Disables the given command.

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

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

```
\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

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

\glssettoctitle Sets the TOC title for the given glossary.

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

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

\entryname

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

descriptionname

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

\symbolname

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

\pagelistname

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

Labels for makeindex's symbol and number groups:

ymbolsgroupname

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

umbersgroupname

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

glspluralsuffix The default plural is formed by appending \glspluralsuffix to the singular form.

```
\newcommand*{\glspluralsuffix}{s}
```

acrpluralsuffix Default plural suffix for acronyms

```
\newcommand*{\glsacrpluralsuffix}{\glspluralsuffix}
```

acrpluralsuffix

```
\newcommand*{\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}}
```

\seename

```
\providecommand*{\seename}{see}
```

\andname

```
\providecommand*{\andname}{\&}
```

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

eGlossariesLang

```
\newcommand*{\RequireGlossariesLang}[1]{%
  \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
}
```

sGlossariesLang

```
\newcommand*{\ProvidesGlossariesLang}[1]{%
  \ProvidesFile{glossaries-#1.ldf}%
}
```

ssarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
\ifglstranslate
```

Load tracklang

```
  \RequirePackage{tracklang}
```

Load translator if required.

```
\@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.)

```
\@ifpackageloaded{translator}
{%
```

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.

```
  \ifboolexpr
  {
    test {\ifdefstring{\trans@languages}{English}}
    and not
    test {\ifdefstring{bbl@loaded}{english}}
  }
  {%
    \let\glsifusetranslator\@secondoftwo
  }%
  {%
     \usedictionary{glossaries-dictionary}%
     \renewcommand*{\addglossarytocaptions}[1]{%
       \ifcsundef{captions#1}{}%
       {%
         \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
         \expandafter\toks@\expandafter{\@gls@tmp
           \renewcommand*{\glossaryname}{\translate{Glossary}}%
         }%
         \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
       }%
     }%
  }%
}%
{}%
```

Check for tracked languages

```
\AnyTrackedLanguages
{%
  \ForEachTrackedDialect{\this@dialect}{%
    \IfTrackedLanguageFileExists{\this@dialect}%
    {glossaries-}% prefix
    {.ldf}%
    {%
      \RequireGlossariesLang{\CurrentTrackedTag}%
    }%
    {%
      \@gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
      }%
    }%
  }%
  {}%
```

if using translator use translator interface.

```
  \glsifusetranslator
  {%
    \renewcommand*{\glssettoctitle}[1]{%
      \ifcsdef{gls@tr@set@#1@toctitle}%
      {%
        \csuse{gls@tr@set@#1@toctitle}%
      }%
      {%
        \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
      }%
    }%
    \renewcommand*{\glossaryname}{\translate{Glossary}}%
    \renewcommand*{\acronymname}{\translate{Acronyms}}%
    \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
    \renewcommand*{\descriptionname}{%
      \translate{Description (glossaries)}}%
    \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
    \renewcommand*{\pagelistname}{%
      \translate{Page List (glossaries)}}%
    \renewcommand*{\glssymbolsgroupname}{%
      \translate{Symbols (glossaries)}}%
    \renewcommand*{\glsnumbersgroupname}{%
      \translate{Numbers (glossaries)}}%
  }{}%
\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.

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

\@nopostdesc Suppress next description terminator.

```
\newcommand*{\@nopostdesc}{%
  \let\org@glspostdescription\glspostdescription
  \def\glspostdescription{%
    \let\glspostdescription\org@glspostdescription}%
}
```

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

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

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

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

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

```
\newcommand{\setStyleFile}[1]{%
  \renewcommand*{\gls@istfilebase}{#1}%
```

Just in case \istfilename has been modified.

```
  \ifglsxindy
    \def\istfilename{\gls@istfilebase.xdy}
  \else
    \def\istfilename{\gls@istfilebase.ist}
  \fi
}
```

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

```
\@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

```
\ifglsxindy
  \def\istfilename{\gls@istfilebase.xdy}
\else
  \def\istfilename{\gls@istfilebase.ist}
\fi
```

gls@istfilebase

```
\newcommand*{\gls@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

```
\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

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

lsSetCompositor Sets the compositor.

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

Only use before \makeglossaries

```
\@onlypremakeg\glsSetCompositor
```

(The page compositor is usually defined as a dash when using `makeindex`, but most of the standard counters used by LaTeX 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.

`Alphacompositor` 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.

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

`AlphaCompositor` Sets the alpha compositor.

```
\ifglsxindy
  \newcommand*\glsSetAlphaCompositor[1]{%
     \renewcommand*\@glsAlphacompositor{#1}}
\else
  \newcommand*\glsSetAlphaCompositor[1]{%
    \glsnoxindywarning\glsSetAlphaCompositor}
\fi
```

Can only be used before `\makeglossaries`

```
\@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.

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

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

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

Only has an effect when used before `\makeglossaries`

```
\@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.

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

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

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

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

```
\ifcsundef{hyperlink}%
{%
  \newcommand*{\glsnumberformat}[1]{#1}%
}%
{%
  \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
}
```

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

`\delimN`

```
\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`

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

The glossary preamble is given by `\glossarypreamble`. This will appear after the glossary sectioning command, and before the theglossary environment. It is designed to allow the user to add information pertaining to the glossary (e.g. "page numbers in italic indicate the primary definition") therefore `\glossarypremable` 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`.

`lossarypreamble`

```
\newcommand*{\glossarypreamble}{%
  \csuse{@glossarypreamble@\currentglossary}%
}
```

`glossarypreamble`

`\setglossarypreamble[⟨type⟩]{⟨text⟩}`

Code provided by Michael Pock.

```
\newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
  \ifglossaryexists{#1}{%
    \csgdef{@glossarypreamble@#1}{#2}%
  }{%
    \GlossariesWarning{%
      Glossary '#1' is not defined%
    }%
  }%
}
```

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{}}
```

ossarypostamble

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

glossarysection 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`.

```
\newcommand*{\glossarysection}[2][\@gls@title]{%
  \def\@gls@title{#2}%
  \ifcsundef{phantomsection}%
  {%
    \@glossarysection{#1}{#2}%
  }%
  {%
    \@p@glossarysection{#1}{#2}%
  }%

  \glsglossarymark{\glossarytoctitle}%
}
```

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

```
\ifcsundef{glossarymark}%
{%
  \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
}%
{%
  \@ifclassloaded{memoir}
  {%
    \newcommand{\glsglossarymark}[1]{%
      \ifglsucmark
        \markboth{\memUChead{#1}}{\memUChead{#1}}%
      \else
        \markboth{#1}{#1}%
      \fi
    }
  }%
  {%
    \newcommand{\glsglossarymark}[1]{%
      \ifglsucmark
        \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
      \else
        \@mkboth{#1}{#1}%
      \fi
    }
  }
}
```

\glossarymark Provided for backward compatibility:

```
\providecommand{\glossarymark}[1]{%
  \ifglsucmark
    \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
  \else
    \@mkboth{#1}{#1}%
  \fi
}
```

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

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

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

glossarysection

```
\newcommand*{\@glossarysection}[2]{%
  \ifdefempty\@@glossarysecstar
  {%
    \csname\@@glossarysec\endcsname[#1]{#2}%
  }%
  {%
    \csname\@@glossarysec\endcsname*{#2}%
    \@gls@toc{#1}{\@@glossarysec}%
  }%
```

Do automatic labelling if required

```
  \@@glossaryseclabel
}
```

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

glossarysection

```
\newcommand*{\@p@glossarysection}[2]{%
  \glsclearpage
  \phantomsection
  \ifdefempty\@@glossarysecstar
  {%
    \csname\@@glossarysec\endcsname{#2}%
  }%
  {%
    \@gls@toc{#1}{\@@glossarysec}%
      \csname\@@glossarysec\endcsname*{#2}%
  }%
```

Do automatic labelling if required

```
  \@@glossaryseclabel
}
```

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

```
\newcommand*{\gls@doclearpage}{%
  \ifthenelse{\equal{\@@glossarysec}{chapter}}%
  {%
    \ifcsundef{cleardoublepage}%
    {%
      \clearpage
    }%
    {%
      \ifcsdef{if@openright}%
      {%
         \if@openright
           \cleardoublepage
         \else
           \clearpage
         \fi
      }%
      {%
         \cleardoublepage
      }%
    }%
  }%
  {}%
}
```

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

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

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

\@gls@toc

```
\newcommand*{\@gls@toc}[2]{%
  \ifglstoc
    \ifglsnumberline
      \addcontentsline{toc}{#2}{\protect\numberline{}#1}%
    \else
      \addcontentsline{toc}{#2}{#1}%
    \fi
  \fi
}
```

1.4 Xindy

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

snoxindywarning

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

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

akeindexwarning

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

```
\newcommand*{\glsnomakeindexwarning}[1]{%
  \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
}
```

\@xdyattributes

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

```
\ifglsxindy
  \edef\@xdyattributes{\string"default\string"}%
\fi
```

dyattributelist

Comma-separated list of attributes.

```
\ifglsxindy
  \edef\@xdyattributelist{}%
\fi
```

\@xdylocref

Define list of markup location references.

```
\ifglsxindy
  \def\@xdylocref{}
\fi
```

\@gls@ifinlist

```
\newcommand*{\@gls@ifinlist}[4]{%
  \def\@do@ifinlist##1,#1,##2\end@doifinlist{%
    \def\@gls@listsuffix{##2}%
    \ifx\@gls@listsuffix\@empty
```

```
       #4%
    \else
       #3%
    \fi
  }%
  \@do@ifinlist,#2,#1,\end@doifinlist
}
```

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.

```
\ifglsxindy
  \newcommand*{\@xdycounters}{\glscounter}
  \newcommand*\GlsAddXdyCounters[1]{%
    \@for\@gls@ctr:=#1\do{%
```

Check if already in list before adding.

```
       \edef\@do@addcounter{%
          \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
          {%
             \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
               \noexpand\@gls@ctr}%
          }%
       }%
       \@do@addcounter
    }
  }
```

Only has an effect before `\writeist`:

```
  \@onlypremakeg\GlsAddXdyCounters
\else
  \newcommand*\GlsAddXdyCounters[1]{%
    \glsnoxindywarning\GlsAddXdyAttribute
  }
\fi
```

saddxdycounters Counters must all be identified before adding attributes.

```
\newcommand*\@disabled@glsaddxdycounters{%
   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
   can't be used after \string\GlsAddXdyAttribute}{Move all
   occurrences of \string\GlsAddXdyCounters\space before the first
   instance of \string\GlsAddXdyAttribute}%
}
```

AddXdyAttribute Adds an attribute.

```
\ifglsxindy
```

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

```
  \newcommand*\@glsaddxdyattribute[2]{%
```

Add to xindy attribute list

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

Add to xindy markup location.

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

Define associated attribute command \glsX⟨*counter*⟩X⟨*attribute*⟩{⟨*Hprefix*⟩}{⟨*n*⟩}

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

High-level command:

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

Add to comma-separated attribute list

```
    \ifx\@xdyattributelist\@empty
      \edef\@xdyattributelist{#1}%
    \else
      \edef\@xdyattributelist{\@xdyattributelist,#1}%
    \fi
```

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

```
    \@for\@this@counter:=\@xdycounters\do{%
      \protected@edef\gls@do@addxdyattribute{%
        \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
      }
      \gls@do@addxdyattribute
    }%
```

All occurrences of \GlsAddXdyCounters must be used before this command

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

Only has an effect before \writeist:

```
  \@onlypremakeg\GlsAddXdyAttribute
\else
  \newcommand*\GlsAddXdyAttribute[1]{%
    \glsnoxindywarning\GlsAddXdyAttribute}
\fi
```

finedattributes Add known attributes for all defined counters

```
\ifglsxindy
\newcommand*{\@gls@addpredefinedattributes}{%
```

```
  \GlsAddXdyAttribute{glsnumberformat}
  \GlsAddXdyAttribute{textrm}
  \GlsAddXdyAttribute{textsf}
  \GlsAddXdyAttribute{texttt}
  \GlsAddXdyAttribute{textbf}
  \GlsAddXdyAttribute{textmd}
  \GlsAddXdyAttribute{textit}
  \GlsAddXdyAttribute{textup}
  \GlsAddXdyAttribute{textsl}
  \GlsAddXdyAttribute{textsc}
  \GlsAddXdyAttribute{emph}
  \GlsAddXdyAttribute{glshypernumber}
  \GlsAddXdyAttribute{hyperrm}
  \GlsAddXdyAttribute{hypersf}
  \GlsAddXdyAttribute{hypertt}
  \GlsAddXdyAttribute{hyperbf}
  \GlsAddXdyAttribute{hypermd}
  \GlsAddXdyAttribute{hyperit}
  \GlsAddXdyAttribute{hyperup}
  \GlsAddXdyAttribute{hypersl}
  \GlsAddXdyAttribute{hypersc}
  \GlsAddXdyAttribute{hyperemph}

  \GlsAddXdyAttribute{glsignore}
}
\else
  \let\@gls@addpredefinedattributes\relax
\fi
```

`dyuseralphabets` List of additional alphabets

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

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

```
\ifglsxindy
  \newcommand*{\GlsAddXdyAlphabet}[2]{%
  \edef\@xdyuseralphabets{%
    \@xdyuseralphabets ^^J
    (define-alphabet "#1" (#2))}}
\else
  \newcommand*{\GlsAddXdyAlphabet}[2]{%
     \glsnoxindywarning\GlsAddXdyAlphabet}
\fi
```

This code is only required for xindy:

```
\ifglsxindy
```

`dy@locationlist` List of predefined location names.

```
  \newcommand*{\@gls@xdy@locationlist}{%
```

```
    roman-page-numbers,%
    Roman-page-numbers,%
    arabic-page-numbers,%
    alpha-page-numbers,%
    Alpha-page-numbers,%
    Appendix-page-numbers,%
    arabic-section-numbers%
}
```

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

`an-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.

```
\protected@edef\@gls@roman{\@roman{0\string"
    \string"roman-numbers-lowercase\string" :sep \string"}}%
\@onelevel@sanitize\@gls@roman
\edef\@tmp{\string" \string"roman-numbers-lowercase\string"
     :sep \string"}%
\@onelevel@sanitize\@tmp
\ifx\@tmp\@gls@roman
  \expandafter
    \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
      \string"roman-numbers-lowercase\string"%
    }%
\else
   \expandafter
    \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
      :sep \string"\@gls@roman\string"%
    }%
\fi
```

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

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

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

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

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

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

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

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

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

```
\expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
  \string"ALPHA\string"
  :sep \string"\@glsAlphacompositor\string"
  \string"arabic-numbers\string"%
}
```

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

```
\expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
  \string"arabic-numbers\string"
   :sep \string"\glscompositor\string"
  \string"arabic-numbers\string"%
}%
```

serlocationdefs List of additional location definitions (separated by `^^J`)

```
\def\@xdyuserlocationdefs{}
```

erlocationnames List of additional user location names

```
\def\@xdyuserlocationnames{}
```

End of xindy-only block:

```
\fi
```

xdycrossrefhook Hook used after writing cross-reference class information.

```
\ifglsxindy
 \newcommand\@xdycrossrefhook{}
\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.)

```
\ifglsxindy
  \newcommand*{\GlsAddXdyLocation}[3][]{%
    \def\@gls@tmp{#1}%
    \ifx\@gls@tmp\@empty
      \edef\@xdyuserlocationdefs{%
         \@xdyuserlocationdefs ^^J%
         (define-location-class \string"#2\string"^^J\space\space
         \space(:sep \string"{}\glsopenbrace\string" #3
                :sep \string"\glsclosebrace\string"))
      }%
    \else
      \edef\@xdyuserlocationdefs{%
         \@xdyuserlocationdefs ^^J%
         (define-location-class \string"#2\string"^^J\space\space
         \space(:sep "\glsopenbrace"
                #1
                :sep "\glsclosebrace\glsopenbrace" #3
                :sep "\glsclosebrace"))
      }%
    \fi
    \edef\@xdyuserlocationnames{%
       \@xdyuserlocationnames^^J\space\space\space
       \string"#2\string"}%
  }
```

Only has an effect before `\writeist`:

```
  \@onlypremakeg\GlsAddXdyLocation
\else
  \newcommand*{\GlsAddXdyLocation}[2]{%
    \glsnoxindywarning\GlsAddXdyLocation}
\fi
```

`ationclassorder` Define location class order

```
\ifglsxindy
  \def\@xdylocationclassorder{^^J\space\space\space
    \string"roman-page-numbers\string"^^J\space\space\space
    \string"arabic-page-numbers\string"^^J\space\space\space
    \string"arabic-section-numbers\string"^^J\space\space\space
    \string"alpha-page-numbers\string"^^J\space\space\space
    \string"Roman-page-numbers\string"^^J\space\space\space
    \string"Alpha-page-numbers\string"^^J\space\space\space
    \string"Appendix-page-numbers\string"
    \@xdyuserlocationnames^^J\space\space\space
    \string"see\string"
  }
\fi
```

Change the location order.

`ationClassOrder`

```
\ifglsxindy
  \newcommand*\GlsSetXdyLocationClassOrder[1]{%
    \def\@xdylocationclassorder{#1}}
\else
  \newcommand*\GlsSetXdyLocationClassOrder[1]{%
    \glsnoxindywarning\GlsSetXdyLocationClassOrder}
\fi
```

`\@xdysortrules` Define sort rules

```
\ifglsxindy
  \def\@xdysortrules{}
\fi
```

\GlsAddSortRule Add a sort rule

```
\ifglsxindy
  \newcommand*\GlsAddSortRule[2]{%
    \expandafter\toks@\expandafter{\@xdysortrules}%
    \protected@edef\@xdysortrules{\the\toks@ ^^J
     (sort-rule \string"#1\string" \string"#2\string")}%
  }
\else
  \newcommand*\GlsAddSortRule[2]{%
    \glsnoxindywarning\GlsAddSortRule}
\fi
```

yrequiredstyles Define list of required styles (this should be a comma-separated list of xindy styles)

```
\ifglsxindy
  \def\@xdyrequiredstyles{tex}
\fi
```

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

```
\ifglsxindy
  \newcommand*\GlsAddXdyStyle[1]{%
    \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
\else
  \newcommand*\GlsAddXdyStyle[1]{%
    \glsnoxindywarning\GlsAddXdyStyle}
\fi
```

GlsSetXdyStyles Reset the list of required styles

```
\ifglsxindy
  \newcommand*\GlsSetXdyStyles[1]{%
    \edef\@xdyrequiredstyles{#1}}
\else
  \newcommand*\GlsSetXdyStyles[1]{%
    \glsnoxindywarning\GlsSetXdyStyles}
\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.

```
\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.

```
\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.

```
\ifglsxindy
  \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
  \ifglossaryexists{#1}{%
    \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
  }{%
    \PackageError{glossaries}{Can't set language type for
    glossary type '#1' --- no such glossary}{%
    You have specified a glossary type that doesn't exist}}}
\else
  \newcommand*\GlsSetXdyLanguage[2][]{%
    \glsnoxindywarning\GlsSetXdyLanguage}
\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.

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

sSetXdyCodePage Define command to set the code page.

```
\ifglsxindy
  \newcommand*{\GlsSetXdyCodePage}[1]{%
    \renewcommand*{\gls@codepage}{#1}%
  }
```

Suggested by egreg:

```
  \AtBeginDocument{%
    \ifx\gls@codepage\@empty
      \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}%
    \fi
  }
\else
  \newcommand*{\GlsSetXdyCodePage}[1]{%
    \glsnoxindywarning\GlsSetXdyCodePage}
\fi
```

xdylettergroups Store letter group definitions.

```
\ifglsxindy
  \ifgls@xindy@glsnumbers
    \def\@xdylettergroups{(define-letter-group
       \string"glsnumbers\string"^^J\space\space\space
       :prefixes (\string"0\string" \string"1\string"
       \string"2\string" \string"3\string" \string"4\string"
       \string"5\string" \string"6\string" \string"7\string"
       \string"8\string" \string"9\string")^^J\space\space\space
       \@xdynumbergrouporder)}
  \else
    \def\@xdylettergroups{}
  \fi
\fi
```

sAddLetterGroup 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.

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

1.5 Loops and conditionals

orallglossaries 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.

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

\forallacronyms

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

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

\forglsentries[⟨*type*⟩]{⟨*cmd*⟩}{⟨*code*⟩}

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

```
\newcommand*{\forglsentries}[3][\glsdefaulttype]{%
  \edef\@@glo@list{\csname glolist@#1\endcsname}%
  \@for#2:=\@@glo@list\do
  {%
    \ifdefempty{#2}{}{#3}%
  }%
}
```

orallglsentries 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@`.

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

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

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

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

```
\newcommand{\ifglossaryexists}[3]{%
  \ifcsundef{@glotype@#1@out}{#3}{#2}%
}
```

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 redefining `\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`

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

`fglsentryexists` 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.

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

`\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*⟩.

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

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 exists, otherwise do ⟨*code*⟩.

```
\newcommand{\glsdoifexists}[2]{%
  \ifglsentryexists{#1}{#2}{%
    \PackageError{glossaries}{Glossary entry `\glsdetoklabel{#1}'
    has not been defined}{You need to define a glossary entry before you
    can use it.}}%
}
```

glsdoifnoexists

`\glsdoifnoexists{⟨label⟩}{⟨code⟩}`

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

```
\newcommand{\glsdoifnoexists}[2]{%
  \ifglsentryexists{#1}{%
    \PackageError{glossaries}{Glossary entry `\glsdetoklabel{#1}' has already
    been defined}{}}{#2}%
}
```

doifexistsorwarn

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

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

```
\newcommand{\glsdoifexistsorwarn}[2]{%
  \ifglsentryexists{#1}{#2}{%
    \GlossariesWarning{Glossary entry `\glsdetoklabel{#1}'
      has not been defined}%
  }%
}
```

lsdoifexistsordo

```
\glsdoifexistsordo{⟨label⟩}{⟨code⟩}{⟨undef code⟩}
```

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

```
\newcommand{\glsdoifexistsordo}[3]{%
  \ifglsentryexists{#1}{#2}{%
    \PackageError{glossaries}{Glossary entry `\glsdetoklabel{#1}'
    has not been defined}{You need to define a glossary entry before you
    can use it.}%
    #3%
  }%
}
```

sarynoexistsordo `\doifglossarynoexistsordo{⟨label⟩}{⟨code⟩}{⟨else code⟩}`

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

```
\newcommand{\doifglossarynoexistsordo}[3]{%
  \ifglossaryexists{#1}%
  {%
    \PackageError{glossaries}{Glossary type '#1' already exists}{}%
    #3%
  }%
  {#2}%
}
```

fglshaschildren `\ifglshaschildren{⟨label⟩}{⟨true part⟩}{⟨false part⟩}`

```
\newcommand{\ifglshaschildren}[3]{%
  \glsdoifexists{#1}%
  {%
     \def\do@glshaschildren{#3}%
     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
     \expandafter\forglsentries\expandafter
       [\csname glo@\@gls@thislabel @type\endcsname]
     {\glo@label}%
     {%
       \letcs\glo@parent{glo@\glo@label @parent}%
       \ifdefequal\@gls@thislabel\glo@parent
       {%
         \def\do@glshaschildren{#2}%
         \@endfortrue
       }%
       {}%
     }%
     \do@glshaschildren
  }%
}
```

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

```
\newcommand{\ifglshasparent}[3]{%
  \glsdoifexists{#1}%
  {%
    \ifcsempty{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
  }%
}
```

\ifglshasdesc `\ifglshasdesc{`⟨*label*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

```
\newcommand*{\ifglshasdesc}[3]{%
  \ifcsempty{glo@\glsdetoklabel{#1}@desc}%
  {#3}%
  {#2}%
}
```

sdescsuppressed `\ifglsdescsuppressed{`⟨*label*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}` Does ⟨*true part*⟩ if the description is just `\nopostdesc` otherwise does ⟨*false part*⟩.

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

\ifglshassymbol `\ifglshassymbol{`⟨*label*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

```
\newcommand*{\ifglshassymbol}[3]{%
  \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
  \ifdefempty\@glo@symbol
  {#3}%
  {%
    \ifdefequal\@glo@symbol\@gls@default@value
    {#3}%
    {#2}%
  }%
}
```

\ifglshaslong `\ifglshaslong{`⟨*label*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

```
\newcommand*{\ifglshaslong}[3]{%
  \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
  \ifdefempty\@glo@long
  {#3}%
  {%
    \ifdefequal\@glo@long\@gls@default@value
    {#3}%
    {#2}%
  }%
}
```

\ifglshasshort `\ifglshasshort{`⟨*label*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

```
\newcommand*{\ifglshasshort}[3]{%
  \letcs{\@glo@short}{glo@\glsdetoklabel{#1}@short}%
  \ifdefempty\@glo@short
  {#3}%
  {%
```

```
    \ifdefequal\@glo@short\@gls@default@value
    {#3}%
    {#2}%
  }%
}
```

\ifglshasfield

```
\ifglshasfield{⟨field⟩}{⟨label⟩}{⟨true part⟩}{⟨false part⟩}
```

```
\newcommand*{\ifglshasfield}[4]{%
  \glsdoifexists{#2}%
  {%
    \letcs{\@glo@thisvalue}{glo@\glsdetoklabel{#2}@#1}%
```

First check supplied field label is defined.

```
    \ifdef\@glo@thisvalue
    {%
```

Is defined, so now check if empty.

```
      \ifdefempty\@glo@thisvalue
      {%
```

Is empty, so doesn't have field set.

```
        #4%
      }%
      {%
```

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

```
        \ifdefequal\@glo@thisvalue\@gls@default@value
        {%
```

Value is set to the default value.

```
          #4%
        }%
        {%
```

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

```
         \let\glscurrentfieldvalue\@glo@thisvalue
         #3%
        }%
      }%
    }%
    {%
```

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

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

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

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

Is defined, so now check if empty.

```
         \letcs{\@glo@thisvalue}{glo@\glsdetoklabel{#2}@\@gls@thisfield}%
         \ifdefempty\@glo@thisvalue
         {%
```

Is empty so field hasn't been set.

```
           #4%
         }%
         {%
```

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

```
           \ifdefequal\@glo@thisvalue\@gls@default@value
           {%
```

Value is set to the default value.

```
             #4%
           }%
           {%
```

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

```
            \let\glscurrentfieldvalue\@glo@thisvalue
            #3%
           }%
         }%
       }%
       {%
```

Not defined.

```
         \GlossariesWarning{Unknown entry field `#1'}%
         #4%
       }%
    }%
  }%
}
```

rrentfieldvalue

```
\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

```
\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.

```
\newcommand*\@gls@provide@newglossary{%
  \protected@write\@auxout{}{\string\providecommand\string\@newglossary[4]{}}%
```

Only need to do this once.

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

\defglsentryfmt Allow different glossaries to have different display styles.

```
\newcommand*{\defglsentryfmt}[2][\glsdefaulttype]{%
  \csgdef{gls@#1@entryfmt}{#2}%
}
```

\gls@doentryfmt

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

ls@forbidtexext 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.

```
\newcommand*{\@gls@forbidtexext}[1]{%
 \ifboolexpr{test {\ifdefstring{#1}{tex}}
          or test {\ifdefstring{#1}{TEX}}}
 {%
   \def#1{nottex}%
   \PackageError{glossaries}%
    {Forbidden '.tex' extension replaced with '.nottex'}%
    {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
     Don't use '.tex' as an extension for a temporary file.}%
 }%
 {%
 }%
}
```

\gls@gobbleopt Discard optional argument.

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

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

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

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

\ns@newglossary Define the unstarred version.

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

Check if default has been set

```
  \ifundef\glsdefaulttype
  {%
    \gdef\glsdefaulttype{#2}%
  }{}%
```

Add this to the list of glossary types:

```
  \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.

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

Store the file extensions:

```
  \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
  \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
  \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
  \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
  \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
  \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
  \expandafter\def\csname @glotype@#2@title\endcsname{#5}%

  \@gls@provide@newglossary
  \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.

```
  \ifcsundef{gls@#2@entryfmt}%
  {%
    \defglsentryfmt[#2]{\glsentryfmt}%
  }%
  {}%
```

Define sort counter if required:

```
  \@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.)

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

\altnewglossary

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

Only define new glossaries in the preamble:

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

Only define new glossaries before \makeglossaries

```
\@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

```
\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

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

Add counter to xindy list, if not already added:

```
  \ifglsxindy
    \GlsAddXdyCounters{#2}%
  \fi
}
```

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

@gls@getcounter

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

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

```
\glsdefmain
```

Define the “acronym” glossaries if required.

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

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

```
\@gls@do@symbolsdef
\@gls@do@numbersdef
\@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.

```
\newcommand*{\newignoredglossary}[1]{%
  \ifdefempty\@ignored@glossaries
  {%
    \edef\@ignored@glossaries{#1}%
  }%
  {%
    \eappto\@ignored@glossaries{,#1}%
  }%
  \csgdef{glolist@#1}{,}%
  \ifcsundef{gls@#1@entryfmt}%
  {%
    \defglsentryfmt[#1]{\glsentryfmt}%
  }%
  {}%
  \ifdefempty\@gls@nohyperlist
  {%
     \renewcommand*{\@gls@nohyperlist}{#1}%
  }%
  {%
     \eappto\@gls@nohyperlist{,#1}%
  }%
}
```

ored@glossaries

List of ignored glossaries.

```
\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.

```
\newcommand*{\ifignoredglossary}[3]{%
  \edef\@gls@igtype{#1}%
  \expandafter\DTLifinlist\expandafter
    {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
}
```

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.

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

`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`.

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

`scriptionplural`

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

`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*⟩.

```
\define@key{glossentry}{sort}{%
\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.

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

`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.

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

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.

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

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.

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

s@default@value

```
\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).

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

symbolplural

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

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

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

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

```
\define@key{glossentry}{counter}{%
  \ifcsundef{c@#1}%
  {%
    \PackageError{glossaries}%
    {There is no counter called '#1'}%
    {%
      The counter key should have the name of a valid counter
      as its value%
    }%
  }%
  {%
    \def\@glo@counter{#1}%
  }%
}
```

see The see key specifies a list of cross-references

```
\define@key{glossentry}{see}{%
  \gls@set@xr@key{see}{\@glo@see}{#1}%
}
```

\gls@set@xr@key

```
\gls@set@xr@key{⟨key name⟩}{⟨cs⟩}{⟨value⟩}
```

Assign a cross-reference key.

```
\newcommand*{\gls@set@xr@key}[3]{%
  \renewcommand*{\gls@xr@key}{#1}%
  \gls@checkseeallowed
  \def#2{#3}%
  \@glo@seeautonumberlist
}
```

\gls@xr@key

```
\newcommand*{\gls@xr@key}{see}
```

checkseeallowed

```
\newcommand*{\gls@checkseeallowed}{%
 \@gls@see@noindex
}
```

ed@preambleonly

```
\newcommand*{\gls@checkseeallowed@preambleonly}{%
  \GlossariesWarning{glossaries}%
  {‘\gls@xr@key’ key doesn’t have any effect when used in the document
   environment. Move the definition to the preamble
    after \string\makeglossaries\space
   or \string\makenoidxglossaries}%
}
```

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

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

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

```
\define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
  \ifcase\nr\relax
    \def\@glo@prefix{\glsnonextpages}%
    \@gls@savenonumberlist{true}%
  \else
```

```
    \def\@glo@prefix{\glsnextpages}%
    \@gls@savenonumberlist{false}%
  \fi
}
```

avenonumberlist

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.

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

nitnonumberlist

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

nitnonumberlist

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

avenonumberlist

Allow the nonumberlist value to be saved.

```
\newcommand*{\@gls@enablesavenonumberlist}{%
  \renewcommand*{\@gls@initnonumberlist}{%
    \undef\@glo@nonumberlist
  }%
  \renewcommand*{\@gls@savenonumberlist}[1]{%
    \def\@glo@nonumberlist{##1}%
  }%
  \renewcommand*{\@gls@storenonumberlist}[1]{%
    \ifdef\@glo@nonumberlist
    {%
      \cslet{glo@\glsdetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
    }%
    {}%
  }%
  \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
}
```

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

user1

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

user2

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

user3

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

user4

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

user5

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

user6

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

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.

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

shortplural This key is provided for use by \newacronym.

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

long This key is provided for use by \newacronym.

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

longplural This key is provided for use by \newacronym.

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

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

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

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

```
\newcommand*\@glsnodesc{%
  \PackageError{glossaries}
  {%
    description key required in \string\newglossaryentry\space
    for entry ‘\@glo@label’%
  }%
  {%
    You haven’t specified the entry description%
  }%
}%
```

lsdefaultplural Now obsolete. Don’t use.

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

ssingnumberlist Define a command to generate warning when numberlist not set.

```
\newcommand*{\@gls@missingnumberlist}[1]{%
  ??%
  \ifglssavenumberlist
    \GlossariesWarning{Missing number list for entry ‘#1’.
     Maybe makeglossaries + rerun required}%
  \else
    \PackageError{glossaries}%
    {Package option ‘savenumberlist=true’ required}%
    {%
      You must use the ‘savenumberlist’ package option
      to reference location lists.%
    }%
  \fi
}
```

@glsdefaultsort Define command to set default sort.

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

\gls@level Register to increment entry levels.

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

@noexpand@field

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

noexpand@fields

```
\newcommand{\@gls@noexpand@fields}[4]{%
  \ifcsdef{gls@assign@#3@field}
  {%
     \ifdefequal{#4}{\@gls@default@value}%
```

```
      {%
        \edef\@gls@value{\expandonce{#1}}%
        \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
      }%
      {%
        \csuse{gls@assign@#3@field}{#2}{#4}%
      }%
  }%
  {%
    \ifdefequal{#4}{\@gls@default@value}%
    {%
        \edef\@gls@value{\expandonce{#1}}%

        \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
    }%
    {%
      \@@gls@noexpand@field{#2}{#3}{#4}%
    }%
  }%
}
```

ls@expand@field

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

s@expand@fields

```
\newcommand{\@gls@expand@fields}[4]{%
  \ifcsdef{gls@assign@#3@field}
  {%
     \ifdefequal{#4}{\@gls@default@value}%
     {%
       \edef\@gls@value{\expandonce{#1}}%
       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
     }%
     {%
       \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
       {%
         \@@gls@expand@field{#2}{#3}{#4}%
       }%
       {%
         \csuse{gls@assign@#3@field}{#2}{#4}%
       }%
     }%
  }%
  {%
    \ifdefequal{#4}{\@gls@default@value}%
    {%
```

```
      \@@gls@expand@field{#2}{#3}{#1}%
    }%
    {%
      \@@gls@expand@field{#2}{#3}{#4}%
    }%
  }%
}
```

swithexpandonce

```
\def\@gls@expandonce{\expandonce}
\def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
  \def\@gls@tmp{#1}%
  \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
}
```

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.

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

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

```
\newcommand*{\glsexpandfields}{%
  \let\gls@assign@field\@gls@expand@fields
}
```

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

```
\newcommand*{\glsnoexpandfields}{%
  \let\gls@assign@field\@gls@noexpand@fields
}
```

ewglossaryentry Define `\newglossaryentry {`⟨*label*⟩`} {`⟨*key-val list*⟩`}`. There are two required fields in ⟨*key-val list*⟩: name (or parent) and description. (See above.)

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

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

```
  \glsdoifnoexists{#1}%
  {%
    \gls@defglossaryentry{#1}{#2}%
  }%
}
```

ewglossaryentry 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.

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

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

```
\newrobustcmd{\provideglossaryentry}[2]{%
  \ifglsentryexists{#1}%
  {}%
  {%
    \gls@defglossaryentry{#1}{#2}%
  }%
}
\@onlypreamble{\provideglossaryentry}
```

w@glossaryentry For use in document environment. This opens the `.glsdefs` file, if not already open, so that the entry definition can be saved for the next LaTeX run. This means that any glossaries at the start of the document can access the entry information.

```
\newrobustcmd{\new@glossaryentry}[2]{%
  \ifundef\@gls@deffile
  {%
     \global\newwrite\@gls@deffile
     \immediate\openout\@gls@deffile=\jobname.glsdefs
  }%
  {}%
  \ifglsentryexists{#1}{}%
  {%
     \gls@defglossaryentry{#1}{#2}%
  }%
  \@gls@writedef{#1}%
}
```

At the start of the document input the `.glsdefs` file if it exists. This is now done by `\gls@begindocdefs`, which is redefined by glossaries-extra, so that this step can be skipped to avoid loading an obsolete `.glsdefs` file if the user switches to glossaries-extra with `docdef=restricted`.

```
\AtBeginDocument{\gls@begindocdefs}
```

The end of the document needs to check if the `.glsdefs` file has been opened, in which case it needs to be closed.

```
\AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}
```

ls@begindocdefs Input the `.glsdefs` file if it exists and enable document definitions if permitted.

```
\newcommand*{\gls@begindocdefs}{%
  \@gls@enablesavenonumberlist
  \edef\@gls@restoreat{\noexpand\catcode`\noexpand\@=\number\catcode`\@\relax}%
  \makeatletter
  \InputIfFileExists{\jobname.glsdefs}{}{}%
  \@gls@restoreat
  \undef\@gls@restoreat
  \gls@defdocnewglossaryentry
}
```

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

```
\newcommand*{\@gls@writedef}[1]{%
  \immediate\write\@gls@deffile
  {%
    \string\ifglsentryexists{#1}{}\glspercentchar^^J%
    \expandafter\@gobble\string\{\glspercentchar^^J%
      \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
      \expandafter\@gobble\string\{\glspercentchar%
  }%
```

Write key value information:

```
  \@for\@gls@map:=\@gls@keymap\do
  {%
    \letcs\glo@value{glo@\glsdetoklabel{#1}@\expandafter\@secondoftwo\@gls@map}%
    \ifdef\glo@value
    {%
      \@onelevel@sanitize\glo@value
      \immediate\write\@gls@deffile
      {%
        \expandafter\@firstoftwo\@gls@map
          =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
          \glspercentchar
      }%
    }%
    {}%
  }%
```

Provide hook:

```
  \glswritedefhook
  \immediate\write\@gls@deffile
  {%
          \glspercentchar^^J%
      \expandafter\@gobble\string\}\glspercentchar^^J%
    \expandafter\@gobble\string\}\glspercentchar%
  }%
}
```

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

```
\newcommand*{\@gls@keymap}{%
  {name}{name},%
  {sort}{sortvalue},% unescaped sort value
  {type}{type},%
  {first}{first},%
  {firstplural}{firstpl},%
  {text}{text},%
```

```
 {plural}{plural},%
 {description}{desc},%
 {descriptionplural}{descplural},%
 {symbol}{symbol},%
 {symbolplural}{symbolplural},%
 {user1}{useri},%
 {user2}{userii},%
 {user3}{useriii},%
 {user4}{useriv},%
 {user5}{userv},%
 {user6}{uservi},%
 {long}{long},%
 {longplural}{longpl},%
 {short}{short},%
 {shortplural}{shortpl},%
 {counter}{counter},%
 {parent}{parent}%
}
```

\@gls@fetchfield

`\@gls@fetchfield{⟨cs⟩}{⟨field⟩}`

Fetches the internal field label from the given user ⟨*field*⟩ and stores in ⟨*cs*⟩.

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

Ensure user field name is fully expanded

```
 \edef\@gls@thisval{#2}%
```

Iterate through known mappings until we find the one for this field.

```
 \@for\@gls@map:=\@gls@keymap\do{%
  \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
  \ifdefequal{\@this@key}{\@gls@thisval}%
  {%
```

Found it.

```
    \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```
    \@endfortrue
  }%
  {}%
 }%
}
```

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.

```
\newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```
\newcommand*{\@sglsaddstoragekey}[1]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \expandafter\newcommand\expandafter*\expandafter
     {\csname gls@assign@#1@field\endcsname}[2]{%
       \@@gls@expand@field{##1}{#1}{##2}%
     }%
  }%
  {}%
  \@glsaddstoragekey{#1}%
}
```

Unstarred version doesn't override default expansion.

```
\newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

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

Set up the key.

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

Set the default value.

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

Assignment code.

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

Define the no-link commands.

```
    \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
  }%
  {%
    \PackageError{glossaries}{Key `#1' already exists}{}%
  }%
}
```

\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.

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

Starred version switches on expansion for this key.

```
\newcommand*{\@sglsaddkey}[1]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \expandafter\newcommand\expandafter*\expandafter
     {\csname gls@assign@#1@field\endcsname}[2]{%
       \@@gls@expand@field{##1}{#1}{##2}%
     }%
  }%
  {}%
  \@glsaddkey{#1}%
}
```

Unstarred version doesn't override default expansion.

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

Check the specified key doesn't already exist.

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

Set up the key.

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

Set the default value.

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

Assignment code.

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

Define the no-link commands.

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

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

```
    \ifcsdef{@gls@user@#1@}%
    {%
       \PackageError{glossaries}%
       {Can't define '\string#5' as helper command
        '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
       {}%
    }%
    {%

      \expandafter\newcommand\expandafter*\expandafter
        {\csname @gls@user@#1\endcsname}[2][]{%
          \new@ifnextchar[%
            {\csuse{@gls@user@#1@}{##1}{##2}}%
            {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
      \csdef{@gls@user@#1@}##1##2[##3]{%
        \@gls@field@link{##1}{##2}{#3{##2}##3}%
      }%
      \newrobustcmd*{#5}{%
        \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
    }%
```

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

```
  \ifcsdef{@Gls@user@#1@}%
  {%
     \PackageError{glossaries}%
     {Can't define '\string#6' as helper command
      '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
     {}%
  }%
  {%

    \expandafter\newcommand\expandafter*\expandafter
      {\csname @Gls@user@#1\endcsname}[2][]{%
        \new@ifnextchar[%
          {\csuse{@Gls@user@#1@}{##1}{##2}}%
          {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
    \csdef{@Gls@user@#1@}##1##2[##3]{%
      \@gls@field@link{##1}{##2}{#4{##2}##3}%
    }%
    \newrobustcmd*{#6}{%
      \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
  }%
```

Finally the all caps version:

```
  \ifcsdef{@GLS@user@#1@}%
  {%
     \PackageError{glossaries}%
     {Can't define '\string#7' as helper command
      '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
     {}%
  }%
  {%

    \expandafter\newcommand\expandafter*\expandafter
      {\csname @GLS@user@#1\endcsname}[2][]{%
        \new@ifnextchar[%
          {\csuse{@GLS@user@#1@}{##1}{##2}}%
          {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
    \csdef{@GLS@user@#1@}##1##2[##3]{%
      \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
    }%
    \newrobustcmd*{#7}{%
      \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
  }%
}%
{%
  \PackageError{glossaries}{Key '#1' already exists}{}%
  }%
}
```

\glsfieldxdef

```
\glsfieldxdef{⟨label⟩}{⟨field⟩}{⟨definition⟩}
```

```
\newcommand{\glsfieldxdef}[3]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
      \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
   }%
   {%
      \PackageError{glossaries}{Key '#2' doesn't exist}{}%
   }%
 }%
}
```

\glsfieldedef

```
\glsfieldedef{⟨label⟩}{⟨field⟩}{⟨definition⟩}
```

```
\newcommand{\glsfieldedef}[3]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
      \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
   }%
   {%
      \PackageError{glossaries}{Key '#2' doesn't exist}{}%
   }%
 }%
}
```

\glsfieldgdef

```
\glsfieldgdef{⟨label⟩}{⟨field⟩}{⟨definition⟩}
```

```
\newcommand{\glsfieldgdef}[3]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
```

```
      \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
   }%
   {%
      \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
   }%
 }%
}
```

\glsfielddef

```
\glsfielddef{⟨label⟩}{⟨field⟩}{⟨definition⟩}
```

```
\newcommand{\glsfielddef}[3]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
      \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
   }%
   {%
      \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
   }%
 }%
}
```

\glsfieldfetch

```
\glsfieldfetch{⟨label⟩}{⟨field⟩}{⟨cs⟩}
```

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

```
\newcommand{\glsfieldfetch}[3]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
      \letcs#3{glo@\@glo@label @#2}%
   }%
   {%
      \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
   }%
 }%
}
```

\ifglsfieldeq

```
\ifglsfieldeq{⟨label⟩}{⟨field⟩}{⟨string⟩}{⟨true⟩}{⟨false⟩}
```

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

```
\newcommand{\ifglsfieldeq}[5]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
      \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
   }%
   {%
      \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
   }%
 }%
}
```

\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.

```
\newcommand{\ifglsfielddefeq}[5]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
      \expandafter\ifdefstrequal
       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
   }%
   {%
      \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
   }%
 }%
}
```

\ifglsfieldcseq

```
\ifglsfieldcseq{⟨label⟩}{⟨field⟩}{⟨cs name⟩}{⟨true⟩}{⟨false⟩}
```

As above but uses `\ifcsstrequal` instead of `\ifdefstrequal`

```
\newcommand{\ifglsfieldcseq}[5]{%
 \glsdoifexists{#1}%
 {%
   \edef\@glo@label{\glsdetoklabel{#1}}%
   \ifcsdef{glo@\@glo@label @#2}%
   {%
      \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
   }%
   {%
      \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
   }%
 }%
}
```

glswritedefhook

```
\newcommand*{\glswritedefhook}{}
```

gls@assign@desc

```
\newcommand*{\gls@assign@desc}[1]{%
  \gls@assign@field{}{#1}{desc}{\@glo@desc}%
  \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
}
```

ewglossaryentry

```
\newcommand{\longnewglossaryentry}[3]{%
  \glsdoifnoexists{#1}%
  {%
     \bgroup
       \let\@org@newglossaryentryprehook\@newglossaryentryprehook
       \long\def\@newglossaryentryprehook{%
         \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
         \@org@newglossaryentryprehook
       }%
       \renewcommand*{\gls@assign@desc}[1]{%
          \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
          \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
        }
       \gls@defglossaryentry{#1}{#2}%
     \egroup
  }
}
```

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

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

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

```
\newcommand{\longprovideglossaryentry}[3]{%
  \ifglsentryexists{#1}{}%
  {\longnewglossaryentry{#1}{#2}{#3}}%
}
\@onlypreamble{\longprovideglossaryentry}
```

defglossaryentry

```
\gls@defglossaryentry{⟨label⟩}{⟨key-val list⟩}
```

Defines a new entry without checking if it already exists.

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

Prevent any further use of `\GlsSetQuote`:

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

Store label

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

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

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

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

```
\let\@glo@name\@glsnoname
\let\@glo@desc\@glsnodesc
```

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

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

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

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

```
\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.)

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

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

Set the default sort:

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

Set the default counter:

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

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

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

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

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

```
\@gls@initnonumberlist
```

```
\def\@glo@useri{}%
\def\@glo@userii{}%
\def\@glo@useriii{}%
\def\@glo@useriv{}%
\def\@glo@userv{}%
\def\@glo@uservi{}%
```

```
\def\@glo@short{}%
\def\@glo@shortpl{}%
\def\@glo@long{}%
\def\@glo@longpl{}%
```

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

```
\@newglossaryentryprehook
```

Extract key-val information from third parameter:

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

Check there is a default glossary.

```
\ifundef\glsdefaulttype
{%
   \PackageError{glossaries}%
   {No default glossary type (have you used ‘nomain’ by mistake?)}%
   {If you use package option ‘nomain’ you must define
    a new glossary before you can define entries}%
}%
{}%
```

Assign type. This must be fully expandable

```
\gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
\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.

```
\ifcsundef{glolist@\@glo@type}%
{%
    \PackageError{glossaries}%
    {Glossary type ‘\@glo@type’ has not been defined}%
    {You need to define a new glossary type, before making entries
     in it}%
}%
{%
```

Check if it’s an ignored glossary

```
  \ifignoredglossary\@glo@type
    {%
```

The description may be omitted for an entry in an ignored glossary.

```
    \ifx\@glo@desc\@glsnodesc
      \let\@glo@desc\@empty
    \fi
  }%
  {%
  }%
  \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
  \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
    \@glolist@{\@glo@label},}%
}%
```

Initialise level to 0.

```
\gls@level=0\relax
```

Has this entry been assigned a parent?

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

Doesn't have a parent. Set `\glo@`⟨*label*⟩`@parent` to empty.

```
  \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
\else
```

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

```
  \ifdefequal\@glo@label\@glo@parent%
  {%
    \PackageError{glossaries}{Entry '\@glo@label' can't be its own parent}{}%
    \def\@glo@parent{}%
    \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
  }%
  {%
```

Check the parent exists:

```
    \ifglsentryexists{\@glo@parent}%
    {%
```

Parent exists. Set `\glo@`⟨*label*⟩`@parent`.

```
      \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
        \@glo@parent}%
```

Determine level.

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

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

```
      \ifx\@glo@name\@glsnoname
        \expandafter\let\expandafter\@glo@name
         \csname glo@\@glo@parent @name\endcsname
```

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

```
        \ifx\@glo@plural\@gls@default@value
          \expandafter\let\expandafter\@glo@plural
             \csname glo@\@glo@parent @plural\endcsname
        \fi
      \fi
    }%
    {%
```

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

```
      \PackageError{glossaries}%
      {%
        Invalid parent '\@glo@parent'
        for entry '\@glo@label' - parent doesn't exist%
      }%
      {%
        Parent entries must be defined before their children%
      }%
      \def\@glo@parent{}%
      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
    }%
  }%
\fi
```

Set the level for this entry

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

Define commands associated with this entry:

```
\gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
\letcs\@glo@sort{glo@\@glo@label @sortvalue}%
\gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
\expandafter\gls@assign@field\expandafter
   {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
   {\@glo@label}{plural}{\@glo@plural}%
\expandafter\gls@assign@field\expandafter
   {\csname glo@\@glo@label @text\endcsname}%
   {\@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.

```
\ifx\@glo@first\@gls@default@value
  \expandafter\gls@assign@field\expandafter
     {\csname glo@\@glo@label @plural\endcsname}%
     {\@glo@label}{firstpl}{\@glo@firstplural}%
\else
  \expandafter\gls@assign@field\expandafter
     {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
     {\@glo@label}{firstpl}{\@glo@firstplural}%
\fi
```

```
\ifcsundef{@glotype@\@glo@type @counter}%
{%
  \def\@glo@defaultcounter{\glscounter}%
}%
{%
  \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
}%
\gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
\gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
\gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
\gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
\gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
\gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
\gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
\gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
\gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
\gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
\gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
```

```
\ifx\@glo@name\@glsnoname
  \@glsnoname
  \let\@gloname\@gls@default@value
\fi
\gls@assign@field{}{\@glo@label}{name}{\@glo@name}%
```

Set default numberlist if not defined:

```
\ifcsundef{glo@\@glo@label @numberlist}%
{%
  \csxdef{glo@\@glo@label @numberlist}{%
     \noexpand\@gls@missingnumberlist{\@glo@label}}%
}%
{}%
```

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

```
\@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.

```
\def\@glo@@desc{\@glo@first}%
\ifx\@glo@desc\@glo@@desc
  \let\@glo@desc\@glo@first
\fi
\ifx\@glo@desc\@glsnodesc
  \@glsnodesc
  \let\@glodesc\@gls@default@value
\fi
\gls@assign@desc{\@glo@label}%
```

Set the sort key for this entry:

```
\@gls@defsort{\@glo@type}{\@glo@label}%
```

```
\def\@glo@@symbol{\@glo@text}%
\ifx\@glo@symbol\@glo@@symbol
  \let\@glo@symbol\@glo@text
\fi
\gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
\expandafter
  \gls@assign@field\expandafter
  {\csname glo@\@glo@label @symbol\endcsname}
  {\@glo@label}{symbolplural}{\@glo@symbolplural}%
```

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

```
\expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
  \noexpand\global
    \noexpand\let\expandafter\noexpand
      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
}%
\expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
  \noexpand\global
```

```
    \noexpand\let\expandafter\noexpand
      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
  }%
  \csname glo@\@glo@label @flagfalse\endcsname
```

Sort out any cross-referencing if required.

```
  \@glo@autosee
```

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

```
  \ifignoredglossary\@glo@type
  {%
    \csdef{glo@\@glo@label @index}{}%
  }
  {%
    \do@glo@storeentry{\@glo@label}%
  }%
```

Define entry counters if enabled:

```
  \@newglossaryentry@defcounters
```

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

```
  \@newglossaryentryposthook
}
```

\@glo@autosee Automatically implement \glssee.

```
\newcommand*{\@glo@autosee}{%
  \ifdefvoid\@glo@see{}%
  {%
    \protected@edef\@do@glssee{%
      \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
      \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
    \@do@glssee
  }%
  \@glo@autoseehook
}%
```

glo@autoseehook

```
\newcommand*{\@glo@autoseehook}{}
```

aryentryprehook Allow extra information to be added to glossary entries:

```
\newcommand*{\@newglossaryentryprehook}{}
```

ryentryposthook Allow extra information to be added to glossary entries:

```
\newcommand*{\@newglossaryentryposthook}{}
```

try@defcounters

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

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

```
\newcommand*{\glsmoveentry}[2]{%
  \edef\@glo@thislabel{\glsdetoklabel{#1}}%
  \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
  \def\glo@list{,}%
  \forglsentries[\glo@type]{\glo@label}%
   {%

     \ifdefequal\@glo@thislabel\glo@label
       {}{\eappto\glo@list{\glo@label,}}%
   }%
  \cslet{glolist@\glo@type}{\glo@list}%
  \csdef{glo@\@glo@thislabel @type}{#2}%
}
```

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

```
\ifglsxindy
  \newcommand*{\@glossaryentryfield}{\string\\glossentry}
\else
  \newcommand*{\@glossaryentryfield}{\string\glossentry}
\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.)

```
\ifglsxindy
  \newcommand*{\@glossarysubentryfield}{%
    \string\\subglossentry}
\else
  \newcommand*{\@glossarysubentryfield}{%
    \string\subglossentry}
\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.)

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

Escape makeindex/xindy special characters in the label:

```
  \edef\@glo@esclabel{#1}%
  \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

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

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

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

Get the parent, if one exists

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

Write the information to the glossary file.

```
\ifglsxindy
```

Store using `xindy` syntax.

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

Entry doesn't have a parent

```
    \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
     (\string"\@glo@sort\string" %
     \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
    }%
  \else
```

Entry has a parent

```
    \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
      \csname glo@\@glo@parent @index\endcsname
      (\string"\@glo@sort\string" %
      \string"\@glo@prefix\@glossarysubentryfield
         {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
     }%
  \fi
\else
```

Store using `makeindex` syntax.

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

Sanitize `\@glo@prefix`

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

Entry doesn't have a parent

```
    \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
      \@glo@sort\@gls@actualchar\@glo@prefix
      \@glossaryentryfield{\@glo@esclabel}%
    }%
  \else
```

Entry has a parent

```
    \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
      \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
      \@glo@sort\@gls@actualchar\@glo@prefix
      \@glossarysubentryfield
        {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
    }%
  \fi
\fi
}
```

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`

```
\AtBeginDocument{%
  \@ifpackageloaded{amsmath}%
  {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
  {}%
}
\newcommand*{\@gls@ifnotmeasuring}[1]{%
  \ifmeasuring@
  \else
    #1%
  \fi
}
\newcommand*\gls@ifnotmeasuring[1]{#1}
```

`lspatchtabularx` Patch `\TX@trial` (as per David Carlisle's answer in http://tex.stackexchange.com/a/94895). This does nothing if `\TX@trial` hasn't been defined.

```
\def\@gls@patchtabularx#1\hbox#2#3!!{%
  \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
}
\newcommand*\glspatchtabularx{%
 \ifdef\TX@trial
 {%
   \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
   \let\glspatchtabularx\relax
 }%
 {}%
}
```

`\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.

```
\newcommand*{\glsreset}[1]{%
  \gls@ifnotmeasuring
  {%
    \glsdoifexists{#1}%
    {%
       \@glsreset{#1}%
    }%
  }%
}
```

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

```
\newcommand*{\glslocalreset}[1]{%
  \gls@ifnotmeasuring
  {%
    \glsdoifexists{#1}%
    {%
      \@glslocalreset{#1}%
    }%
  }%
}
```

\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.

```
\newcommand*{\glsunset}[1]{%
  \gls@ifnotmeasuring
  {%
    \glsdoifexists{#1}%
    {%
      \@glsunset{#1}%
    }%
  }%
}
```

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

```
\newcommand*{\glslocalunset}[1]{%
  \gls@ifnotmeasuring
  {%
    \glsdoifexists{#1}%
    {%
      \@glslocalunset{#1}%
    }%
  }%
}
```

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

```
\newcommand*{\@glslocalunset}{\@@glslocalunset}
```

@@glslocalunset Local unset without checks.

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

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

```
\newcommand*{\@glsunset}{\@@glsunset}
```

\@@glsunset Global unset without checks.

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

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

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

@@glslocalreset Local reset without checks.

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

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

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

\@@glsreset Global reset without checks.

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

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

\glsresetall

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

As above, but with only a local effect:

lslocalresetall

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

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

\glsunsetall

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

As above, but with only a local effect:

lslocalunsetall

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

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.

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

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

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

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

Enable new entry fields.

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

Disable `\newglossaryentry` in the document environment.

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

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

```
  \newcommand*{\glsentrycurrcount}[1]{%
   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
   {0}{\@gls@entry@field{##1}{currcount}}%
  }%
  \newcommand*{\glsentryprevcount}[1]{%
  \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
   {0}{\@gls@entry@field{##1}{prevcount}}%
  }%
```

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

```
  \renewcommand*{\@glsunset}[1]{%
    \@@glsunset{##1}%
    \@gls@increment@currcount{##1}%
  }%
  \renewcommand*{\@glslocalunset}[1]{%
    \@@glslocalunset{##1}%
    \@gls@local@increment@currcount{##1}%
  }%
  \renewcommand*{\@glsreset}[1]{%
    \@@glsreset{##1}%
    \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
  }%
  \renewcommand*{\@glslocalreset}[1]{%
    \@@glslocalreset{##1}%
    \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
  }%
```

Alter behaviour of `\cgls`. (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.)

```
 \def\@cgls@##1##2[##3]{%
   \ifnum\glsentryprevcount{##2}=1\relax
     \cglsformat{##2}{##3}%
     \glsunset{##2}%
   \else
     \@gls@{##1}{##2}[##3]%
   \fi
 }%
```

Similarly for the analogous commands. No case change plural:

```
 \def\@cglspl@##1##2[##3]{%
   \ifnum\glsentryprevcount{##2}=1\relax
     \cglsplformat{##2}{##3}%
     \glsunset{##2}%
   \else
     \@glspl@{##1}{##2}[##3]%
   \fi
 }%
```

First letter uppercase singular:

```
 \def\@cGls@##1##2[##3]{%
   \ifnum\glsentryprevcount{##2}=1\relax
     \cGlsformat{##2}{##3}%
     \glsunset{##2}%
   \else
     \@Gls@{##1}{##2}[##3]%
   \fi
```

```
}%
```

First letter uppercase plural:

```
\def\@cGlspl@##1##2[##3]{%
  \ifnum\glsentryprevcount{##2}=1\relax
    \cGlsplformat{##2}{##3}%
    \glsunset{##2}%
  \else
    \@Glspl@{##1}{##2}[##3]%
  \fi
}%
```

Write information to aux file at the end of the document

```
 \AtEndDocument{\@gls@write@entrycounts}%
```

Fetch previous count information from aux file. (No check here to determine if the entry is still defined.)

```
 \renewcommand*{\@gls@entry@count}[2]{%
   \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
 }%
```

`\glsenableentrycount` may only be used once and only in the preamble.

```
 \let\glsenableentrycount\relax
}
\@onlypreamble\glsenableentrycount
```

ement@currcount

```
\newcommand*{\@gls@increment@currcount}[1]{%
 \csxdef{glo@\glsdetoklabel{#1}@currcount}{%
  \number\numexpr\glsentrycurrcount{#1}+1}%
}
```

ement@currcount

```
\newcommand*{\@gls@local@increment@currcount}[1]{%
 \csedef{glo@\glsdetoklabel{#1}@currcount}{%
  \number\numexpr\glsentrycurrcount{#1}+1}%
}
```

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.)

```
\newcommand*{\@gls@write@entrycounts}{%
 \immediate\write\@auxout
   {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
 \forallglsentries{\@glsentry}{%
   \ifglsused{\@glsentry}%
   {\immediate\write\@auxout
     {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}}%
   {}%
```

```
  }%
}
```

gls@entry@count Default behaviour is to ignore arguments. Activated by \glsenableentrycount.

```
\newcommand*{\@gls@entry@count}[2]{}
```

\cgls 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.)

```
\newrobustcmd*{\cgls}{\@gls@hyp@opt\@cgls}
```

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

```
\newcommand*{\@cgls}[2][]{%
  \new@ifnextchar[{\@cgls@{#1}{#2}}{\@cgls@{#1}{#2}[]}%
}
```

\@cgls@ Read in the final optional argument. This defaults to same behaviour as \gls but issues a warning.

```
\def\@cgls@#1#2[#3]{%
 \GlossariesWarning{\string\cgls\space is defaulting to
   \string\gls\space since you haven't enabled entry counting}%
 \@gls@{#1}{#2}[#3]%
}
```

\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.

```
\newcommand*{\cglsformat}[2]{%
  \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
}
```

\cGls 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.)

```
\newrobustcmd*{\cGls}{\@gls@hyp@opt\@cGls}
```

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

```
\newcommand*{\@cGls}[2][]{%
  \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2}[]}%
}
```

\@cGls@ Read in the final optional argument. This defaults to same behaviour as \Gls but issues a warning.

```
\def\@cGls@#1#2[#3]{%
 \GlossariesWarning{\string\cGls\space is defaulting to
   \string\Gls\space since you haven't enabled entry counting}%
 \@Gls@{#1}{#2}[#3]%
}
```

\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.

```
\newcommand*{\cGlsformat}[2]{%
  \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
}
```

\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.)

```
\newrobustcmd*{\cglspl}{\@gls@hyp@opt\@cglspl}
```

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

```
\newcommand*{\@cglspl}[2][]{%
  \new@ifnextchar[{\@cglspl@{#1}{#2}}{\@cglspl@{#1}{#2}[]}%
}
```

\@cglspl@ Read in the final optional argument. This defaults to same behaviour as \glspl but issues a warning.

```
\def\@cglspl@#1#2[#3]{%
\GlossariesWarning{\string\cglspl\space is defaulting to
  \string\glspl\space since you haven't enabled entry counting}%
\@glspl@{#1}{#2}[#3]%
}
```

\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.

```
\newcommand*{\cglsplformat}[2]{%
  \ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}#2%
}
```

\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.)

```
\newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

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

```
\newcommand*{\@cGlspl}[2][]{%
  \new@ifnextchar[{\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[]}%
}
```

\@cGlspl@ Read in the final optional argument. This defaults to same behaviour as \Glspl but issues a warning.

```
\def\@cGlspl@#1#2[#3]{%
\GlossariesWarning{\string\cGlspl\space is defaulting to
  \string\Glspl\space since you haven't enabled entry counting}%
\@Glspl@{#1}{#2}[#3]%
}
```

\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.

```
\newcommand*{\cGlsplformat}[2]{%
  \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
}
```

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

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

\loadglsentries can only be used in the preamble:

```
\@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

```
\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

[1] and 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

```
\newcommand*{\glsentryfmt}{%
  \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
}
```

Format that provides backwards compatibility:

```
\newcommand*{\@@gls@default@entryfmt}[2]{%
  \ifdefempty\glscustomtext
  {%
    \glsifplural
    {%
```

Plural form

```
      \glscapscase
      {%
```

Don't adjust case

```
        \ifglsused\glslabel
        {%
```

Subsequent use

```
          #2{\glsentryplural{\glslabel}}%
            {\glsentrydescplural{\glslabel}}%
            {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
        }%
        {%
```

First use

```
          #1{\glsentryfirstplural{\glslabel}}%
            {\glsentrydescplural{\glslabel}}%
            {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
        }%
      }%
      {%
```

Make first letter upper case

```
        \ifglsused\glslabel
        {%
```

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.)

```
          \ifbool{glscompatible-3.07}%
          {%
            \protected@edef\@glo@etext{%
              #2{\glsentryplural{\glslabel}}%
                {\glsentrydescplural{\glslabel}}%
                {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
            \xmakefirstuc\@glo@etext
          }%
```

```
     {%
      #2{\Glsentryplural{\glslabel}}%
       {\glsentrydescplural{\glslabel}}%
       {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
     }%
    }%
    {%
```
First use
```
     \ifbool{glscompatible-3.07}%
     {%
      \protected@edef\@glo@etext{%
       #1{\glsentryfirstplural{\glslabel}}%
        {\glsentrydescplural{\glslabel}}%
        {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
      \xmakefirstuc\@glo@etext
     }%
     {%
      #1{\Glsentryfirstplural{\glslabel}}%
       {\glsentrydescplural{\glslabel}}%
       {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
     }%
    }%
   }%
   {%
```
Make all upper case
```
    \ifglsused\glslabel
    {%
```
Subsequent use
```
     \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
      {\glsentrydescplural{\glslabel}}%
      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
    }%
    {%
```
First use
```
     \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
      {\glsentrydescplural{\glslabel}}%
      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
    }%
   }%
  }%
  {%
```
Singular form
```
   \glscapscase
   {%
```
Don't adjust case
```
    \ifglsused\glslabel
```

```
          {%
```

Subsequent use

```
            #2{\glsentrytext{\glslabel}}%
              {\glsentrydesc{\glslabel}}%
              {\glsentrysymbol{\glslabel}}{\glsinsert}%
          }%
          {%
```

First use

```
            #1{\glsentryfirst{\glslabel}}%
              {\glsentrydesc{\glslabel}}%
              {\glsentrysymbol{\glslabel}}{\glsinsert}%
          }%
        }%
        {%
```

Make first letter upper case

```
          \ifglsused\glslabel
          {%
```

Subsequent use

```
            \ifbool{glscompatible-3.07}%
            {%
              \protected@edef\@glo@etext{%
                #2{\glsentrytext{\glslabel}}%
                  {\glsentrydesc{\glslabel}}%
                  {\glsentrysymbol{\glslabel}}{\glsinsert}}%
              \xmakefirstuc\@glo@etext
            }%
            {%
              #2{\Glsentrytext{\glslabel}}%
                {\glsentrydesc{\glslabel}}%
                {\glsentrysymbol{\glslabel}}{\glsinsert}%
            }%
          }%
          {%
```

First use

```
            \ifbool{glscompatible-3.07}%
            {%
              \protected@edef\@glo@etext{%
                #1{\glsentryfirst{\glslabel}}%
                  {\glsentrydesc{\glslabel}}%
                  {\glsentrysymbol{\glslabel}}{\glsinsert}}%
                \xmakefirstuc\@glo@etext
            }%
            {%
              #1{\Glsentryfirst{\glslabel}}%
                {\glsentrydesc{\glslabel}}%
                {\glsentrysymbol{\glslabel}}{\glsinsert}%
```

```
          }%
        }%
      }%
      {%
```
Make all upper case
```
        \ifglsused\glslabel
        {%
```
Subsequent use
```
          \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}%
            {\glsentrydesc{\glslabel}}%
            {\glsentrysymbol{\glslabel}}{\glsinsert}}%
        }%
        {%
```
First use
```
          \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}%
            {\glsentrydesc{\glslabel}}%
            {\glsentrysymbol{\glslabel}}{\glsinsert}}%
        }%
      }%
    }%
  }%
  {%
```
Custom text provided in \glsdisp
```
    \ifglsused{\glslabel}%
    {%
```
Subsequent use
```
      #2{\glscustomtext}%
        {\glsentrydesc{\glslabel}}%
        {\glsentrysymbol{\glslabel}}{}%
    }%
    {%
```
First use
```
      #1{\glscustomtext}%
        {\glsentrydesc{\glslabel}}%
        {\glsentrysymbol{\glslabel}}{}%
    }%
  }%
}
```
\glsgenentryfmt Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.
```
\newcommand*{\glsgenentryfmt}{%
  \ifdefempty\glscustomtext
  {%
    \glsifplural
    {%
```

Plural form

```
    \glscapscase
    {%
```

Don't adjust case

```
      \ifglsused\glslabel
      {%
```

Subsequent use

```
        \glsentryplural{\glslabel}\glsinsert
      }%
      {%
```

First use

```
        \glsentryfirstplural{\glslabel}\glsinsert
      }%
    }%
    {%
```

Make first letter upper case

```
      \ifglsused\glslabel
      {%
```

Subsequent use.

```
         \Glsentryplural{\glslabel}\glsinsert
      }%
      {%
```

First use

```
         \Glsentryfirstplural{\glslabel}\glsinsert
      }%
    }%
    {%
```

Make all upper case

```
      \ifglsused\glslabel
      {%
```

Subsequent use

```
        \mfirstucMakeUppercase
           {\glsentryplural{\glslabel}\glsinsert}%
      }%
      {%
```

First use

```
        \mfirstucMakeUppercase
           {\glsentryfirstplural{\glslabel}\glsinsert}%
      }%
    }%
  }%
  {%
```

Singular form

```
    \glscapscase
    {%
```

Don't adjust case

```
      \ifglsused\glslabel
      {%
```

Subsequent use

```
        \glsentrytext{\glslabel}\glsinsert
      }%
      {%
```

First use

```
        \glsentryfirst{\glslabel}\glsinsert
      }%
    }%
    {%
```

Make first letter upper case

```
      \ifglsused\glslabel
      {%
```

Subsequent use

```
         \Glsentrytext{\glslabel}\glsinsert
      }%
      {%
```

First use

```
        \Glsentryfirst{\glslabel}\glsinsert
      }%
    }%
    {%
```

Make all upper case

```
      \ifglsused\glslabel
      {%
```

Subsequent use

```
        \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
      }%
      {%
```

First use

```
        \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
      }%
    }%
  }%
}%
{%
```

Custom text provided in `\glsdisp`. (The insert is most likely to be empty at this point.)

```
  \glscustomtext\glsinsert
  }%
}
```

`\glsgenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```
\newcommand*{\glsgenacfmt}{%
  \ifdefempty\glscustomtext
  {%
    \ifglsused\glslabel
    {%
```

Subsequent use:

```
      \glsifplural
      {%
```

Subsequent plural form:

```
        \glscapscase
        {%
```

Subsequent plural form, don't adjust case:

```
          \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent plural form, make first letter upper case:

```
          \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent plural form, all caps:

```
          \mfirstucMakeUppercase
            {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
        }%
      }%
      {%
```

Subsequent singular form

```
        \glscapscase
        {%
```

Subsequent singular form, don't adjust case:

```
          \acronymfont{\glsentryshort{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent singular form, make first letter upper case:

```
          \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent singular form, all caps:

```
          \mfirstucMakeUppercase
            {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
          }%
        }%
      }%
      {%
```

First use:

```
        \glsifplural
        {%
```

First use plural form:

```
          \glscapscase
          {%
```

First use plural form, don't adjust case:

```
            \genplacrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use plural form, make first letter upper case:

```
            \Genplacrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use plural form, all caps:

```
            \mfirstucMakeUppercase
              {\genplacrfullformat{\glslabel}{\glsinsert}}%
          }%
        }%
        {%
```

First use singular form

```
          \glscapscase
          {%
```

First use singular form, don't adjust case:

```
            \genacrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use singular form, make first letter upper case:

```
            \Genacrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use singular form, all caps:

```
            \mfirstucMakeUppercase
             {\genacrfullformat{\glslabel}{\glsinsert}}%
          }%
        }%
      }%
    }%
    {%
```

User supplied text.

```
    \glscustomtext
  }%
}
```

genacrfullformat

```
\genacrfullformat{⟨label⟩}{⟨insert⟩}
```

The full format used by \glsgenacfmt (singular).

```
\newcommand*{\genacrfullformat}[2]{%
   \glsentrylong{#1}#2\space
   (\protect\firstacronymfont{\glsentryshort{#1}})%
}
```

Genacrfullformat

```
\Genacrfullformat{⟨label⟩}{⟨insert⟩}
```

As above but makes the first letter upper case.

```
\newcommand*{\Genacrfullformat}[2]{%
  \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
  \xmakefirstuc\gls@text
}
```

nplacrfullformat

```
\genplacrfullformat{⟨label⟩}{⟨insert⟩}
```

The full format used by \glsgenacfmt (plural).

```
\newcommand*{\genplacrfullformat}[2]{%
   \glsentrylongpl{#1}#2\space
   (\protect\firstacronymfont{\glsentryshortpl{#1}})%
}
```

nplacrfullformat

```
\Genplacrfullformat{⟨label⟩}{⟨insert⟩}
```

As above but makes the first letter upper case.

```
\newcommand*{\Genplacrfullformat}[2]{%
  \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
  \xmakefirstuc\gls@text
}
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

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

\glsdisplay Deprecated. Kept for backward compatibility.

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

\defglsdisplay Deprecated. Kept for backward compatibility.

```
\newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
  \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
  Use \string\defglsentryfmt\space instead}%
  \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
  \edef\@gls@doentrydef{%
    \noexpand\defglsentryfmt[#1]{%
      \noexpand\ifcsdef{gls@#1@displayfirst}%
      {%
        \noexpand\@@gls@default@entryfmt
          {\noexpand\csuse{gls@#1@displayfirst}}%
          {\noexpand\csuse{gls@#1@display}}%
      }%
      {%
        \noexpand\@@gls@default@entryfmt
          {\noexpand\glsdisplayfirst}%
          {\noexpand\csuse{gls@#1@display}}%
      }%
    }%
  }%
  \@gls@doentrydef
}
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
\newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
  \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
  Use \string\defglsentryfmt\space instead}%
  \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
  \edef\@gls@doentrydef{%
    \noexpand\defglsentryfmt[#1]{%
      \noexpand\ifcsdef{gls@#1@display}%
      {%
        \noexpand\@@gls@default@entryfmt
          {\noexpand\csuse{gls@#1@displayfirst}}%
          {\noexpand\csuse{gls@#1@display}}%
      }%
      {%

        \noexpand\@@gls@default@entryfmt
          {\noexpand\csuse{gls@#1@displayfirst}}%
          {\noexpand\glsdisplay}%
      }%
    }%
  }%
  \@gls@doentrydef
}
```

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 `\defglsentryfmt`). 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.

```
\define@key{glslink}{counter}{%
  \ifcsundef{c@#1}%
  {%
    \PackageError{glossaries}%
    {There is no counter called '#1'}%
    {%
       The counter key should have the name of a valid counter
       as its value%
    }%
  }%
  {%
    \def\@gls@counter{#1}%
  }%
}
```

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

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

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

Initialise hyper key.

```
\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.

```
\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.

```
\newcommand*{\glslinkvar}[3]{#1}
```

\glsifhyper Now deprecated.

```
\newcommand*{\glsifhyper}[2]{%
 \glslinkvar{#1}{#2}{#1}%
 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
  you mean \string\glsifhyperon\space or \string\glslinkvar?}%
}
```

\@gls@hyp@opt Used by the commands such as \glslink to determine whether to modify the hyper option.

```
\newcommand*{\@gls@hyp@opt}[1]{%
 \let\glslinkvar\@firstofthree
 \let\@gls@hyp@opt@cs#1\relax
 \@ifstar{\s@gls@hyp@opt}%
 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
}
```

\s@gls@hyp@opt Starred version

```
\newcommand*{\s@gls@hyp@opt}[1][]{%
 \let\glslinkvar\@secondofthree
 \@gls@hyp@opt@cs[hyper=false,#1]}
```

\p@gls@hyp@opt Plus version

```
\newcommand*{\p@gls@hyp@opt}[1][]{%
 \let\glslinkvar\@thirdofthree
 \@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

```
\newrobustcmd*{\glslink}{%
 \@gls@hyp@opt\@gls@@link
}
```

\@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.

```
\newcommand*{\@gls@@link}[3][]{%
  \glsdoifexistsordo{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax
    \@gls@link[#1]{#2}{#3}%
  }{%
```

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

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

glspostlinkhook

```
\newcommand*{\glspostlinkhook}{}
```

checkfirsthyper Check for first use and switch off hyper key if hyperlink not wanted. (Should be off if first use and hyper=false is on or if first use and both the entry is in an acronym list and the acrfootnote setting is on.) This assumes the glossary type is stored in `\glstype` and the label is stored in `\glslabel`.

```
\newcommand*{\@gls@link@checkfirsthyper}{%
  \ifglsused{\glslabel}%
  {%
  }%
  {%
    \gls@checkisacronymlist\glstype
    \ifglshyperfirst
      \if@glsisacronymlist
        \ifglsacrfootnote
           \KV@glslink@hyperfalse
        \fi
      \fi
    \else
       \KV@glslink@hyperfalse
    \fi
  }%
```

Allow user to hook into this

```
  \glslinkcheckfirsthyperhook
}
```

`kfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro

```
\newcommand*{\glslinkcheckfirsthyperhook}{}
```

`linkpostsetkeys`

```
\newcommand*{\glslinkpostsetkeys}{}
```

`\glsifhyperon` Check the value of the hyper key:

```
\newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}
```

`ablehyperinlist` Disable hyperlink if in the “nohyper” list.

```
\newcommand*{\do@glsdisablehyperinlist}{%
  \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
    {\KV@glslink@hyperfalse}{}%
}
```

`lt@glslink@opts` Hook to set default options for `\@glslink`.

```
\newcommand*{\@gls@setdefault@glslink@opts}{}
```

`\@gls@link`

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

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

```
    \leavevmode
    \edef\glslabel{\glsdetoklabel{#2}}%
```

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

```
    \def\@gls@link@opts{#1}%
    \let\@gls@link@label\glslabel
    \def\@glsnumberformat{glsnumberformat}%
    \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
```

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

```
    \edef\glstype{\csname glo@\glslabel @type\endcsname}%
```

Save original setting

```
    \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Set defaults:

```
    \@gls@setdefault@glslink@opts
```

Switch off hyper setting if the glossary type has been identified in nohyperlist.

```
    \do@glsdisablehyperinlist
```

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`.

```
    \do@gls@link@checkfirsthyper
    \setkeys{glslink}{#1}%
```

Add a hook for the user to customise things after the keys have been set.

```
    \glslinkpostsetkeys
```

Store the entry's counter in `\theglsentrycounter`

```
    \@gls@saveentrycounter
```

Define sort key if necessary:

```
    \@gls@setsort{\glslabel}%
```

(De-tok'ing done by `\@@do@wrglossary`)

```
    \@do@wrglossary{#2}%
    \ifKV@glslink@hyper
      \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
    \else
      \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
    \fi
```

Restore original setting

```
    \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
}
```

`\glolinkprefix`

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

`glsentrycounter` Set default value of entry counter

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

`aveentrycounter` Need to check if using equation counter in align environment:

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

Are we using equation counter?

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

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

```
    \ifcsundef{xatlevel@}%
    {%
      \edef\theglsentrycounter{\expandafter\noexpand
        \csname the\@gls@counter\endcsname}%
    }%
    {%
      \ifx\xatlevel@\@empty
        \edef\theglsentrycounter{\expandafter\noexpand
          \csname the\@gls@counter\endcsname}%
      \else
        \savecounters@
        \advance\c@equation by 1\relax
          \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%
```

Check if hyperref version of this counter

```
        \ifcsundef{theH\@gls@counter}%
        {%
           \def\@gls@Hcounter{\theglsentrycounter}%
        }%
        {%
          \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
        }%
        \protected@edef\theHglsentrycounter{\@gls@Hcounter}%
        \restorecounters@
      \fi
    }%
  }%
  {%
```

Not using equation counter so no special measures:

```
    \edef\theglsentrycounter{\expandafter\noexpand
      \csname the\@gls@counter\endcsname}%
  }%
```

Check if hyperref version of this counter

```
  \ifx\@gls@Hcounter\@empty
    \ifcsundef{theH\@gls@counter}%
    {%
       \def\theHglsentrycounter{\theglsentrycounter}%
    }%
    {%
      \protected@edef\theHglsentrycounter{\expandafter\noexpand
        \csname theH\@gls@counter\endcsname}%
    }%
  \fi
}
```

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.

```
\def\@set@glo@numformat#1#2#3#4{%
  \expandafter\@glo@check@mkidxrangechar#3\@nil
  \protected@edef#1{%
    \@glo@prefix setentrycounter[#4]{#2}%
    \expandafter\string\csname\@glo@suffix\endcsname
  }%
  \@gls@checkmkidxchars#1%
}
```

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

```
\def\@glo@check@mkidxrangechar#1#2\@nil{%
\if#1(\relax
  \def\@glo@prefix{(}%
  \if\relax#2\relax
    \def\@glo@suffix{glsnumberformat}%
  \else
    \def\@glo@suffix{#2}%
  \fi
\else
  \if#1)\relax
    \def\@glo@prefix{)}%
    \if\relax#2\relax
      \def\@glo@suffix{glsnumberformat}%
    \else
      \def\@glo@suffix{#2}%
  \fi
  \else
    \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
  \fi
\fi}
```

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

```
\newcommand*{\@gls@escbsdq}[1]{%
  \def\@gls@checkedmkidx{}%
  \let\gls@xdystring=#1\relax
  \@onelevel@sanitize\gls@xdystring
  \edef\do@gls@xdycheckbackslash{%
    \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
    \@backslashchar\@backslashchar\noexpand\null}%
  \do@gls@xdycheckbackslash
  \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
  \def\@gls@checkedmkidx{}%
  \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
  \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
```

Unsanitize `\gls@numberpage`, `\gls@alphpage`, `\gls@Alphpage` and `\glsromanpage` (thanks to David Carlise for the suggestion.)

```
  \@for\@gls@tmp:=\gls@protected@pagefmts\do
  {%
    \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\\expandonce\@gls@tmp}%
    \@onelevel@sanitize\@gls@sanitized@tmp
    \edef\gls@dosubst{%
      \noexpand\DTLsubstituteall\noexpand\gls@xdystring
      {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
    }%
    \gls@dosubst
  }%
```

Assign to required control sequence

```
  \let#1=\gls@xdystring
```

```
}
```

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

checkmkidxchars

```
\newcommand{\@gls@checkmkidxchars}[1]{%
  \ifglsxindy
    \@gls@escbsdq{#1}%
  \else
    \def\@gls@checkedmkidx{}%
    \expandafter\@gls@checkquote#1\@nil""\null
    \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
    \def\@gls@checkedmkidx{}%
    \expandafter\@gls@checkescquote#1\@nil\"\"\null
    \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
    \def\@gls@checkedmkidx{}%
    \expandafter\@gls@checkescactual#1\@nil\?\?\null
    \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
    \def\@gls@checkedmkidx{}%
    \expandafter\@gls@checkactual#1\@nil??\null
    \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
    \def\@gls@checkedmkidx{}%
    \expandafter\@gls@checkbar#1\@nil||\null
    \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
    \def\@gls@checkedmkidx{}%
    \expandafter\@gls@checkescbar#1\@nil\|\|\null
    \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
    \def\@gls@checkedmkidx{}%
    \expandafter\@gls@checklevel#1\@nil!!\null
    \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
  \fi
}
```

Update the control sequence and strip trailing `\@nil`:

s@updatechecked

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

\@gls@tmpb Define temporary token

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

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

```
\def\@gls@checkquote#1"#2"#3\null{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#1}%
  \ifx\null#2\null
   \ifx\null#3\null
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
    \def\@@gls@checkquote{\relax}%
   \else
```

```
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
        \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
      \def\@@gls@checkquote{\@gls@checkquote#3\null}%
    \fi
  \else
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
      \@gls@quotechar\@gls@quotechar}%
    \ifx\null#3\null
      \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
    \else
      \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
    \fi
  \fi
  \@@gls@checkquote
}
```

s@checkescquote Do the same for \":

```
\def\@gls@checkescquote#1\"#2\"#3\null{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#1}%
  \ifx\null#2\null
    \ifx\null#3\null
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
      \def\@@gls@checkescquote{\relax}%
    \else
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
        \@gls@quotechar\string\"\@gls@quotechar
        \@gls@quotechar\string\"\@gls@quotechar}%
      \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
    \fi
  \else
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
      \@gls@quotechar\string\"\@gls@quotechar}%
    \ifx\null#3\null
      \def\@@gls@checkescquote{\@gls@checkescquote#2\"\"\null}%
    \else
      \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
    \fi
  \fi
\@@gls@checkescquote
}
```

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

```
\def\@gls@checkescactual#1\?#2\?#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null
  \ifx\null#3\null
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
    \def\@@gls@checkescactual{\relax}%
   \else
     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
     \@gls@quotechar\string\"\@gls@actualchar
     \@gls@quotechar\string\"\@gls@actualchar}%
     \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
   \fi
  \else
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\string\"\@gls@actualchar}%
    \ifx\null#3\null
      \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
    \else
      \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
   \fi
  \fi
\@@gls@checkescactual
}
```

gls@checkescbar Similarly for `\|`:

```
\def\@gls@checkescbar#1\|#2\|#3\null{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#1}%
  \ifx\null#2\null
   \ifx\null#3\null
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
    \def\@@gls@checkescbar{\relax}%
   \else
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
      \@gls@quotechar\string\"\@gls@encapchar
      \@gls@quotechar\string\"\@gls@encapchar}%
    \def\@@gls@checkescbar{\@gls@checkescbar#3\null}%
   \fi
  \else
   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
     \@gls@quotechar\string\"\@gls@encapchar}%
   \ifx\null#3\null
    \def\@@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
   \else
    \def\@@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
   \fi
  \fi
\@@gls@checkescbar
}
```

s@checkesclevel Similarly for `\!`:

```
\def\@gls@checkesclevel#1\!#2\!#3\null{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#1}%
```

```
 \ifx\null#2\null
  \ifx\null#3\null
   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
   \def\@@gls@checkesclevel{\relax}%
  \else
   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
     \@gls@quotechar\string\"\@gls@levelchar
     \@gls@quotechar\string\"\@gls@levelchar}%
   \def\@@gls@checkesclevel{\@gls@checkesclevel#3\null}%
  \fi
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\string\"\@gls@levelchar}%
  \ifx\null#3\null
   \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
  \else
   \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
  \fi
 \fi
\@@gls@checkesclevel
}
```

\@gls@checkbar and for |:

```
\def\@gls@checkbar#1|#2|#3\null{%
 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
 \toks@={#1}%
 \ifx\null#2\null
  \ifx\null#3\null
   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
   \def\@@gls@checkbar{\relax}%
  \else
   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
     \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
   \def\@@gls@checkbar{\@gls@checkbar#3\null}%
  \fi
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\@gls@encapchar}%
  \ifx\null#3\null
    \def\@@gls@checkbar{\@gls@checkbar#2||\null}%
  \else
    \def\@@gls@checkbar{\@gls@checkbar#2|#3\null}%
  \fi
 \fi
 \@@gls@checkbar
}
```

@gls@checklevel and for !:

```
\def\@gls@checklevel#1!#2!#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null
  \ifx\null#3\null
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
    \def\@@gls@checklevel{\relax}%
  \else
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
    \def\@@gls@checklevel{\@gls@checklevel#3\null}%
  \fi
\else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
  \@gls@quotechar\@gls@levelchar}%
  \ifx\null#3\null
    \def\@@gls@checklevel{\@gls@checklevel#2!!\null}%
  \else
    \def\@@gls@checklevel{\@gls@checklevel#2!#3\null}%
  \fi
\fi
\@@gls@checklevel
}
```

`gls@checkactual` and for ?:

```
\def\@gls@checkactual#1?#2?#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null
  \ifx\null#3\null
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
    \def\@@gls@checkactual{\relax}%
   \else
    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
      \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
    \def\@@gls@checkactual{\@gls@checkactual#3\null}%
   \fi
  \else
   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
     \@gls@quotechar\@gls@actualchar}%
   \ifx\null#3\null
     \def\@@gls@checkactual{\@gls@checkactual#2??\null}%
   \else
     \def\@@gls@checkactual{\@gls@checkactual#2?#3\null}%
   \fi
  \fi
\@@gls@checkactual
}
```

`s@xdycheckquote` As before but for use with xindy

```
\def\@gls@xdycheckquote#1"#2"#3\null{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#1}%
  \ifx\null#2\null
    \ifx\null#3\null
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
      \def\@@gls@xdycheckquote{\relax}%
     \else
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
        \string\"\string\"}%
      \def\@@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
     \fi
    \else
     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
       \string\"}%
     \ifx\null#3\null
       \def\@@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
     \else
       \def\@@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
     \fi
    \fi
  \@@gls@xdycheckquote
}
```

ycheckbackslash Need to escape all backslashes for `xindy`. Define command that will define `\@gls@xdycheckbackslash`

```
\edef\def@gls@xdycheckbackslash{%
 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
   ##2\@backslashchar##3\noexpand\null{%
  \noexpand\@gls@tmpb=\noexpand\expandafter
    {\noexpand\@gls@checkedmkidx}%
  \noexpand\toks@={##1}%
  \noexpand\ifx\noexpand\null##2\noexpand\null
   \noexpand\ifx\noexpand\null##3\noexpand\null
    \noexpand\edef\noexpand\@gls@checkedmkidx{%
       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
    \noexpand\def\noexpand\@@gls@xdycheckbackslash{\relax}%
   \noexpand\else
    \noexpand\edef\noexpand\@gls@checkedmkidx{%
      \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
    \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
  \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
     \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
   \noexpand\fi
  \noexpand\else
   \noexpand\edef\noexpand\@gls@checkedmkidx{%
     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
   \@backslashchar\@backslashchar}%
 \noexpand\ifx\noexpand\null##3\noexpand\null
   \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
```

```
      \noexpand\@gls@xdycheckbackslash##2\@backslashchar
      \@backslashchar\noexpand\null}%
   \noexpand\else
     \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
        \noexpand\@gls@xdycheckbackslash##2\@backslashchar
           ##3\noexpand\null}%
   \noexpand\fi
  \noexpand\fi
  \noexpand\@@gls@xdycheckbackslash
 }%
}
```

Now go ahead and define \@gls@xdycheckbackslash

```
\def@gls@xdycheckbackslash
```

lsdohypertarget

```
\newlength\gls@tmplen
\newcommand*{\glsdohypertarget}[2]{%
  \@glsshowtarget{#1}%
  \settoheight{\gls@tmplen}{#2}%
  \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
}
```

\glsdohyperlink

```
\newcommand*{\glsdohyperlink}[2]{%
 \@glsshowtarget{#1}%
 \hyperlink{#1}{#2}%
}
```

lsdonohyperlink

```
\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.

```
\ifcsundef{hyperlink}%
{%
  \let\@glslink\glsdonohyperlink
}%
{%
  \let\@glslink\glsdohyperlink
}
```

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

```
\ifcsundef{hypertarget}%
{%
  \let\@glstarget\@secondoftwo
}%
```

```
{%
  \let\@glstarget\glsdohypertarget
}
```

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

glsdisablehyper

```
\newcommand{\glsdisablehyper}{%
  \KV@glslink@hyperfalse
  \let\@glslink\glsdonohyperlink
  \let\@glstarget\@secondoftwo
}
```

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

\glsenablehyper

```
\newcommand{\glsenablehyper}{%
 \KV@glslink@hypertrue
 \let\@glslink\glsdohyperlink
 \let\@glstarget\glsdohypertarget
}
```

Provide some convenience commands if not already defined:

```
\providecommand{\@firstofthree}[3]{#1}
\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

```
\newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

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

\@gls

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

\@gls@ Read in the final optional argument:

```
\def\@gls@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glscustomtext\@empty
    \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```
    \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.

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

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%

  \glspostlinkhook
}
```

\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

```
\newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
\newcommand*{\@Gls}[2][]{%
  \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}%
}
```

\@Gls@ Read in the final optional argument:

```
\def\@Gls@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glscustomtext\@empty
    \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
    \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.

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

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%

  \glspostlinkhook
}
```

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

`\GLS`

```
\newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
\newcommand*{\@GLS}[2][]{%
  \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
}
```

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

```
\def\@GLS@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glscustomtext\@empty
    \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`). Note that `\@gls@link` sets `\glstype`.

```
    \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.

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

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%

  \glspostlinkhook
}
```

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

\glspl

```
\newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

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

\@glspl@ Read in the final optional argument:

```
\def\@glspl@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glscustomtext\@empty
    \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```
    \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.

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

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%

  \glspostlinkhook
}
```

\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

```
\newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

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

\@Glspl@ Read in the final optional argument:

```
\def\@Glspl@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glscustomtext\@empty
    \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.

```
    \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.

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

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
  \glspostlinkhook
}
```

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

\GLSpl

```
\newrobustcmd*{\GLSpl}{\@gls@hyp@opt\@GLSpl}
```

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

```
\newcommand*{\@GLSpl}[2][]{%
  \new@ifnextchar[{\@GLSpl@{#1}{#2}}{\@GLSpl@{#1}{#2}[]}%
}
```

\@GLSpl Read in the final optional argument:

```
\def\@GLSpl@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glscustomtext\@empty
    \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
    \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.

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

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%

  \glspostlinkhook
}
```

\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:

```
\newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}
```

Defined the un-starred form.

\@glsdisp

```
\newcommand*{\@glsdisp}[3][]{%
  \glsdoifexists{#2}{%

    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \def\glscustomtext{#3}%
    \def\glsinsert{}%
```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```
    \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.

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

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
  \glspostlinkhook
}
```

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.

```
\newcommand*{\@gls@link@nocheckfirsthyper}{}
```

@gls@field@link

```
\newcommand{\@gls@field@link}[3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \@gls@link[#1]{#2}{#3}%
  }%
  \glspostlinkhook
}
```

\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

```
\newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

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

Read in the final optional argument:

```
\def\@glstext@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentrytext{#2}#3}%
}
```

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

\GLStext

```
\newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

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

Read in the final optional argument:

```
\def\@GLStext@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrytext{#2}#3}}%
}
```

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

\Glstext

```
\newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
\newcommand*{\@Glstext}[2][]{%
  \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glstext@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentrytext{#2}#3}%
}
```

\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

```
\newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
\newcommand*{\@glsfirst}[2][]{%
  \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsfirst@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
}
```

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

\Glsfirst

```
\newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
\newcommand*{\@Glsfirst}[2][]{%
  \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsfirst@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
}
```

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

`\GLSfirst`

```
\newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
\newcommand*{\@GLSfirst}[2][]{%
  \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSfirst@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
}
```

`\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`

```
\newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
\newcommand*{\@glsplural}[2][]{%
  \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
}
```

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

`\Glsplural`

```
\newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
\newcommand*{\@Glsplural}[2][]{%
  \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
}
```

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

`\GLSplural`

```
\newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
\newcommand*{\@GLSplural}[2][]{%
  \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
}
```

\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

```
\newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
\newcommand*{\@glsfirstplural}[2][]{%
  \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsfirstplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
}
```

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

\Glsfirstplural

```
\newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
\newcommand*{\@Glsfirstplural}[2][]{%
  \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsfirstplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
}
```

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

\GLSfirstplural

```
\newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
\newcommand*{\@GLSfirstplural}[2][]{%
  \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSfirstplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
}
```

`\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`

```
\newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
\newcommand*{\@glsname}[2][]{%
  \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsname@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
}
```

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

`\Glsname`

```
\newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

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

Read in the final optional argument:

```
\def\@Glsname@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
}
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
\newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
\newcommand*{\@GLSname}[2][]{%
  \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSname@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
}
```

`\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`

```
\newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
\newcommand*{\@glsdesc}[2][]{%
  \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsdesc@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
}
```

`\Glsdesc` behaves like `\glsdesc` except that the first letter is converted to uppercase.

`\Glsdesc`

```
\newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
\newcommand*{\@Glsdesc}[2][]{%
  \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsdesc@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
}
```

`\GLSdesc` behaves like `\glsdesc` except that the link text is converted to uppercase.

`\GLSdesc`

```
\newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
\newcommand*{\@GLSdesc}[2][]{%
  \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSdesc@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
}
```

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

`\glsdescplural`

```
\newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
\newcommand*{\@glsdescplural}[2][]{%
  \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsdescplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
}
```

`\Glsdescplural` behaves like `\glsdescplural` except that the first letter is converted to uppercase.

`\Glsdescplural`

```
\newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
\newcommand*{\@Glsdescplural}[2][]{%
  \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsdescplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
}
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
\newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
\newcommand*{\@GLSdescplural}[2][]{%
  \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSdescplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
}
```

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

`\glssymbol`

```
\newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
\newcommand*{\@glssymbol}[2][]{%
  \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glssymbol@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
}
```

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
\newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
\newcommand*{\@Glssymbol}[2][]{%
  \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glssymbol@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}%
}
```

\GLSsymbol behaves like \glssymbol except that the link text is converted to uppercase.

\GLSsymbol

```
\newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
\newcommand*{\@GLSsymbol}[2][]{%
  \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSsymbol@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
}
```

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

glssymbolplural

```
\newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

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

```
\newcommand*{\@glssymbolplural}[2][]{%
  \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glssymbolplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
}
```

\Glssymbolplural behaves like \glssymbolplural except that the first letter is converted to uppercase.

Glssymbolplural

```
\newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
\newcommand*{\@Glssymbolplural}[2][]{%
  \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glssymbolplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
}
```

\GLSsymbolplural behaves like \glssymbolplural except that the link text is converted to uppercase.

GLSsymbolplural

```
\newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
\newcommand*{\@GLSsymbolplural}[2][]{%
  \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSsymbolplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
}
```

\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

```
\newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
\newcommand*{\@glsuseri}[2][]{%
  \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsuseri@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
}
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
\newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
\newcommand*{\@Glsuseri}[2][]{%
  \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsuseri@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
}
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
\newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
\newcommand*{\@GLSuseri}[2][]{%
  \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSuseri@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
}
```

\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

```
\newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
\newcommand*{\@glsuserii}[2][]{%
  \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsuserii@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
}
```

`\Glsuserii` behaves like `\glsuserii` except that the first letter is converted to uppercase.

`\Glsuserii`

```
\newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
\newcommand*{\@Glsuserii}[2][]{%
  \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsuserii@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
}
```

`\GLSuserii` behaves like `\glsuserii` except that the link text is converted to uppercase.

`\GLSuserii`

```
\newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
\newcommand*{\@GLSuserii}[2][]{%
  \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSuserii@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
}
```

`\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`

```
\newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
\newcommand*{\@glsuseriii}[2][]{%
  \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsuseriii@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%
}
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

\Glsuseriii

```
\newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
\newcommand*{\@Glsuseriii}[2][]{%
  \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsuseriii@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
}
```

\GLSuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
\newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
\newcommand*{\@GLSuseriii}[2][]{%
  \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSuseriii@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
}
```

\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

```
\newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
\newcommand*{\@glsuseriv}[2][]{%
  \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsuseriv@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
}
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
\newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
\newcommand*{\@Glsuseriv}[2][]{%
  \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsuseriv@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
}
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
\newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
\newcommand*{\@GLSuseriv}[2][]{%
  \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSuseriv@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
}
```

\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

```
\newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
\newcommand*{\@glsuserv}[2][]{%
  \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsuserv@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
}
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
\newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
\newcommand*{\@Glsuserv}[2][]{%
\new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsuserv@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
}
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
\newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
\newcommand*{\@GLSuserv}[2][]{%
\new@ifnextchar[{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSuserv@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
}
```

`\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`

```
\newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
\newcommand*{\@glsuservi}[2][]{%
  \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glsuservi@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
}
```

`\Glsuservi` behaves like `\glsuservi` except that the first letter is converted to uppercase.

`\Glsuservi`

```
\newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
\newcommand*{\@Glsuservi}[2][]{%
  \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glsuservi@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
}
```

`\GLSuservi` behaves like `\glsuservi` except that the link text is converted to uppercase.

`\GLSuservi`

```
\newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
\newcommand*{\@GLSuservi}[2][]{%
  \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSuservi@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
}
```

Now deal with acronym related keys. First the short form:

`\acrshort`

```
\newrobustcmd*{\acrshort}{\@gls@hyp@opt\ns@acrshort}
```

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

```
\newcommand*{\ns@acrshort}[2][]{%
  \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsentryshort{#2}}#3%
    }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\Acrshort`

```
\newrobustcmd*{\Acrshort}{\@gls@hyp@opt\ns@Acrshort}
```

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

```
\newcommand*{\ns@Acrshort}[2][]{%
  \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsentryshort{#2}}#3%
    }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\ACRshort

```
\newrobustcmd*{\ACRshort}{\@gls@hyp@opt\ns@ACRshort}
```

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

```
\newcommand*{\ns@ACRshort}[2][]{%
  \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@ACRshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

    \def\glslabel{#2}%
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
    }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

Short plural:

\acrshortpl

```
\newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}
```

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

```
\newcommand*{\ns@acrshortpl}[2][]{%
  \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@acrshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsentryshortpl{#2}}#3%
    }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\Acrshortpl`

```
\newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

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

```
\newcommand*{\ns@Acrshortpl}[2][]{%
  \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Acrshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsentryshortpl{#2}}#3%
    }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\ACRshortpl`

```
\newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}
```

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

```
\newcommand*{\ns@ACRshortpl}[2][]{%
  \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@ACRshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
    }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\acrlong`

```
\newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}
```

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

```
\newcommand*{\ns@acrlong}[2][]{%
  \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
    \def\glscustomtext{%
      \glsentrylong{#2}#3%
    }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\Acrlong

```
\newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}
```

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

```
\newcommand*{\ns@Acrlong}[2][]{%
  \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
    \def\glscustomtext{%
      \Glsentrylong{#2}#3%
    }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\ACRlong

```
\newrobustcmd*{\ACRlong}{\@gls@hyp@opt\ns@ACRlong}
```

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

```
\newcommand*{\ns@ACRlong}[2][]{%
  \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@ACRlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
    }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%

  \glspostlinkhook
}
```

Short plural:

`\acrlongpl`

```
\newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\ns@acrlongpl}
```

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

```
\newcommand*{\ns@acrlongpl}[2][]{%
  \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@acrlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%

    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

    \def\glslabel{#2}%
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
    \def\glscustomtext{%
      \glsentrylongpl{#2}#3%
    }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%

  \glspostlinkhook
}
```

\Acrlongpl

```
\newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```

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

```
\newcommand*{\ns@Acrlongpl}[2][]{%
  \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Acrlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
    \def\glscustomtext{%
      \Glsentrylongpl{#2}#3%
    }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\ACRlongpl

```
\newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

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

```
\newcommand*{\ns@ACRlongpl}[2][]{%
  \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@ACRlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glslabel{#2}%
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
    }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%

  \glspostlinkhook
}
```

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⟩}
```

```
\newcommand*{\@gls@entry@field}[2]{%
  \csname glo@\glsdetoklabel{#1}@#2\endcsname
}
```

`glsletentryfield`

```
\glsletentryfield{⟨cs⟩}{⟨label⟩}{⟨field⟩}
```

```
\newcommand*{\glsletentryfield}[3]{%
  \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
}
```

`Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{⟨label⟩}{⟨field⟩}
```

```
\newcommand*{\@Gls@entry@field}[2]{%
  \glsdoifexistsordo{#1}%
  {%
    \letcs\@glo@text{glo@\glsdetoklabel{#1}@#2}%
    \ifdef\@glo@text
    {%
      \xmakefirstuc{\@glo@text}%
    }%
    {%
      ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
      entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
      label and the field name}%
    }%
  }%
  {%
    ??%
  }%
}
```

Get the entry name (as specified by the name key when the entry was defined). The argument is the label associated with the entry. Note that unless you used `name=false` in the sanitize package option you may get unexpected results if the name key contains any commands.

`\glsentryname`

```
\newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}
```

`\Glsentryname`

```
\newrobustcmd*{\Glsentryname}[1]{%
  \@Gls@entryname{#1}%
}
```

`\@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:

```
\newcommand*{\@Gls@entryname}[1]{%
  \@Gls@entry@field{#1}{name}%
}
```

`ls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```
\newcommand*{\@Gls@acrentryname}[1]{%
  \ifglshaslong{#1}%
  {%
    \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
    \expandafter\@gls@getbody\@glo@text{}\@nil
    \expandafter\ifx\@gls@body\glsentrylong\relax
      \expandafter\Glsentrylong\@gls@rest
    \else
      \expandafter\ifx\@gls@body\glsentryshort\relax
        \expandafter\Glsentryshort\@gls@rest
      \else
        \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.

```
          {%
             \let\glsentryshort\Glsentryshort
             \@glo@text
          }%
        \else
        \xmakefirstuc{\@glo@text}%
      \fi
    \fi
  \fi
 }%
 {%
```

Not an acronym

```
  \@Gls@entry@field{#1}{name}%
 }%
}
```

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`

```
\newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
\newrobustcmd*{\Glsentrydesc}[1]{%
  \@Gls@entry@field{#1}{desc}%
}
```

Plural form:

`entrydescplural`

```
\newcommand*{\glsentrydescplural}[1]{%
  \@gls@entry@field{#1}{descplural}%
}
```

`entrydescplural`

```
\newrobustcmd*{\Glsentrydescplural}[1]{%
  \@Gls@entry@field{#1}{descplural}%
}
```

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`

```
\newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
\newrobustcmd*{\Glsentrytext}[1]{%
  \@Gls@entry@field{#1}{text}%
}
```

Get the plural form:

\glsentryplural

```
\newcommand*{\glsentryplural}[1]{%
  \@gls@entry@field{#1}{plural}%
}
```

\Glsentryplural

```
\newrobustcmd*{\Glsentryplural}[1]{%
  \@Gls@entry@field{#1}{plural}%
}
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

\glsentrysymbol

```
\newcommand*{\glsentrysymbol}[1]{%
  \@gls@entry@field{#1}{symbol}%
}
```

\Glsentrysymbol

```
\newrobustcmd*{\Glsentrysymbol}[1]{%
  \@Gls@entry@field{#1}{symbol}%
}
```

Plural form:

trysymbolplural

```
\newcommand*{\glsentrysymbolplural}[1]{%
  \@gls@entry@field{#1}{symbolplural}%
}
```

trysymbolplural

```
\newrobustcmd*{\Glsentrysymbolplural}[1]{%
  \@Gls@entry@field{#1}{symbolplural}%
}
```

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

```
\newcommand*{\glsentryfirst}[1]{%
  \@gls@entry@field{#1}{first}%
}
```

\Glsentryfirst

```
\newrobustcmd*{\Glsentryfirst}[1]{%
  \@Gls@entry@field{#1}{first}%
}
```

Get the plural form (as specified by the firstplural key when the entry was defined).

ntryfirstplural

```
\newcommand*{\glsentryfirstplural}[1]{%
  \@gls@entry@field{#1}{firstpl}%
}
```

ntryfirstplural

```
\newrobustcmd*{\Glsentryfirstplural}[1]{%
  \@Gls@entry@field{#1}{firstpl}%
}
```

sentrytitlecase

```
\newrobustcmd*{\@glsentrytitlecase}[2]{%
  \glsfieldfetch{#1}{#2}{\@gls@value}%
  \xcapitalisewords{\@gls@value}%
}
\ifdef\texorpdfstring
{
  \newcommand*{\glsentrytitlecase}[2]{%
    \texorpdfstring
      {\@glsentrytitlecase{#1}{#2}}%
      {\@gls@entry@field{#1}{#2}}%
  }
}
{
  \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
}
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
\newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
\newcommand*{\glsentrysort}[1]{%
  \@gls@entry@field{#1}{sort}%
}
```

\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.

```
\newcommand*{\glsentryuseri}[1]{%
  \@gls@entry@field{#1}{useri}%
}
```

\Glsentryuseri

```
\newrobustcmd*{\Glsentryuseri}[1]{%
```

```
  \@Gls@entry@field{#1}{useri}%
}
```

\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.

```
\newcommand*{\glsentryuserii}[1]{%
  \@gls@entry@field{#1}{userii}%
}
```

\Glsentryuserii

```
\newrobustcmd*{\Glsentryuserii}[1]{%
  \@Gls@entry@field{#1}{userii}%
}
```

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.

```
\newcommand*{\glsentryuseriii}[1]{%
  \@gls@entry@field{#1}{useriii}%
}
```

Glsentryuseriii

```
\newrobustcmd*{\Glsentryuseriii}[1]{%
  \@Gls@entry@field{#1}{useriii}%
}
```

\glsentryuseriv Get the fourth user key (as specified by the user4 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuseriv}[1]{%
  \@gls@entry@field{#1}{useriv}%
}
```

\Glsentryuseriv

```
\newrobustcmd*{\Glsentryuseriv}[1]{%
  \@Gls@entry@field{#1}{useriv}%
}
```

\glsentryuserv Get the fifth user key (as specified by the user5 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuserv}[1]{%
  \@gls@entry@field{#1}{userv}%
}
```

\Glsentryuserv

```
\newrobustcmd*{\Glsentryuserv}[1]{%
  \@Gls@entry@field{#1}{userv}%
}
```

\glsentryuservi Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuservi}[1]{%
  \@gls@entry@field{#1}{uservi}%
}
```

\Glsentryuservi

```
\newrobustcmd*{\Glsentryuservi}[1]{%
  \@Gls@entry@field{#1}{uservi}%
}
```

\glsentryshort Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryshort}[1]{\@gls@entry@field{#1}{short}}
```

\Glsentryshort

```
\newrobustcmd*{\Glsentryshort}[1]{%
  \@Gls@entry@field{#1}{short}%
}
```

glsentryshortpl Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

Glsentryshortpl

```
\newrobustcmd*{\Glsentryshortpl}[1]{%
  \@Gls@entry@field{#1}{shortpl}%
}
```

\glsentrylong Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentrylong}[1]{\@gls@entry@field{#1}{long}}
```

\Glsentrylong

```
\newrobustcmd*{\Glsentrylong}[1]{%
  \@Gls@entry@field{#1}{long}%
}
```

\glsentrylongpl Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

\Glsentrylongpl

```
\newrobustcmd*{\Glsentrylongpl}[1]{%
  \@Gls@entry@field{#1}{longpl}%
}
```

Short cut macros to access full form:

\glsentryfull

```
\newcommand*{\glsentryfull}[1]{%
  \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
}
```

\Glsentryfull

```
\newrobustcmd*{\Glsentryfull}[1]{%
  \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
}
```

\glsentryfullpl

```
\newcommand*{\glsentryfullpl}[1]{%
  \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
}
```

\Glsentryfullpl

```
\newrobustcmd*{\Glsentryfullpl}[1]{%
  \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
}
```

entrynumberlist Displays the number list as is.

```
\newcommand*{\glsentrynumberlist}[1]{%
  \glsdoifexists{#1}%
  {%
    \@gls@entry@field{#1}{numberlist}%
  }%
}
```

splaynumberlist Formats the number list for the given entry label. Doesn't work with hyperref.

```
\@ifpackageloaded{hyperref} {%
  \newcommand*{\glsdisplaynumberlist}[1]{%
    \GlossariesWarning
    {%
      \string\glsdisplaynumberlist\space
      doesn't work with hyperref.^^JUsing
      \string\glsentrynumberlist\space instead%
    }%
    \glsentrynumberlist{#1}%
  }%
}%
{%
  \newcommand*{\glsdisplaynumberlist}[1]{%
    \glsdoifexists{#1}%
    {%
      \bgroup
```

```
         \edef\@glo@label{\glsdetoklabel{#1}}%
         \let\@org@glsnumberformat\glsnumberformat
         \def\glsnumberformat##1{##1}%
         \protected@edef\the@numberlist{%
           \csname glo@\@glo@label @numberlist\endcsname}%
         \def\@gls@numlist@sep{}%
         \def\@gls@numlist@nextsep{}%
         \def\@gls@numlist@lastsep{}%
         \def\@gls@thislist{}%
         \def\@gls@donext@def{}%
         \renewcommand\do[1]{%
           \protected@edef\@gls@thislist{%
             \@gls@thislist
             \noexpand\@gls@numlist@sep
             ##1%
           }%
           \let\@gls@numlist@sep\@gls@numlist@nextsep
           \def\@gls@numlist@nextsep{\glsnumlistsep}%
           \@gls@donext@def
           \def\@gls@donext@def{%
             \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
           }%
         }%
         \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
         \let\@gls@numlist@sep\@gls@numlist@lastsep
         \@gls@thislist
      \egroup
    }%
  }
}
```

\glsnumlistsep

```
\newcommand*{\glsnumlistsep}{, }
```

snumlistlastsep

```
\newcommand*{\glsnumlistlastsep}{ \& }
```

\glshyperlink Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\glslink` or `\glsadd` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```
\newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
 \def\@glo@label{#2}%
 \@glslink{\glolinkprefix\glsdetoklabel{#2}}{#1}}
```

1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
\define@key{glossadd}{counter}{\def\@gls@counter{#1}}
```

```
\define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
\define@key{glossadd}{types}{\def\@glo@type{#1}}
```

`\glsadd[⟨options⟩]{⟨label⟩}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that ⟨*options*⟩ only has two keys: counter and format (the types key will be ignored).

`\glsadd`

```
\newrobustcmd*{\glsadd}[2][]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```
  \@gls@adjustmode
  \glsdoifexists{#2}%
  {%
    \def\@glsnumberformat{glsnumberformat}%
    \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
    \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
    \@gls@saveentrycounter
```

Define sort key if necessary:

```
    \@gls@setsort{#2}%
```

This should use `\@@do@wrglossary` rather than `\@do@wrglossary` since the whole point of `\glsadd` is to add a line to the glossary.

```
    \@@do@wrglossary{#2}%
  }%
}
```

`@gls@adjustmode`

```
\newcommand*{\@gls@adjustmode}{}
\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`

```
\newrobustcmd*{\glsaddall}[1][]{%
  \edef\@glo@type{\@glo@types}%
  \setkeys{glossadd}{#1}%
  \forallglsentries[\@glo@type]{\@glo@entry}{%
    \glsadd[#1]{\@glo@entry}%
  }%
}
```

\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.

```
\newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
 \forallglsentries[#1]{\@glo@entry}%
 {%
    \ifglsused{\@glo@entry}{}{\glsadd[format=glsignore]{\@glo@entry}}%
 }%
}
```

\glsignore

```
\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, \@glsl@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.

```
\edef\glsopenbrace{\expandafter\@gobble\string\{}
```

\glsclosebrace Define \glsclosebrace to make it easier to write an opening brace to a file.

```
\edef\glsclosebrace{\expandafter\@gobble\string\}}
```

\glsbackslash Define \glsbackslash to make it easier to write a backslash to a file.

```
\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.

```
\edef\glsquote#1{\string"#1\string"}
```

\glspercentchar Define \glspercentchar to make it easier to write a percent character to a file.

```
\edef\glspercentchar{\expandafter\@gobble\string\%}
```

\glstildechar Define \glstildechar to make it easier to write a tilde character to a file.

```
\edef\glstildechar{\string~}
```

@glsfirstletter Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
\ifglsxindy
  \newcommand*{\@glsfirstletter}{A}
\fi
```

tterAfterDigits Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.

```
\newcommand*{\GlsSetXdyFirstLetterAfterDigits}{%
  \@ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
\ifglsxindy
  \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
    \renewcommand*{\@glsfirstletter}{#1}}
  \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
    \renewcommand*{\@glsfirstletter}{#1}%
    \@onelevel@sanitize\@glsfirstletter
  }
\else
  \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
    \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
  \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
    \@GlsSetXdyFirstLetterAfterDigits
  }
\fi
```

umbergrouporder Specifies the order of the number group.

```
\ifglsxindy
  \newcommand*{\@xdynumbergrouporder}{:before \string"\@glsfirstletter\string"}
\fi
```

umberGroupOrder Sets the relative location of the number group. The starred version sanitizes.

```
\newcommand*{\GlsSetXdyNumberGroupOrder}[1]{%
 \@ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
}
\ifglsxindy
  \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
    \renewcommand*{\@xdynumbergrouporder}{#1}%
  }
  \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
    \renewcommand*{\@xdynumbergrouporder}{#1}%
    \@onelevel@sanitize\@xdynumbergrouporder
  }
\else
  \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
    \glsnoxindywarning\GlsSetXdyNumberGroupOrder}
  \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
    \@GlsSetXdyNumberGroupOrder}
\fi
```

\@glsminrange Define the minimum number of successive location references to merge into a range.

```
\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.

```
\ifglsxindy
  \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
    \renewcommand*{\@glsminrange}{#1}}
\else
  \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
    \glsnoxindywarning\GlsSetXdyMinRangeLength}
\fi
```

\writeist

```
\ifglsxindy
```

Code to use if xindy is required.

```
  \def\writeist{%
```

Define write register if not already defined

```
    \ifundef{\glswrite}{\newwrite\glswrite}{}%
```

Update attributes list

```
    \@gls@addpredefinedattributes
```

Open the file.

```
    \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
    \write\glswrite{;; xindy style file created by the glossaries
        package}%
    \write\glswrite{;; for document '\jobname' on
       \the\year-\the\month-\the\day}%
```

Specify the required styles

```
    \write\glswrite{^^J; required styles^^J}
    \@for\@xdystyle:=\@xdyrequiredstyles\do{%
         \ifx\@xdystyle\@empty
         \else
```

```
        \protected@write\glswrite{}{(require
          \string"\@xdystyle.xdy\string")}%
      \fi
  }%
```

List the allowed attributes (possible values used by the format key)

```
  \write\glswrite{^^J%
     ; list of allowed attributes (number formats)^^J}%
  \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
  \write\glswrite{^^J; user defined alphabets^^J}%
  \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
  \write\glswrite{^^J; location class definitions^^J}%
```

As from version 3.0, locations are now specified as `{`⟨*Hprefix*⟩`}{`⟨*number*⟩`}`, so need to add all possible combinations of location types.

```
  \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case where ⟨*Hprefix*⟩ is empty:

```
    \protected@write\glswrite{}{(define-location-class
      \string"\@gls@classI\string"^^J\space\space\space
      (
        :sep "{}{"
        \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
        :sep "}"
      )
      ^^J\space\space\space
      :min-range-length \@glsminrange^^J%
      )
    }%
```

Nested iteration over all classes:

```
    {%
      \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
        \protected@write\glswrite{}{(define-location-class
          \string"\@gls@classII-\@gls@classI\string"
            ^^J\space\space\space
          (
            :sep "{"
            \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
            :sep "}{"
            \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
            :sep "}"
          )
          ^^J\space\space\space
          :min-range-length \@glsminrange^^J%
          )
        }%
      }%
    }%
  }%
```

User defined location classes (needs checking for new location format).

```
  \write\glswrite{^^J; user defined location classes}%
  \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.)

```
  \write\glswrite{^^J; define cross-reference class^^J}%
  \write\glswrite{(define-crossref-class \string"see\string"
      :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.)

```
  \write\glswrite{(markup-crossref-list
       :class \string"see\string"^^J\space\space\space
       :open \string"\string\glsseeformat\string"
       :close \string"{}\string")}%
```

Provide hook to write extra material here (used by glossaries-extra to define a seealso class).

```
  \@xdycrossrefhook
```

List the order to sort the classes.

```
  \write\glswrite{^^J; define the order of the location classes}%
  \write\glswrite{(define-location-class-order
       (\@xdylocationclassorder))}%
```

Specify what to write to the start and end of the glossary file.

```
  \write\glswrite{^^J; define the glossary markup^^J}%

  \write\glswrite{(markup-index^^J\space\space\space
      :open \string"\string
      \glossarysection[\string\glossarytoctitle]{\string
      \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)

```
  \@for\@this@ctr:=\@xdycounters\do{%
    {%
      \@for\@this@attr:=\@xdyattributelist\do{%
         \protected@write\glswrite{}{\string\providecommand*%
           \expandafter\string
           \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
           {%
              \string\setentrycounter
                [\expandafter\@gobble\string\#1]{\@this@ctr}%
              \expandafter\string
              \csname\@this@attr\endcsname
                {\expandafter\@gobble\string\#2}%
            }%
          }%
        }%
      }%
    }%
```

Add the end part of the open tag and the rest of the markup-index information:

```
    \write\glswrite{%
        \string\begin
        {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
        \space\space:close \string"\glspercentchar\glstildechar n\string
          \end{theglossary}\string\glossarypostamble
          \glstildechar n\string" ^^J\space\space\space
        :tree)}%
```

Specify what to put between letter groups

```
    \write\glswrite{(markup-letter-group-list
        :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Specify what to put between entries

```
    \write\glswrite{(markup-indexentry
        :open \string"\string\relax \string\glsresetentrylist
           \glstildechar n\string")}%
```

Specify how to format entries

```
    \write\glswrite{(markup-locclass-list :open
       \string"\glsopenbrace\string\glossaryentrynumbers
         \glsopenbrace\string\relax\space \string"^^J\space\space\space
       :sep \string", \string"
       :close \string"\glsclosebrace\glsclosebrace\string")}%
```

Specify how to separate location numbers

```
    \write\glswrite{(markup-locref-list
       :sep \string"\string\delimN\space\string")}%
```

Specify how to indicate location ranges

```
    \write\glswrite{(markup-range
       :sep \string"\string\delimR\space\string")}%
```

Specify 2-page and 3-page suffixes, if defined. First, the values must be sanitized to write them explicity.

```
    \@onelevel@sanitize\gls@suffixF
    \@onelevel@sanitize\gls@suffixFF
```

```
    \ifx\gls@suffixF\@empty
    \else
      \write\glswrite{(markup-range
        :close "\gls@suffixF" :length 1 :ignore-end)}%
    \fi
    \ifx\gls@suffixFF\@empty
    \else
      \write\glswrite{(markup-range
        :close "\gls@suffixFF" :length 2 :ignore-end)}%
    \fi
```

Specify how to format locations.

```
    \write\glswrite{^^J; define format to use for locations^^J}%
    \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
    \write\glswrite{^^J; define letter group list format^^J}%
    \write\glswrite{(markup-letter-group-list
       :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Define letter group headings.

```
    \write\glswrite{^^J; letter group headings^^J}%
    \write\glswrite{(markup-letter-group
        :open-head \string"\string\glsgroupheading
        \glsopenbrace\string"^^J\space\space\space
        :close-head \string"\glsclosebrace\string")}%
```

Define additional letter groups.

```
    \write\glswrite{^^J; additional letter groups^^J}%
    \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
    \write\glswrite{^^J; additional sort rules^^J}
    \write\glswrite{\@xdysortrules}%
```

Hook for any additional information:

```
    \@gls@writeisthook
```

Close the style file

```
    \closeout\glswrite
```

Suppress any further calls.

```
    \let\writeist\relax
  }
\else
```

Code to use if makeindex is required.

```
  \edef\@gls@actualchar{\string?}
  \edef\@gls@encapchar{\string|}
  \edef\@gls@levelchar{\string!}
  \edef\@gls@quotechar{\string"}%
  \let\GlsSetQuote\gls@nosetquote
  \def\writeist{\relax
   \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
   \openout\glswrite=\istfilename
    \write\glswrite{\glspercentchar\space makeindex style file
      created by the glossaries package}
    \write\glswrite{\glspercentchar\space for document
      '\jobname' on \the\year-\the\month-\the\day}
    \write\glswrite{actual '\@gls@actualchar'}
    \write\glswrite{encap '\@gls@encapchar'}
    \write\glswrite{level '\@gls@levelchar'}
    \write\glswrite{quote '\@gls@quotechar'}
    \write\glswrite{keyword \string"\string\\glossaryentry\string"}
    \write\glswrite{preamble \string"\string\\glossarysection[\string
      \\glossarytoctitle]{\string\\glossarytitle}\string
      \\glossarypreamble\string\n\string\\begin{theglossary}\string
      \\glossaryheader\string\n\string"}
    \write\glswrite{postamble \string"\string\%\string\n\string
      \\end{theglossary}\string\\glossarypostamble\string\n
      \string"}
    \write\glswrite{group_skip \string"\string\\glsgroupskip\string\n
      \string"}
    \write\glswrite{item_0 \string"\string\%\string\n\string"}
    \write\glswrite{item_1 \string"\string\%\string\n\string"}
    \write\glswrite{item_2 \string"\string\%\string\n\string"}
    \write\glswrite{item_01 \string"\string\%\string\n\string"}
    \write\glswrite{item_x1
      \string"\string\\relax \string\\glsresetentrylist\string\n
      \string"}
    \write\glswrite{item_12 \string"\string\%\string\n\string"}
    \write\glswrite{item_x2
      \string"\string\\relax \string\\glsresetentrylist\string\n
      \string"}

    \write\glswrite{delim_0 \string"\string\{\string
      \\glossaryentrynumbers\string\{\string\\relax \string"}
    \write\glswrite{delim_1 \string"\string\{\string
      \\glossaryentrynumbers\string\{\string\\relax \string"}
    \write\glswrite{delim_2 \string"\string\{\string
      \\glossaryentrynumbers\string\{\string\\relax \string"}
    \write\glswrite{delim_t \string"\string\}\string\}\string"}
    \write\glswrite{delim_n \string"\string\\delimN \string"}
    \write\glswrite{delim_r \string"\string\\delimR \string"}
    \write\glswrite{headings_flag 1}
    \write\glswrite{heading_prefix
       \string"\string\\glsgroupheading\string\{\string"}
    \write\glswrite{heading_suffix
       \string"\string\}\string\\relax
       \string\\glsresetentrylist \string"}
    \write\glswrite{symhead_positive \string"glssymbols\string"}
    \write\glswrite{numhead_positive \string"glsnumbers\string"}
    \write\glswrite{page_compositor \string"\glscompositor\string"}
    \@gls@escbsdq\gls@suffixF
    \@gls@escbsdq\gls@suffixFF
    \ifx\gls@suffixF\@empty
    \else
      \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
    \fi
```

```
    \ifx\gls@suffixFF\@empty
    \else
      \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
    \fi
```

Hook for any additional information:

```
    \@gls@writeisthook
```

Close the file and disable `\writeist`.

```
    \closeout\glswrite
    \let\writeist\relax
  }
\fi
```

SetWriteIstHook Allow user to append information to the style file.

```
\newcommand*{\GlsSetWriteIstHook}[1]{\renewcommand*{\@gls@writeisthook}{#1}}
\@onlypremakeg\GlsSetWriteIstHook
```

ls@writeisthook

```
\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.

```
\ifglsxindy
 \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
 \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
\else
 \newcommand*{\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}%
```

If German is in use, set the extra `makeindex` option so `makeglossaries` can pick it up.

```
    \@ifpackageloaded{tracklang}%
    {%
      \IfTrackedLanguage{german}%
      {%
        \def\@@gls@extramakeindexopts{-g}%
      }%
      {}%
    }%
    {}%
```

Need to redefine `\@gls@checkquote`

```
   \edef\@gls@docheckquotedef{%
     \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
       \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
       \noexpand\toks@={####1}%
       \noexpand\ifx\noexpand\null####2\noexpand\null
        \noexpand\ifx\noexpand\null####3\noexpand\null
         \noexpand\edef\noexpand\@gls@checkedmkidx{%
           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
         \noexpand\def\noexpand\@@gls@checkquote{\noexpand\relax}%
        \noexpand\else
         \noexpand\edef\noexpand\@gls@checkedmkidx{%
           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
           \noexpand\@gls@quotechar\noexpand\@gls@quotechar
           \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
         \noexpand\def\noexpand\@@gls@checkquote{%
           \noexpand\@gls@checkquote####3\noexpand\null}%
        \noexpand\fi
      \noexpand\else
        \noexpand\edef\noexpand\@gls@checkedmkidx{%
          \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
          \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
        \noexpand\ifx\noexpand\null####3\noexpand\null
          \noexpand\def\noexpand\@@gls@checkquote{%
            \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
        \noexpand\else
          \noexpand\def\noexpand\@@gls@checkquote{%
            \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
        \noexpand\fi
      \noexpand\fi
      \noexpand\@@gls@checkquote
    }%
  }%
  \@gls@docheckquotedef
  \edef\@gls@docheckquotedef{%
    \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
      \noexpand\def\noexpand\@gls@checkedmkidx{}%
      \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
        #1#1\noexpand\null
      \noexpand\expandafter\noexpand\@gls@updatechecked
        \noexpand\@gls@checkedmkidx{####1}%
      \noexpand\def\noexpand\@gls@checkedmkidx{}%
      \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
        \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
        \noexpand\null
      \noexpand\expandafter\noexpand\@gls@updatechecked
        \noexpand\@gls@checkedmkidx{####1}%
      \noexpand\def\noexpand\@gls@checkedmkidx{}%
      \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
        \noexpand\?\noexpand\?\noexpand\null
      \noexpand\expandafter\noexpand\@gls@updatechecked
        \noexpand\@gls@checkedmkidx{####1}%
      \noexpand\def\noexpand\@gls@checkedmkidx{}%
      \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
        \noexpand?\noexpand?\noexpand\null
      \noexpand\expandafter\noexpand\@gls@updatechecked
        \noexpand\@gls@checkedmkidx{####1}%
      \noexpand\def\noexpand\@gls@checkedmkidx{}%
      \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
        \noexpand|\noexpand|\noexpand\null
      \noexpand\expandafter\noexpand\@gls@updatechecked
        \noexpand\@gls@checkedmkidx{####1}%
      \noexpand\def\noexpand\@gls@checkedmkidx{}%
      \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
        \noexpand\|\noexpand\|\noexpand\null
      \noexpand\expandafter\noexpand\@gls@updatechecked
        \noexpand\@gls@checkedmkidx{####1}%
      \noexpand\def\noexpand\@gls@checkedmkidx{}%
      \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
        \noexpand!\noexpand!\noexpand\null
      \noexpand\expandafter\noexpand\@gls@updatechecked
        \noexpand\@gls@checkedmkidx{####1}%
    }%
  }%
  \@gls@docheckquotedef
  \edef\@gls@docheckquotedef{%
    \noexpand\def\noexpand\@gls@checkescquote####1%
      \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
      ####3\noexpand\null{%
      \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
      \noexpand\toks@={####1}%
      \noexpand\ifx\noexpand\null####2\noexpand\null
       \noexpand\ifx\noexpand\null####3\noexpand\null
        \noexpand\edef\noexpand\@gls@checkedmkidx{%
          \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
        \noexpand\def\noexpand\@@gls@checkescquote{\noexpand\relax}%
        \noexpand\else
        \noexpand\edef\noexpand\@gls@checkedmkidx{%
          \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
          \noexpand\@gls@quotechar\noexpand\string\expandonce{%
            \csname#1\endcsname}\noexpand\@gls@quotechar
          \noexpand\@gls@quotechar\noexpand\string\expandonce{%
            \csname#1\endcsname}\noexpand\@gls@quotechar}%
        \noexpand\def\noexpand\@@gls@checkescquote{%
          \noexpand\@gls@checkescquote####3\noexpand\null}%
       \noexpand\fi
       \noexpand\else
       \noexpand\edef\noexpand\@gls@checkedmkidx{%
         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
         \noexpand\@gls@quotechar\noexpand\string
           \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
       \noexpand\ifx\noexpand\null####3\noexpand\null
         \noexpand\def\noexpand\@@gls@checkescquote{%
          \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
          \expandonce{\csname#1\endcsname}\noexpand\null}%
       \noexpand\else
         \noexpand\def\noexpand\@@gls@checkescquote{%
           \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
```

```
            ####3\noexpand\null}%
        \noexpand\fi
      \noexpand\fi
    \noexpand\@@gls@checkescquote
   }%
  }%
  \@gls@docheckquotedef
}
\newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
  {\string\GlsSetQuote\space not permitted here}%
  {Move \string\GlsSetQuote\space earlier in the preamble, as
   soon as possible after glossaries.sty has been loaded}}
\fi
```

ramakeindexopts

```
\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

```
\newcommand{\noist}{%
```

Update attributes list

```
  \@gls@addpredefinedattributes
  \let\writeist\relax
}
```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the ⟨*out-ext*⟩ parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file).

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where TeX is trying to write to a non-existant file). The relevant glossary must be defined prior to using `\@makeglossary`.

\@makeglossary

```
\newcommand*{\@makeglossary}[1]{%
  \ifglossaryexists{#1}%
  {%
```

Only create a new write if savewrites=false otherwise create a token to collect the information.

```
    \ifglssavewrites
      \expandafter\newtoks\csname glo@#1@filetok\endcsname
    \else
      \expandafter\newwrite\csname glo@#1@file\endcsname
      \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
    \fi
    \@gls@renewglossary
    \writeist
  }%
  {%
    \PackageError{glossaries}%
    {Glossary type `#1' not defined}%
    {New glossaries must be defined before using \string\makeglossary}%
  }%
}
```

\@glsopenfile Open write file associated with the given glossary.

```
\newcommand*{\@glsopenfile}[2]{%
  \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
  \PackageInfo{glossaries}{Writing glossary file
     \jobname.\csname @glotype@#2@out\endcsname}%
}
```

\@closegls

```
\newcommand*{\@closegls}[1]{%
  \closeout\csname glo@#1@file\endcsname
}
```

\@gls@automake

```
\ifglsxindy
 \newcommand*{\@gls@automake}[1]{%
   \ifglossaryexists{#1}
   {%
     \@closegls{#1}%
     \ifdefstring{\glsorder}{letter}%
      {\def\@gls@order{-M ord/letorder }}%
      {\let\@gls@order\@empty}%
     \ifcsundef{@xdy@#1@language}%
      {\let\@gls@langmod\@xdy@main@language}%
      {\letcs\@gls@langmod{@xdy@#1@language}}%
     \edef\@gls@dothiswrite{\noexpand\write18{xindy
       -I xindy
       \@gls@order
       -L \@gls@langmod\space
       -M \gls@istfilebase\space
       -C \gls@codepage\space
       -t \jobname.\csuse{@glotype@#1@log}
       -o \jobname.\csuse{@glotype@#1@in}
       \jobname.\csuse{@glotype@#1@out}}%
     }%
     \@gls@dothiswrite
   }%
   {%
```

```
     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
   }%
 }
\else
 \newcommand*{\@gls@automake}[1]{%
   \ifglossaryexists{#1}
   {%
     \@closegls{#1}%
     \ifdefstring{\glsorder}{letter}%
      {\def\@gls@order{-l }}%
      {\let\@gls@order\@empty}%
     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
       -s \istfilename\space
       -t \jobname.\csuse{@glotype@#1@log}
       -o \jobname.\csuse{@glotype@#1@in}
       \jobname.\csuse{@glotype@#1@out}}%
     }%
     \@gls@dothiswrite
   }%
   {%
     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
   }%
 }
\fi
```

omakeglossaries Issue warning that \makeglossaries hasn't been used.

```
\newcommand*{\@warn@nomakeglossaries}{}
```

Only use this if warning if \printglossary has been used without \makeglossaries

```
\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

```
\newcommand*{\makeglossaries}{%
```

Define the write used for style file also used for all other output files if savewrites=true.

```
  \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.

```
  \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}}
  \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}}
```

If \@@gls@extramakeindexopts has been defined, write it:

```
  \ifundef\@@gls@extramakeindexopts
  {}%
  {%
    \protected@write\@auxout{}{\string\providecommand
```

```
        \string\@gls@extramakeindexopts[1]{}}
    \protected@write\@auxout{}{\string\@gls@extramakeindexopts
      {\@@gls@extramakeindexopts}}%
  }%
```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```
  \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
  \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
```

Iterate through each glossary type and activate it.

```
  \@for\@glo@type:=\@glo@types\do{%
    \ifthenelse{\equal{\@glo@type}{}}{}{%
    \@makeglossary{\@glo@type}}%
  }%
```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```
  \renewcommand*\newglossary[4][]{%
  \PackageError{glossaries}{New glossaries
  must be created before \string\makeglossaries}{You need
  to move \string\makeglossaries\space after all your
  \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
  \let\@makeglossary\relax
  \let\makeglossary\relax
  \let\makeglossaries\relax
```

Disable all commands that have no effect after `\makeglossaries`

```
  \@disable@onlypremakeg
```

Allow see key:

```
  \let\gls@checkseeallowed\relax
```

Suppress warning about no `\makeglossaries`

```
  \let\warn@nomakeglossaries\relax
```

Activate warning about missing `\printglossary`

```
  \def\warn@noprintglossary{%
    \ifdefstring{\@glo@types}{,}%
    {%
      \GlossariesWarningNoLine{No glossaries have been defined}%
    }%
    {%
      \GlossariesWarningNoLine{No \string\printglossary\space
        or \string\printglossaries\space
        found. ^^J(Remove \string\makeglossaries\space if you
        don't want any glossaries.) ^^JThis document will not
        have a glossary}%
    }%
  }%
```

Declare list parser for `\glsdisplaynumberlist`

```
  \ifglssavenumberlist
  \edef\@gls@dodeflistparser{\noexpand\DeclareListParser
    {\noexpand\glsnumlistparser}{\delimN}}%
  \@gls@dodeflistparser
\fi
```

Prevent user from also using `\makenoidxglossaries`

```
\let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
\renewcommand*{\@printgloss@setsort}{%
  \let\@glo@assign@sortkey\@glo@no@assign@sortkey
}%
```

Check the automake setting:

```
\ifglsautomake
  \renewcommand*{\@gls@doautomake}{%
    \@for\@gls@type:=\@glo@types\do{%
      \ifdefempty{\@gls@type}{}%
      {\@gls@automake{\@gls@type}}%
    }%
  }%
\fi
```

Check the sort setting:

```
\@glo@check@sortallowed\makeglossaries
}
```

Must occur in the preamble:

```
\@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`

```
\let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
\AtEndDocument{%
  \warn@nomakeglossaries
  \warn@noprintglossary
}
```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
\newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
\renewcommand{\@gls@noref@warn}[1]{%
  \GlossariesWarning{Empty glossary for
  \string\printnoidxglossary[type={##1}].
  Rerun may be required (or you may have forgotten to use
  commands like \string\gls)}%
}%
```

Don't escape makeindex/xindy characters:

```
\let\@gls@checkmkidxchars\@gobble
```

Don't escape locations:

```
\glsesclocationsfalse
```

Write glossary information to aux instead of glossary files

```
\let\@@do@@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
\let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
\renewcommand{\@do@seeglossary}[2]{%
  \edef\@gls@label{\glsdetoklabel{##1}}%
  \protected@write\@auxout{}{%
    \string\@gls@reference
      {\csname glo@\@gls@label @type\endcsname}%
      {\@gls@label}%
      {%
        \string\glsseeformat##2{}%
      }%
  }%
}%
```

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.

```
\AtBeginDocument
{%
  \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
}%
```

Change warning about no glossaries

```
\def\warn@noprintglossary{%
  \GlossariesWarningNoLine{No \string\printnoidxglossary\space
    or \string\printnoidxglossaries ^^J
    found. (Remove \string\makenoidxglossaries\space if you
    don't want any glossaries.)^^JThis document will not have a glossary}%
}%
```

Suppress warning about no `\makeglossaries`

```
\let\warn@nomakeglossaries\relax
```

Prevent user from also using `\makeglossaries`

```
\let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
\renewcommand*{\@printgloss@setsort}{%
  \let\@glo@assign@sortkey\@@glo@assign@sortkey
```

Initialise default sort order:

```
  \def\@glo@sorttype{\@glo@default@sorttype}%
}%
```

All entries must be defined in the preamble:

```
\renewcommand*\new@glossaryentry[2]{%
  \PackageError{glossaries}{Glossary entries must be
   defined in the preamble^^Jwhen you use
   \string\makenoidxglossaries}%
  {Either move your definitions to the preamble or use
   \string\makeglossaries}%
}%
```

Redefine `\glsentrynumberlist`

```
\renewcommand*{\glsentrynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \glsnoidxloclist{\@gls@loclist}%
  }%
  {%
    ??\glsdoifexists{##1}%
    {%
      \GlossariesWarning{Missing location list for ‘##1’. Either
        a rerun is required or you haven’t referenced the entry}%
    }%
  }%
}%
```

Redefine `\glsdisplaynumberlist`

```
\renewcommand*{\glsdisplaynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \def\@gls@noidxloclist@sep{%
      \def\@gls@noidxloclist@sep{%
        \def\@gls@noidxloclist@sep{%
          \glsnumlistsep
        }%
        \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
      }%
    }%
    \def\@gls@noidxloclist@finalsep{}%
    \def\@gls@noidxloclist@prev{}%
```

```
      \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
      \@gls@noidxloclist@finalsep
      \@gls@noidxloclist@prev
    }%
    {%
      ??\glsdoifexists{##1}%
      {%
        \GlossariesWarning{Missing location list for '##1'. Either
          a rerun is required or you haven't referenced the entry}%
      }%
    }%
  }%
```

Provide a generic way of iterating through the number list:

```
  \renewcommand*{\glsnumberlistloop}[3]{%
    \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
    \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
    \let\@gls@org@glsseeformat\glsseeformat
    \let\glsnoidxdisplayloc##2\relax
    \let\glsseeformat##3\relax
    \ifdef\@gls@loclist
    {%
      \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
    }%
    {%
      ??\glsdoifexists{##1}%
      {%
        \GlossariesWarning{Missing location list for '##1'. Either
          a rerun is required or you haven't referenced the entry}%
      }%
    }%
    \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
    \let\glsseeformat\@gls@org@glsseeformat
  }%
```

Modify sanitize sort function

```
  \let\@@gls@sanitizesort\@gls@noidx@sanitizesort
  \let\@@gls@nosanitizesort\@@gls@noidx@nosanitizesort
  \@gls@noidx@setsanitizesort
```

Check sort option allowed.

```
  \@glo@check@sortallowed\makenoidxglossaries
}
```

Preamble-only command:

```
\@onlypreamble{\makenoidxglossaries}
```

lsnumberlistloop

\glsnumberlistloop{⟨*label*⟩}{⟨*handler*⟩}

```
\newcommand*{\glsnumberlistloop}[2]{%
  \PackageError{glossaries}{\string\glsnumberlistloop\space
   only works with \string\makenoidxglossaries}{}%
}
```

listloophandler Handler macro for \glsnumberlistloop. (The argument should be in the form \glsnoidxdisplayloc {⟨*prefix*⟩}{⟨*counter*⟩}{⟨*format*⟩}{⟨*n*⟩})

```
\newcommand*{\glsnoidxnumberlistloophandler}[1]{%
  #1%
}
```

@makeglossaries Can't use both \makeglossaries and \makenoidxglossaries

```
\newcommand*{\@no@makeglossaries}{%
  \PackageError{glossaries}{You can't use both
  \string\makeglossaries\space and \string\makenoidxglossaries}%
  {Either use one or other (or none) of those commands but not both
  together.}%
}
```

@gls@noref@warn Warning when no instances of \@gls@reference found.

```
\newcommand{\@gls@noref@warn}[1]{%
  \GlossariesWarning{\string\makenoidxglossaries\space
   is required to make \string\printnoidxglossary[type={#1}] work}%
}
```

1.14 Writing information to associated files

s@noidxglossary Write the glossary information to the aux file (for the 'noidx' method):

```
\newcommand*{\gls@noidxglossary}{%
  \protected@write\@auxout{}{%
    \string\@gls@reference
      {\csname glo@\@gls@label @type\endcsname}%
      {\@gls@label}%
      {\string\glsnoidxdisplayloc
        {\@glo@counterprefix}%
        {\@gls@counter}%
        {\@glsnumberformat}%
        {\@glslocref}%
      }%
  }%
}
```

\istfile Deprecated.

```
\providecommand\istfile{\glswrite}
```

At the end of the document, the files should be created if savewrites=true.

```
\AtEndDocument{%
```

```
  \glswritefiles
}
```

\@glswritefiles Only write the files if savewrites=true.

```
\newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries.

```
  \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
     \ifcsundef{glo@\@glo@type @filetok}%
     {%
        \def\gls@tmp{}%
     }%
     {%
        \edef\gls@tmp{\expandafter\the
           \csname glo@\@glo@type @filetok\endcsname}%
     }%
     \ifx\gls@tmp\@empty
        \ifx\@glo@type\glsdefaulttype
          \GlossariesWarningNoLine{Glossary '\@glo@type' has no
             entries.^^JRemember to use package option 'nomain' if
you
             don't want to^^Juse the main glossary}%
        \else
          \GlossariesWarningNoLine{Glossary '\@glo@type' has no
             entries}%
        \fi
     \else
        \@glsopenfile{\glswrite}{\@glo@type}%
        \immediate\write\glswrite{%
           \expandafter\the
             \csname glo@\@glo@type @filetok\endcsname}%
        \immediate\closeout\glswrite
     \fi
  }%
}
```

As from v4.10, the \glossary command isn't 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 was never intended as a user level command (and wasn't documented in the user manual), 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

```
\if@gls@docloaded
\else
  \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
\fi
```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
\newcommand*{\gls@glossary}[1]{%
  \@gls@glossary{#1}%
}
```

`\@gls@glossary` `\@gls@glossary{⟨type⟩}{⟨indexing info⟩}`

(In v4.10, `\@glossary` was redefined to `\@gls@glossary` to avoid conflict with other packages.) Initially define internal `\@gls@glossary` to ignore its argument. Indexing will be enabled when `\@gls@glossary` is redefined by `\@makeglossary`.

This command was originally defined to do `\@index{`⟨*indexing info*⟩`}` so that it behaved much like `\index`. The definition was then changed to use `\index` as memoir changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.)

However, if normal indexing is enabled (for example with `\makeindex`) but no glossary lists are required (so `\@makeglossary` isn't used), then `\index` will cause a problem here. The `\@index` trick allows for special characters within ⟨*indexing info*⟩ (so you can do, for example, `\index{%@\%}`), and the original design of `\@glossary` here was actually a legacy from the old glossary package. With the glossaries package, the indexing information supplied in the second argument is more constrained and just consists of the sort value (given by the sort key), the actual value (given by `\glossentry{`⟨*label*⟩`}` or `\subglossentry{`⟨*level*⟩`}{`⟨*label*⟩`}`), and the format. This means that there's no need to worry about special characters appearing in the second argument as they can't be in the label or sort value. (If they are in the sort value then the category code would've needed to be changed when the entry was defined or `\glspercentchar` would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```
\newcommand*{\@gls@glossary}[2]{%
  \if@gls@debug
    \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
  \fi
}
```

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`

```
\newcommand{\@gls@renewglossary}{%
  \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
  \let\@gls@renewglossary\@empty
}
```

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`

```
\newcommand*{\gls@wrglossary}[2]{%
  \ifglssavewrites
    \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
    \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
       \expandafter{\@gls@tmp^^J}%
  \else
    \ifcsdef{glo@#1@file}%
    {%
      \expandafter\protected@write\csname glo@#1@file\endcsname{%
        \gls@disablepagerefexpansion}{#2}%
    }%
    {%
       \ifignoredglossary{#1}{}%
       {%
          \GlossariesWarning{No file defined for glossary '#1'}%
       }%
    }%
  \fi
  \endgroup\@esphack
}
```

`\@do@wrglossary`

```
\newcommand*{\@do@wrglossary}[1]{%
  \glswriteentry{#1}{\@@do@wrglossary{#1}}%
}
```

`\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.

```
\newcommand*{\glswriteentry}[2]{%
  \ifglsindexonlyfirst
    \ifglsused{#1}{}{#2}%
  \else
    #2%
  \fi
}
```

`tected@pagefmts` List of page formats to be protected against expansion.

```
\newcommand{\gls@protected@pagefmts}{\gls@numberpage,\gls@alphpage,%
 \gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage}
```

`agerefexpansion`

```
\newcommand*{\gls@disablepagerefexpansion}{%
  \@for\@gls@this:=\gls@protected@pagefmts\do
  {%
    \expandafter\let\@gls@this\relax
  }%
}
```

\gls@alphpage

```
\newcommand*{\gls@alphpage}{\@alph\c@page}
```

\gls@Alphpage

```
\newcommand*{\gls@Alphpage}{\@Alph\c@page}
```

\gls@numberpage

```
\newcommand*{\gls@numberpage}{\number\c@page}
```

\gls@arabicpage

```
\newcommand*{\gls@arabicpage}{\@arabic\c@page}
```

\gls@romanpage

```
\newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

\gls@Romanpage

```
\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).

```
\newcommand*{\glsaddprotectedpagefmt}[1]{%
  \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
  \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
  \eappto\@wrglossarynumberhook{%
    \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
      \expandonce{\csname#1\endcsname}%
    \noexpand\def\expandonce{\csname#1\endcsname}{%
      \noexpand\@wrglossary@pageformat
         \expandonce{\csname gls#1page\endcsname}%
         \expandonce{\csname org@gls#1\endcsname}%
    }%
  }%
}
```

ssarynumberhook Hook used by \@@do@wrglossary

```
\newcommand*\@wrglossarynumberhook{}
```

sary@pageformat

```
\newcommand{\@wrglossary@pageformat}[3]{%
  \ifx#3\c@page #1\else #2#3\fi
}
```

@@do@wrglossary Write the glossary entry in the appropriate format.

```
\newcommand*{\@@do@wrglossary}[1]{%
  \ifglsesclocations
    \@@do@esc@wrglossary{#1}%
  \else
    \@@do@noesc@wrglossary{#1}%
  \fi
}
```

oesc@wrglossary Write the glossary entry in the appropriate format. The locations don't need to be pre-processed before writing the information to the glossary file, but the prefix still needs to be found.

```
\newcommand*{\@@do@noesc@wrglossary}[1]{%
```

Don't fully expand yet.

```
  \expandafter\def\expandafter\@glslocref\expandafter{\theglsentrycounter}%
  \expandafter\def\expandafter\@glsHlocref\expandafter{\theHglsentrycounter}%
```

Find the prefix if `\@glsHlocref` and `\@glslocref` aren't the same.

```
  \ifx\@glsHlocref\@glslocref
    \def\@glo@counterprefix{}%
  \else
```

The value of the counter isn't important here as it's the prefix that's of interest. (`\c@page` will have the same value in both `\theglsentrycounter` and `\theHglsentrycounter` at this point, even if it hasn't been updated yet. The page number is not expected to occur in the prefix.)

```
    \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
      {\@glslocref}{\@glsHlocref}%
    }%
    \@do@gls@getcounterprefix
  \fi
```

De-tok label if required.

```
  \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
  \@@do@@wrglossary
}
```

owprimitivemods Conditional to determine whether or not `\@@do@esc@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
\newif\ifglswrallowprimitivemods
\glswrallowprimitivemodstrue
```

`@esc@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. This is far more complicated with `xindy` than with other indexing methods. There are two necessary but conflicting requirements with `xindy`:

1. all backslashes in the location must be escaped;
2. `\c@page` can't be prematurely expanded.

(With `makeindex` there's the remote possibility that the page compositor is a `makeindex` special character, so that would also need to be escaped.)

For example, suppose `\thepage` is defined as

```
\renewcommand{\thepage}{\tally{page}}
\newcommand{\tally}[1]{\tallynum{\expandafter\the\csname c@#1\endcsname}}
```

where `\tallynum` is a robust command that takes a number as its argument. With all indexing methods other than `xindy`, a deferred write with `\thepage` as the location will expand to `\tallynum{`⟨*n*⟩`}` where ⟨*n*⟩ is the page number. Since the write is deferred, the page number is correct. (`makeindex` won't accept this location format, but `\makenoidxglossaries` and `bib2gls` are quite happy with it.) Unfortunately, this fails with `xindy` because `xindy` interprets this location as `tallynum{`⟨*n*⟩`}` because `\t` represents a the character "t". The location must be written as `\\tallynum{`⟨*n*⟩`}`.

This means that the location `\tally{page}` must be expanded and then the backslashes must be doubled. Unfortunately `\c@page` mustn't be expanded until the deferred write is performed, so the location actually needs to be expanded to `\tallynum{\the\c@page}` but the backslashes in `\the\c@page` mustn't be escaped. All other backslashes must be escaped. (In this case, only the backslash in `\tallynum` but the location format may include other control sequences.) The code below works on the assumption that commands like `\tally` are defined in the form

```
\newcommand{\tally}[1]{\tallynum{\expandafter\the\csname c@#1\endcsname}}
```

(note the use of `\expandafter` and `\name`) or in the form

```
\newcommand{\tally}[1]{\tallynum{\arabic{#1}}}
```

In the second case, `\arabic` is one of the known commands that's temporarily adjusted to prevent `\c@page` from being prematurely expanded. In the first case, `\the` is temporarily modified (unless `\glswrallowprimitivemodsfalse`) to check if it's followed by `\c@page`. The `\expandafter` ensures that it is. If `\tally` is defined in another way that hides `\c@page` for example using `\the\value{#1}` then the process fails.

With `makeindex`, `\tallynum` needs to expand to just the decimal number while writing the location to the glossary file, otherwise `makeindex` will reject it. This can be done by defining `\glstallypage` so that `\tally` can locally be set to `\arabic` while expansion is occurring. Again, `\c@page` must be protected from expansion until the deferred write occurs.

The expansion before the write occurs also allows the hyper prefix to be determined where `\theH`⟨*counter*⟩ is defined in the form ⟨*prefix*⟩`.\the`⟨*counter*⟩. It's possible (although again unlikely) that a `makeindex` character might occur in the prefix, which therefore needs escaping. The prefix is passed as the optional argument of `\setentrycounter` which is needed by commands like `\glshypernumber` to create a hyperlink for a given counter (like `\hyperpage` but for an arbitrary counter).

```
\newcommand*{\@@do@esc@wrglossary}[1]{% please read documented code!
  \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions (scoped):

```
    \let\gls@orgthe\the
    \let\gls@orgnumber\number
    \let\gls@orgarabic\@arabic
    \let\gls@orgromannumeral\romannumeral
    \let\gls@orgalph\@alph
    \let\gls@orgAlph\@Alph
    \let\gls@orgRoman\@Roman
```

Redefine:

```
    \ifglswrallowprimitivemods
```

The redefinition of `\the` to use `\expandafter` solves the problem of `\the\csname c@`⟨*counter*⟩`\endcsname` but is only a partial solution to the problem of `\the\value`. With `\value`, `\c@page` is too deeply hidden and will be expanded too soon, but at least there won't be an error.

```
      \def\gls@the##1{%
        \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
      \def\the{\expandafter\gls@the}%
      \def\gls@number##1{%
        \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%
      \def\number{\expandafter\gls@number}%
    \fi
    \def\@arabic##1{%
      \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
    \def\romannumeral##1{%
      \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
    \def\@Roman##1{%
      \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
    \def\@alph##1{%
      \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
    \def\@Alph##1{%
      \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%
```

Add hook to allow for other number formats:

```
  \@wrglossarynumberhook
```

Prevent expansion:

```
    \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
    \protected@xdef\@glslocref{\theglsentrycounter}%
  \endgroup
```

Escape any special characters. It's possible that with makeindex the separator might be a makeindex special character. Although not likely, it still needs to be taken into account.

```
  \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
  \expandafter\ifx\theHglsentrycounter\theglsentrycounter\relax
    \def\@glo@counterprefix{}%
  \else
    \protected@edef\@glsHlocref{\theHglsentrycounter}%
    \@gls@checkmkidxchars\@glsHlocref
    \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
      {\@glslocref}{\@glsHlocref}%
    }%
    \@do@gls@getcounterprefix
  \fi
```

De-tok label if required

```
  \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
  \@@do@@wrglossary
}
```

@do@@wrglossary

```
\newcommand*{\@@do@@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
  \ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```
    \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
    \def\@glo@range{}%
    \expandafter\if\@glo@prefix(\relax
      \def\@glo@range{:open-range}%
    \else
      \expandafter\if\@glo@prefix)\relax
        \def\@glo@range{:close-range}%
      \fi
    \fi
```

Write to the glossary file using xindy syntax.

```
    \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
    (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

      :locref \string"{\@glo@counterprefix}{\@glslocref}\string" %
      :attr \string"\@gls@counter\@glo@suffix\string"
      \@glo@range
    )
    }%
  \else
```

Convert the format information into the format required for makeindex

```
    \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
      {\@glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```
    \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
    \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
      \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
  \fi
}
```

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.)

```
\newcommand*\@gls@getcounterprefix[2]{%
  \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
  \ifx\@gls@thisloc\@gls@thisHloc
    \def\@glo@counterprefix{}%
  \else
    \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
      \def\@glo@tmp{##2}%
      \ifx\@glo@tmp\@empty
        \def\@glo@counterprefix{}%
      \else
        \def\@glo@counterprefix{##1}%
      \fi
    }%
    \@gls@get@counterprefix#2.#1\end@getprefix
```

Warn if no prefix can be formed.

```
    \ifx\@glo@counterprefix\@empty
      \GlossariesWarning{Hyper target `#2' can't be formed by
       prefixing^^Jlocation `#1'. You need to modify the
       definition of \string\theH\@gls@counter^^Jotherwise you
       will get the warning: "`name{\@gls@counter.#1}' has been^^J
       referenced but does not exist"}%
    \fi
  \fi
}
```

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 `[`⟨*tag*⟩`]{`⟨*list*⟩`}`, where ⟨*tag*⟩ is a tag such as "see" and ⟨*list*⟩ is a list of labels.

```
\newcommand{\@do@seeglossary}[2]{%
\def\@gls@xref{#2}%
\@onelevel@sanitize\@gls@xref
\@gls@checkmkidxchars\@gls@xref
\ifglsxindy
  \gls@glossary{\csname glo@#1@type\endcsname}{%
    (indexentry
      :tkey (\csname glo@#1@index\endcsname)
      :xref (\string"\@gls@xref\string")
      :attr \string"see\string"
    )
  }%
\else
  \gls@glossary{\csname glo@#1@type\endcsname}{%
  \string\glossaryentry{\csname glo@#1@index\endcsname
  \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
\fi
}
```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```
\def\@gls@fixbraces#1#2#3\@nil{%
  \ifx#2[\relax
   \@@gls@fixbraces#1#2#3\@end@fixbraces
  \else
    \def#1{{#2#3}}%
  \fi
}
```

`@@gls@fixbraces`

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

`\glssee` `\glssee{`⟨*label*⟩`}{`⟨*cross-ref list*⟩`}`

```
\DeclareRobustCommand*{\glssee}[3][\seename]{%
  \@do@seeglossary{#2}{[#1]{#3}}}
\newcommand*{\@glssee}[3][\seename]{%
  \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.

```
\DeclareRobustCommand*{\glsseeformat}[3][\seename]{%
  \emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{`⟨*list*⟩`}` formats list of entry labels.

```
\DeclareRobustCommand*{\glsseelist}[1]{%
```

If there is only one item in the list, set the last separator to do nothing.

```
  \let\@gls@dolast\relax
```

Don't display separator on the first iteration of the loop

```
  \let\@gls@donext\relax
```

Iterate through the labels

```
  \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
    \ifx\@xfor@nextelement\@nnil
      \@gls@dolast
    \else
      \@gls@donext
    \fi
```

Display the entry for this label. (Expanding label as it's a temporary control sequence that's used elsewhere.)

```
    \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
    \let\@gls@dolast\glsseelastsep
    \let\@gls@donext\glsseesep
  }%
}
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
\newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
\newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{`⟨*label*⟩`}` formats individual entry in a cross-referencing list.

```
\DeclareRobustCommand*{\glsseeitem}[1]{\glshyperlink[\glsseeitemformat{#1}]{#1}}
```

`lsseeitemformat` As from v3.0, default is to use `\glsentrytext` instead of `\glsentryname`. (To avoid problems with the name key being sanitized, although this is no longer a problem now.)

```
\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.

```
\newcommand*{\gls@save@numberlist}[1]{%
  \ifglssavenumberlist
    \toks@{#1}%
    \edef\@do@writeaux@info{%
```

```
        \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
    }%
    \@onelevel@sanitize\@do@writeaux@info
    \protected@write\@auxout{}{\@do@writeaux@info}%
  \fi
}
```

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.)

```
\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.

```
\ifcsundef{printglossary}{}%
{%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
  \@gls@warnonglossdefined
  \undef\printglossary
}
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
\newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
  \@printglossary{#1}{\@print@glossary}%
}
```

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

```
\newcommand*{\printglossaries}{%
  \forallglossaries{\@@glo@type}{\printglossary[type=\@@glo@type]}%
}
```

ntnoidxglossary 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.

```
\newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
  \@printglossary{#1}{\@print@noidx@glossary}%
}
```

noidxglossaries Analogous to \printglossaries

```
\newcommand*{\printnoidxglossaries}{%
  \forallglossaries{\@@glo@type}{\printnoidxglossary[type=\@@glo@type]}%
}
```

ntgloss@setsort Initialise to do nothing.

```
\newcommand*{\@printgloss@setsort}{}
```

preglossaryhook

```
\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.

```
\newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
  \def\@glo@type{\glsdefaulttype}%
  \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%

  \def\glossarytoctitle{\glossarytitle}%
  \let\org@glossarytitle\glossarytitle

  \def\@glossarystyle{%
    \ifx\@glossary@default@style\relax
      \GlossariesWarning{No default glossary style provided \MessageBreak
        for the glossary ‘\@glo@type’. \MessageBreak
        Using deprecated fallback. \MessageBreak
        To fix this set the style with \MessageBreak
        \string\setglossarystyle\space or use the \MessageBreak
        style key=value option}%
    \fi
  }%
  \def\gls@dotoctitle{\glssettoctitle{\@glo@type}}%
```

Store current value of \glossaryentrynumbers. (This may be changed via the optional argument)

```
  \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
  \bgroup
```

Activate or deactivate sort key:

```
  \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
  \setkeys{printgloss}{#1}%
```

Does the glossary exist?

```
  \ifglossaryexists{\@glo@type}%
  {%
```

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)

```
    \ifx\glossarytitle\org@glossarytitle
    \else
      \expandafter\let\csname @glotype@\@glo@type @title\endcsname
                    \glossarytitle
    \fi
```

Allow a high-level user command to indicate the current glossary

```
    \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
    \let\org@glossaryentrynumbers\glossaryentrynumbers
    \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
    \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
    \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
    \gls@dotoctitle
```

Set the glossary style

```
    \@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):

```
    \let\gls@org@glossaryentryfield\glossentry
    \let\gls@org@glossarysubentryfield\subglossentry
    \renewcommand{\glossentry}[1]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
      \gls@org@glossaryentryfield{##1}%
    }%
    \renewcommand{\subglossentry}[2]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
      \gls@org@glossarysubentryfield{##1}{##2}%
    }%

    \@gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
    #2%
  }%
  {\GlossariesWarning{Glossary '\@glo@type' doesn't exist}}%
```

End the current scope

```
\egroup
```

Reset `\glossaryentrynumbers`

```
\global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no \printglossary

```
  \global\let\warn@noprintglossary\relax
}
```

@print@glossary Internal workings of \printglossary dealing with reading the external file.

```
\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.)

```
  \makeatletter
```

Input the glossary file, if it exists.

```
  \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
  \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
  {}%
  {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
  \ifglsxindy
    \ifcsundef{@xdy@\@glo@type @language}%
    {%
      \edef\@do@auxoutstuff{%
        \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.

```
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
        }%
      }%
    }%
    {%
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
              @language\endcsname}}%
        }%
      }%
    }%
    \@do@auxoutstuff
    \edef\@do@auxoutstuff{%
      \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.

```
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\providecommand\string\@gls@codepage[2]{}}%
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
      }%
    }%
    \@do@auxoutstuff
  \fi
```

Activate warning if `\makeglossaries` hasn't been used.

```
  \renewcommand*{\@warn@nomakeglossaries}{%
    \GlossariesWarningNoLine{\string\makeglossaries\space
    hasn't been used,^^Jthe glossaries will not be updated}%
  }%
}
```

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

```
\newcommand*{\@glo@sortentries}[2]{%
  \glosortentrieswarning
  \def\@glo@sortinglist{}%
  \def\@glo@sortinghandler{#1}%
  \edef\@glo@type{#2}%
  \forlistcsloop{\@glo@do@sortentries}{@glsref@#2}%
  \csdef{@glsref@#2}{}%
  \@for\@this@label:=\@glo@sortinglist\do{%
```

Has this entry already been added?

```
    \xifinlistcs{\@this@label}{@glsref@#2}%
    {}%
    {%
      \listcsxadd{@glsref@#2}{\@this@label}%
    }%
    \ifcsdef{@glo@sortingchildren@\@this@label}%
    {%
      \@glo@addchildren{#2}{\@this@label}%
    }%
    {}%
```

```
  }%
}
```

@glo@addchildren

\@glo@addchildren{⟨*type*⟩}{⟨*parent*⟩}

```
\newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```
  \bgroup
     \letcs{\@glo@childlist}{@glo@sortingchildren@#2}%
     \@for\@this@childlabel:=\@glo@childlist\do
     {%
```

Check this label hasn't already been added.

```
       \xifinlistcs{\@this@childlabel}{@glsref@#1}%
       {}%
       {%
         \listcsxadd{@glsref@#1}{\@this@childlabel}%
       }%
```

Does this child have children?

```
       \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
       {%
         \@glo@addchildren{#1}{\@this@childlabel}%
       }%
       {%
       }%
     }%
  \egroup
}
```

@do@sortentries

```
\newcommand*{\@glo@do@sortentries}[1]{%
  \ifglshasparent{#1}%
  {%
```

This entry has a parent, so add it to the child list

```
    \edef\@glo@parent{\csuse{glo@\glsdetoklabel{#1}@parent}}%
    \ifcsundef{@glo@sortingchildren@\@glo@parent}%
    {%
      \csdef{@glo@sortingchildren@\@glo@parent}{}%
    }%
    {}%
    \expandafter\@glo@sortedinsert
      \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
    \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%
    {%
```

Yes, it has so do nothing.

```
      }%
      {%
```

No, it hasn't so add it now.

```
        \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
      }%
    }%
    {%
      \@glo@sortedinsert{\@glo@sortinglist}{#1}%
    }%
}
```

glo@sortedinsert

\@glo@sortedinsert{⟨*list*⟩}{⟨*entry label*⟩}

Insert into list.

```
\newcommand*{\@glo@sortedinsert}[2]{%
  \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
}%
```

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).

orthandler@word

```
\newcommand*{\@glo@sorthandler@word}[2]{%
  \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
  \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
  \edef\glo@do@compare{%
    \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
    {\expandonce\@gls@sort@B}%
    {\expandonce\@gls@sort@A}%
  }%
  \glo@do@compare
}
```

thandler@letter

```
\newcommand*{\@glo@sorthandler@letter}[2]{%
  \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
  \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
  \edef\glo@do@compare{%
    \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
    {\expandonce\@gls@sort@B}%
    {\expandonce\@gls@sort@A}%
  }%
  \glo@do@compare
}
```

orthandler@case Case-sensitive sort.

```
\newcommand*{\@glo@sorthandler@case}[2]{%
  \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
  \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
  \edef\glo@do@compare{%
    \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
    {\expandonce\@gls@sort@B}%
    {\expandonce\@gls@sort@A}%
  }%
  \glo@do@compare
}
```

thandler@nocase Case-insensitive sort.

```
\newcommand*{\@glo@sorthandler@nocase}[2]{%
  \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
  \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
  \edef\glo@do@compare{%
    \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
    {\expandonce\@gls@sort@B}%
    {\expandonce\@gls@sort@A}%
  }%
  \glo@do@compare
}
```

@sortmacro@word Sort macro for 'word'

```
\newcommand*{\@glo@sortmacro@word}[1]{%
  \ifdefstring{\@glo@default@sorttype}{standard}%
  {%
    \@glo@sortentries{\@glo@sorthandler@word}{#1}%
  }%
  {%
    \PackageError{glossaries}{Conflicting sort options:^^J
     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
     \string\printnoidxglossary[sort=word]}{}%
  }%
}
```

ortmacro@letter Sort macro for 'letter'

```
\newcommand*{\@glo@sortmacro@letter}[1]{%
  \ifdefstring{\@glo@default@sorttype}{standard}%
  {%
    \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
  }%
  {%
    \PackageError{glossaries}{Conflicting sort options:^^J
     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
     \string\printnoidxglossary[sort=letter]}{}%
  }%
}
```

tmacro@standard Sort macro for 'standard'. (Use either 'word' or 'letter' order.)

```
\newcommand*{\@glo@sortmacro@standard}[1]{%
  \ifdefstring{\@glo@default@sorttype}{standard}%
  {%
    \ifcsdef{@glo@sorthandler@\glsorder}%
    {%
      \@glo@sortentries{\csuse{@glo@sorthandler@\glsorder}}{#1}%
    }%
    {%
      \PackageError{glossaries}{Unknown sort handler '\glsorder'}{}%
    }%
  }%
  {%
    \PackageError{glossaries}{Conflicting sort options:^^J
     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
     \string\printnoidxglossary[sort=standard]}{}%
  }%
}
```

@sortmacro@case Sort macro for 'case'

```
\newcommand*{\@glo@sortmacro@case}[1]{%
  \ifdefstring{\@glo@default@sorttype}{standard}%
  {%
    \@glo@sortentries{\@glo@sorthandler@case}{#1}%
  }%
  {%
    \PackageError{glossaries}{Conflicting sort options:^^J
     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
     \string\printnoidxglossary[sort=case]}{}%
  }%
}
```

ortmacro@nocase Sort macro for 'nocase'

```
\newcommand*{\@glo@sortmacro@nocase}[1]{%
  \ifdefstring{\@glo@default@sorttype}{standard}%
  {%
    \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
  }%
  {%
    \PackageError{glossaries}{Conflicting sort options:^^J
     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
     \string\printnoidxglossary[sort=nocase]}{}%
  }%
}
```

o@sortmacro@def Sort macro for 'def'. The order of definition is given in `\glolist@`⟨*type*⟩.

```
\newcommand*{\@glo@sortmacro@def}[1]{%
  \def\@glo@sortinglist{}%
  \forglsentries[#1]{\@gls@thislabel}%
  {%
    \xifinlistcs{\@gls@thislabel}{@glsref@#1}%
    {%
      \listeadd{\@glo@sortinglist}{\@gls@thislabel}%
    }%
    {%
```

Hasn't been referenced.

```
    }%
  }%
  \cslet{@glsref@#1}{\@glo@sortinglist}%
}
```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
\newcommand*{\@glo@sortmacro@def@do}[1]{%
  \ifinlistcs{#1}{@glsref@\@glo@type}%
  {}%
  {%
    \listcsadd{@glsref@\@glo@type}{#1}%
  }%
  \ifcsdef{@glo@sortingchildren@#1}%
  {%
    \@glo@addchildren{\@glo@type}{#1}%
  }%
  {}%
}
```

o@sortmacro@use Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
\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.

```
\newcommand*{\@print@noidx@glossary}{%
  \ifcsdef{@glsref@\@glo@type}%
  {%
```

Sort the entries:

```
    \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
    {%
      \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
    }%
    {%
       \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{}%
    }%
```

Do the glossary heading and preamble

```
    \glossarysection[\glossarytoctitle]{\glossarytitle}%
    \glossarypreamble
```

The glossary style might use a tabular-like environment, which may cause scoping problems when setting the current letter group. The predefined tabular-like styles don't support letter group headings, but there's nothing to stop the user from defining their own custom style that might, so any redefinition of this command within theglossary will have to be done globally.

```
    \def\@gls@currentlettergroup{}%
    \begin{theglossary}%
    \glossaryheader
    \glsresetentrylist
```

Iterate through the entries.

```
   \forlistcsloop{\@gls@noidx@do}{@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```
    \end{theglossary}%
    \glossarypostamble
  }%
  {%
    \@gls@noref@warn{\@glo@type}%
  }%
}
```

\glo@grabfirst

```
\def\glo@grabfirst#1#2\@nil{%
  \def\@gls@firsttok{#1}%
  \ifdefempty\@gls@firsttok
  {%
    \def\@glo@thislettergrp{0}%
  }%
  {%
```

Sanitize it:

```
    \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
    \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
  }%
}
```

\@glo@grabfirst

```
\def\@glo@grabfirst#1#2\@nil{%
  \ifdefempty\@glo@thislettergrp
  {%
     \def\@glo@thislettergrp{glssymbols}%
  }%
  {%
    \count@=\uccode`#1\relax
    \ifnum\count@=0\relax
      \def\@glo@thislettergrp{glssymbols}%
    \else
      \ifdefstring\@glo@sorttype{case}%
      {%
         \count@=`#1\relax
      }%
      {%
      }%
      \edef\@glo@thislettergrp{\the\count@}%
    \fi
  }%
}
```

\@gls@noidx@do Handler for list iteration used by `\@print@noidx@glossary`. The argument is the entry label. This only allows one sublevel.

```
\newcommand{\@gls@noidx@do}[1]{%
```

Get this entry's location list

```
  \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
```

Does this entry have a parent?

```
  \ifglshasparent{#1}%
  {%
```

Has a parent.

```
    \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
    \ifdefvoid{\@gls@loclist}
    {%
      \subglossentry{\gls@level}{#1}{}%
    }%
    {%
      \subglossentry{\gls@level}{#1}%
      {%
        \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
      }%
    }%
  }%
  {%
```

Doesn't have a parent Get this entry's sort key

```
    \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
    \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
    \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
    {}%
    {%
```

Do the group header:

```
      \ifdefempty{\@gls@currentlettergroup}{}%
      {%
```

The group skip may start a new scope, so make a global assignment.

```
        \global\let\@glo@thislettergrp\@glo@thislettergrp
        \glsgroupskip
      }%
      \glsgroupheading{\@glo@thislettergrp}%
    }%
    \global\let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
    \ifdefvoid{\@gls@loclist}
    {%
      \glossentry{#1}{}%
    }%
    {%
      \glossentry{#1}%
      {%
        \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
      }%
    }%
  }%
}
```

\glsnoidxloclist

\glsnoidxloclist{⟨*list cs*⟩}

Display location list.

```
\newcommand*{\glsnoidxloclist}[1]{%
  \def\@gls@noidxloclist@sep{}%
  \def\@gls@noidxloclist@prev{}%
  \forlistloop{\glsnoidxloclisthandler}{#1}%
}
```

xloclisthandler

Handler for location list iterator.

```
\newcommand*{\glsnoidxloclisthandler}[1]{%
  \ifdefstring{\@gls@noidxloclist@prev}{#1}%
  {%
```

Same as previous location so skip.

```
  }%
  {%
    \@gls@noidxloclist@sep
    #1%
    \def\@gls@noidxloclist@sep{\delimN}%
    \def\@gls@noidxloclist@prev{#1}%
  }%
}
```

yloclisthandler

Handler for location list iterator when used with \glsdisplaynumberlist.

```
\newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
  \ifdefstring{\@gls@noidxloclist@prev}{#1}%
  {%
```

Same as previous location so skip.

```
  }%
  {%
    \@gls@noidxloclist@sep
    \@gls@noidxloclist@prev
    \def\@gls@noidxloclist@prev{#1}%
  }%
}
```

snoidxdisplayloc

```
\glsnoidxdisplayloc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}
```

Display a location in the location list.

```
\newcommand*\glsnoidxdisplayloc[4]{%
  \setentrycounter[#1]{#2}%
  \csuse{#3}{#4}%
}
```

\@gls@reference

```
\@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.

```
\newcommand*{\@gls@reference}[3]{%
```

Add to label list

```
  \glsdoifexistsorwarn{#2}%
  {%
    \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
    \ifinlistcs{#2}{@glsref@#1}%
    {}%
    {\listcsgadd{@glsref@#1}{#2}}%
```

Add to location list

```
    \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
    {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}%
    {}%
    \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
  }%
}
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
\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.

```
\define@key{printgloss}{title}{%
 \def\glossarytitle{#1}%
 \let\gls@dotoctitle\relax
}
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
\define@key{printgloss}{toctitle}{%
 \def\glossarytoctitle{#1}%
 \let\gls@dotoctitle\relax
}
```

The style key sets the glossary style (but only for the given glossary).

```
\define@key{printgloss}{style}{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}%
    {Glossary style '#1' undefined}{}%
  }%
  {%
    \def\@glossarystyle{\setglossentrycompatibility
      \csname @glsstyle@#1\endcsname}%
  }%
}
```

The numberedsection key determines if this glossary should be in a numbered section.

```
\define@choicekey{printgloss}{numberedsection}[\val\nr]{%
false,nolabel,autolabel,nameref}[nolabel]{%
  \ifcase\nr\relax
    \renewcommand*{\@@glossarysecstar}{*}%
    \renewcommand*{\@@glossaryseclabel}{}%
  \or
    \renewcommand*{\@@glossarysecstar}{}%
    \renewcommand*{\@@glossaryseclabel}{}%
  \or
    \renewcommand*{\@@glossarysecstar}{}%
    \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
  \or
    \renewcommand*{\@@glossarysecstar}{*}%
    \renewcommand*{\@@glossaryseclabel}{%
      \protected@edef\@currentlabelname{\glossarytoctitle}%
      \label{\glsautoprefix\@glo@type}}%
  \fi
}
```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```
\define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
  \csuse{glsnogroupskip#1}%
}
```

The nopostdot key has the same effect as the package option of the same name.

```
\define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
  \csuse{glsnopostdot#1}%
}
```

nterLabelPrefix Make it easier to redefine the label prefix.

```
\newcommand*{\GlsEntryCounterLabelPrefix}{glsentry-}
```

The conditionals have been moved inside the appropriate commands to make it easier for the user to redefine them in the preamble and selectively switch the counter display on and off. Previously the helper commands were redefined by the entrycounter option, which would counteract any earlier customisation.

The entrycounter key is the same as the package option but localised to the current glossary.

```
\define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
  \csuse{glsentrycounter#1}%
  \@gls@define@glossaryentrycounter
}
```

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.

```
\define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
  \csuse{glssubentrycounter#1}%
  \@gls@define@glossarysubentrycounter
}
```

The nonumberlist key determines if this glossary should have a number list.

```
\define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
\ifglsnonumberlist
   \def\glossaryentrynumbers##1{}%
\else
   \def\glossaryentrynumbers##1{##1}%
\fi}
```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```
\define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}
```

@assign@sortkey Issue error if used with \printglossary

```
\newcommand*{\@glo@no@assign@sortkey}[1]{%
   \PackageError{glossaries}{‘sort’ key not permitted with
   \string\printglossary}%
   {The ‘sort’ key may only be used with \string\printnoidxglossary}%
}
```

@assign@sortkey For use with \printnoidxglossary

```
\newcommand*{\@@glo@assign@sortkey}[1]{%
  \def\@glo@sorttype{#1}%
}
```

@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 redefined.

```
\newcommand*{\@glsnonextpages}{%
  \gdef\glossaryentrynumbers##1{%
     \glsresetentrylist
  }%
}
```

\@glsnextpages Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```
\newcommand*{\@glsnextpages}{%
  \gdef\glossaryentrynumbers##1{%
     ##1\glsresetentrylist}}
```

sresetentrylist Resets `\glossaryentrynumbers`

```
\newcommand*{\glsresetentrylist}{%
  \global\let\glossaryentrynumbers\org@glossaryentrynumbers}
```

\glsnonextpages Outside of `\printglossary` this does nothing.

```
\newcommand*{\glsnonextpages}{}
```

\glsnextpages Outside of `\printglossary` this does nothing.

```
\newcommand*{\glsnextpages}{}
```

Process entrycounter and then subentrycounter options (this ensures the sub-counter can pick up the main counter as the master if required):

```
\@gls@define@glossaryentrycounter
\@gls@define@glossarysubentrycounter
```

subentrycounter Resets the glossarysubentry counter.

```
\newcommand*{\glsresetsubentrycounter}{%
  \ifglssubentrycounter
    \setcounter{glossarysubentry}{0}%
  \fi
}
```

subentrycounter Resets the glossaryentry counter.

```
\newcommand*{\glsresetentrycounter}{%
  \ifglsentrycounter
    \setcounter{glossaryentry}{0}%
  \fi
}
```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```
\newcommand*{\glsstepentry}[1]{%
  \ifglsentrycounter
    \refstepcounter{glossaryentry}%
    \label{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
  \fi
}
```

glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
\newcommand*{\glsstepsubentry}[1]{%
  \ifglssubentrycounter
    \edef\currentglssubentry{\glsdetoklabel{#1}}%
    \refstepcounter{glossarysubentry}%
    \label{\GlsEntryCounterLabelPrefix\currentglssubentry}%
  \fi
}
```

\glsrefentry Reference the entry or sub-entry counter if in use, otherwise just do \gls.

```
\newcommand*{\glsrefentry}[1]{%
 \ifglsentrycounter
   \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
 \else
   \ifglssubentrycounter
     \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
   \else
     \gls{#1}%
   \fi
 \fi
}
```

trycounterlabel Defines how to display the glossaryentry counter.

```
\newcommand*{\glsentrycounterlabel}{%
 \ifglsentrycounter
   \theglossaryentry.\space
 \fi
}
```

trycounterlabel Defines how to display the glossarysubentry counter.

```
\newcommand*{\glssubentrycounterlabel}{%
 \ifglssubentrycounter
   \theglossarysubentry)\space
 \fi
}
```

\glsentryitem Step and display glossaryentry counter, if appropriate.

```
\newcommand*{\glsentryitem}[1]{%
```

```
  \ifglsentrycounter
    \glsstepentry{#1}\glsentrycounterlabel
  \else
    \glsresetsubentrycounter
  \fi
}
```

glssubentryitem Step and display glossarysubentry counter, if appropriate.

```
\newcommand*{\glssubentryitem}[1]{%
  \ifglssubentrycounter
    \glsstepsubentry{#1}\glssubentrycounterlabel
  \fi
}
```

theglossary If the theglossary environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
\ifcsundef{theglossary}%
{%
  \newenvironment{theglossary}{}{}%
}%
{%
  \@gls@warnontheglossdefined
  \renewenvironment{theglossary}{}{}%
}
```

The glossary header is given by `\glossaryheader`. This forms part of the glossary style, and must indicate what should appear immediately after the start of the theglossary environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don't want a header row, the glossary style must redefine `\glossaryheader` to do nothing.

\glossaryheader

```
\newcommand*{\glossaryheader}{}
```

\glstarget `\glstarget{`⟨*label*⟩`}{`⟨*name*⟩`}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
\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.

atibleglossentry `\glossentry{`⟨*label*⟩`}{`⟨*page-list*⟩`}`

```
\providecommand*{\compatibleglossentry}[2]{%
  \toks@{#2}%
  \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
    {\noexpand\glsnamefont
       {\expandafter\expandonce\csname glo@#1@name\endcsname}}%
    {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
    {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
    {\the\toks@}%
  }%
  \@do@glossentry
}
```

\glossentryname

```
\newcommand*{\glossentryname}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
    \expandafter\glsnamefont\expandafter{\glo@name}%
  }%
}
```

\Glossentryname

```
\newcommand*{\Glossentryname}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glsnamefont{\Glsentryname{#1}}%
  }%
}
```

\glossentrydesc

```
\newcommand*{\glossentrydesc}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
     \glsentrydesc{#1}%
  }%
}
```

\Glossentrydesc

```
\newcommand*{\Glossentrydesc}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \Glsentrydesc{#1}%
  }%
}
```

lossentrysymbol

```
\newcommand*{\glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
```

```
    \glsentrysymbol{#1}%
  }%
}
```

lossentrysymbol

```
\newcommand*{\Glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \Glsentrysymbol{#1}%
  }%
}
```

blesubglossentry

```
\subglossentry{⟨level⟩}{⟨label⟩}{⟨page-list⟩}
```

```
\providecommand*{\compatiblesubglossentry}[3]{%
  \toks@{#3}%
  \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
  {#2}%
    {\noexpand\glsnamefont
      {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
    {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
    {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
    {\the\toks@}%
  }%
  \@do@subglossentry
}
```

rycompatibility

```
\newcommand*{\setglossentrycompatibility}{%
  \let\glossentry\compatibleglossentry
  \let\subglossentry\compatiblesubglossentry
}
\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.

```
\newcommand{\glossaryentryfield}[5]{%
  \GlossariesWarning
  {Deprecated use of \string\glossaryentryfield.^^J
   I recommend you change to \string\glossentry.^^J
   If you've just upgraded, try removing your gls auxiliary
   files^^J and recompile}%
  \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

arysubentryfield

```
\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.)

```
\newcommand*{\glossarysubentryfield}[6]{%
  \GlossariesWarning
  {Deprecated use of \string\glossarysubentryfield.^^J
   I recommend you change to \string\subglossentry.^^J
   If you've just upgraded, try removing your gls auxiliary
   files^^J and recompile}%
  \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

```
\newcommand*{\glsgroupskip}{}
```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glssymbols`, `glsnumbers`, `A`, ..., `Z`. Glossary styles must redefined this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

glsgroupheading

```
\newcommand*{\glsgroupheading}[1]{}
```

It is possible to "trick" `makeindex` into treating entries as though they belong to the same group, even if the terms don't start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an `a`, while entries belonging to another group could be defined so that the sort key starts with a `b`, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

```
\glsgetgrouptitle{⟨label⟩}
```

This command produces the title for the glossary group whose label is given by ⟨*label*⟩. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, `A`, . . . , `Z`. If you want to redefine the group titles, you will need to redefine this command. 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`

```
\newcommand*{\glsgetgrouptitle}[1]{%
  \@gls@getgrouptitle{#1}{\@gls@grptitle}%
  \@gls@grptitle
}
```

`s@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
\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.

```
 \dtl@ifsingle{#1}%
 {%
   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
 }%
 {%
   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
            or test{\ifstrequal{#1}{glsnumbers}}}%
   {%
     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
   }%
   {%
     \def#2{#1}%
   }%
 }%
}
```

`x@getgrouptitle` Version for the no-indexing app option:

```
\newcommand*{\@gls@noidx@getgrouptitle}[2]{%
  \DTLifint{#1}%
  {\edef#2{\char#1\relax}}%
  {%
    \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
  }%
}
```

`\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`.

lsgetgrouplabel

```
\newcommand*{\glsgetgrouplabel}[1]{%
\ifthenelse{\equal{#1}{\glssymbolsgroupname}}{glssymbols}{%
\ifthenelse{\equal{#1}{\glsnumbersgroupname}}{glsnumbers}{#1}}}
```

The command `\setentrycounter` sets the entry's associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

setentrycounter

```
\newcommand*{\setentrycounter}[2][]{%
  \def\@glo@counterprefix{#1}%
  \ifx\@glo@counterprefix\@empty
    \def\@glo@counterprefix{.}%
  \else
    \def\@glo@counterprefix{.#1.}%
  \fi
  \def\glsentrycounter{#2}%
}
```

The current glossary style can be set using `\setglossarystyle{`⟨*style*⟩`}`.

etglossarystyle

```
\newcommand*{\setglossarystyle}[1]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}{Glossary style '#1' undefined}{}%
  }%
  {%
    \csname @glsstyle@#1\endcsname
  }%
```

Set the default style if it's not already set.

```
  \ifx\@glossary@default@style\relax
    \protected@edef\@glossary@default@style{#1}%
  \fi
}
```

\glossarystyle

```
\newcommand*{\glossarystyle}[1]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}{Glossary style '#1' undefined}{}%
  }%
  {%
    \GlossariesWarning
    {Deprecated command \string\glossarystyle.^^J
     I recommend you switch to \string\setglossarystyle\space unless
     you want to maintain backward compatibility}%
    \setglossentrycompatibility
    \csname @glsstyle@#1\endcsname
    \ifcsdef{@glscompstyle@#1}%
    {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
    {}%
  }%
```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```
  \ifx\@glossary@default@style\relax
    \protected@edef\@glossary@default@style{#1}%
  \fi
}
```

ewglossarystyle

New glossary styles can be defined using:

`\newglossarystyle{`⟨*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 `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```
\newcommand{\newglossarystyle}[2]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
  }%
  {%
    \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
  }%
}
```

ewglossarystyle

Code for this macro supplied by Marco Daniel.

```
\newcommand{\renewglossarystyle}[2]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
  }%
  {%
    \csdef{@glsstyle@#1}{#2}%
  }%
}
```

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

```
\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

```
\ifcsundef{hyperlink}%
{%
  \def\glshypernumber#1{#1}%
}%
{%
  \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}\@nil}
}
```

@glshypernumber This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
\def\@glshypernumber#1\nohyperpage#2#3\@nil{%
  \ifx\\#1\\%
  \else
    \@delimR#1\delimR\delimR\\%
  \fi
  \ifx\\#2\\%
  \else
    #2%
  \fi
  \ifx\\#3\\%
  \else
    \@glshypernumber#3\@nil
  \fi
}
```

`\@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

```
\def\@delimR#1\delimR #2\delimR #3\\{%
\ifx\\#2\\%
  \@delimN{#1}%
\else
  \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
\fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

\@delimN

```
\def\@delimN#1{\@@delimN#1\delimN \delimN\\}
\def\@@delimN#1\delimN #2\delimN#3\\{%
\ifx\\#3\\%
  \@gls@numberlink{#1}%
\else
  \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
\fi
}
```

The following code is modified from hyperref's `\HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```
\def\@gls@numberlink#1{%
\begingroup
 \toks@={}%
 \@gls@removespaces#1 \@nil
\endgroup}
\def\@gls@removespaces#1 #2\@nil{%
 \toks@=\expandafter{\the\toks@#1}%
 \ifx\\#2\\%
   \edef\x{\the\toks@}%
   \ifx\x\empty
   \else

     \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
               {\the\toks@}%
   \fi
 \else
   \@gls@ReturnAfterFi{%
     \@gls@removespaces#2\@nil
   }%
 \fi
}
\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

```
\newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}
```

\hypersf

```
\newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}
```

\hypertt

```
\newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}
```

\hyperbf

```
\newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}
```

\hypermd

```
\newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}
```

\hyperit

```
\newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}
```

\hypersl

```
\newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}
```

\hyperup

```
\newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}
```

\hypersc

```
\newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

\hyperemph

```
\newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}
```

1.17 Acronyms

\oldacronym

```
\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.

```
\newcommand{\oldacronym}[4][\gls@label]{%
  \def\gls@label{#2}%
  \newacronym[#4]{#1}{#2}{#3}%
  \ifcsundef{xspace}%
  {%
    \expandafter\edef\csname#1\endcsname{%
      \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}%
    }%
  }%
  {%
    \expandafter\edef\csname#1\endcsname{%
      \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
      \noexpand\gls{#1}\noexpand\xspace}%
    }%
  }%
}
```

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

```
\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.

```
\newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If garamondx has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
\newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
\newcommand*{\glsshortkey}{short}
```

`sshortpluralkey`

```
\newcommand*{\glsshortpluralkey}{shortplural}
```

\glslongkey

```
\newcommand*{\glslongkey}{long}
```

lslongpluralkey

```
\newcommand*{\glslongpluralkey}{longplural}
```

\acrfull Full form of the acronym.

```
\newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}
```

```
\newcommand*\ns@acrfull[2][]{%
  \new@ifnextchar[{\@acrfull{#1}{#2}}%
                  {\@acrfull{#1}{#2}[]}%
}
```

\@acrfull Low-level macro:

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

Make it easier for acronym styles to change this:

```
  \acrfullfmt{#1}{#2}{#3}%
}
```

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.

```
\newcommand*{\acrfullfmt}[3]{%
  \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
}
```

rlinkfullformat Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{`⟨*long cs*⟩`}{`⟨*short cs*⟩`}` `{`⟨*options*⟩`}{`⟨*label*⟩`}{`⟨*insert*⟩`}`

```
\newcommand{\acrlinkfullformat}[5]{%
  \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%
}
```

\acrfullformat Default full form is ⟨*long*⟩ `(`⟨*short*⟩`)`.

```
\newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

\glsspace Robust space to ensure it's written to the `.glsdefs` file.

```
\newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

\Acrfull

```
\newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}
```

```
\newcommand*\ns@Acrfull[2][]{%
  \new@ifnextchar[{\@Acrfull{#1}{#2}}%
                  {\@Acrfull{#1}{#2}[]}%
}
```

Low-level macro:

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

Make it easier for acronym styles to change this:

```
  \Acrfullfmt{#1}{#2}{#3}%
}
```

\Acrfullfmt First letter upper case full format.

```
\newcommand*{\Acrfullfmt}[3]{%
  \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
}
```

\ACRfull

```
\newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}
```

```
\newcommand*\ns@ACRfull[2][]{%
  \new@ifnextchar[{\@ACRfull{#1}{#2}}%
                  {\@ACRfull{#1}{#2}[]}%
}
```

Low-level macro:

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

Make it easier for acronym styles to change this:

```
  \ACRfullfmt{#1}{#2}{#3}%
}
```

\ACRfullfmt All upper case full format.

```
\newcommand*{\ACRfullfmt}[3]{%
  \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
}
```

Plural:

\acrfullpl

```
\newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}
```

```
\newcommand*\ns@acrfullpl[2][]{%
  \new@ifnextchar[{\@acrfullpl{#1}{#2}}%
                  {\@acrfullpl{#1}{#2}[]}%
}
```

Low-level macro:

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

Make it easier for acronym styles to change this:

```
  \acrfullplfmt{#1}{#2}{#3}%
}
```

\acrfullplfmt No case change plural full format.

```
\newcommand*{\acrfullplfmt}[3]{%
  \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
}
```

\Acrfullpl

```
\newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}
```

```
\newcommand*\ns@Acrfullpl[2][]{%
  \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%
                  {\@Acrfullpl{#1}{#2}[]}%
}
```

Low-level macro:

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

Make it easier for acronym styles to change this:

```
  \Acrfullplfmt{#1}{#2}{#3}%
}
```

\Acrfullplfmt First letter upper case plural full format.

```
\newcommand*{\Acrfullplfmt}[3]{%
  \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
}
```

\ACRfullpl

```
\newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}
```

```
\newcommand*\ns@ACRfullpl[2][]{%
  \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
                  {\@ACRfullpl{#1}{#2}[]}%
}
```

Low-level macro:

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

Make it easier for acronym styles to change this:

```
  \ACRfullplfmt{#1}{#2}{#3}%
}
```

\ACRfullplfmt All upper case plural full format.

```
\newcommand*{\ACRfullplfmt}[3]{%
  \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
}
```

1.18 Predefined acronym styles

\acronymfont This is only used with the additional acronym styles:

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

irstacronymfont This is only used with the additional acronym styles:

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

\acrnameformat The styles that allow an additional description use \acrnameformat{⟨*short*⟩}{⟨*long*⟩} to determine what information is displayed in the name.

```
\newcommand*{\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by \newacronym:

\glskeylisttok

```
\newtoks\glskeylisttok
```

\glslabeltok

```
\newtoks\glslabeltok
```

\glsshorttok

```
\newtoks\glsshorttok
```

\glslongtok

```
\newtoks\glslongtok
```

\newacronymhook Provide a hook for \newacronym:

```
\newcommand*{\newacronymhook}{}
```

nericNewAcronym New improved version of setting the acronym style.

```
\newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc

```
  \let\@Gls@entryname\@Gls@acrentryname
```

Change the way acronyms are defined:

```
  \renewcommand{\newacronym}[4][]{%
    \ifdefempty{\@glsacronymlists}%
    {%
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
    }%
    {}%
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \protected@edef\@do@newglossaryentry{%
      \noexpand\newglossaryentry{\the\glslabeltok}%
      {%
        type=\acronymtype,%
        name={\expandonce{\acronymentry{##2}}},%
        sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
        text={\the\glsshorttok},%
        short={\the\glsshorttok},%
        shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
        long={\the\glslongtok},%
        longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
        \GenericAcronymFields,%
        \the\glskeylisttok
      }%
    }%
    \@do@newglossaryentry
  }%
```

Make sure that `\acrfull` etc reflects the new style:

```
  \renewcommand*{\acrfullfmt}[3]{%
    \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullfmt}[3]{%
    \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullfmt}[3]{%
    \glslink[##1]{##2}{%
      \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
  \renewcommand*{\acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullplfmt}[3]{%
    \glslink[##1]{##2}{%
      \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%
```

Make sure that `\glsentryfull` etc reflects the new style:

```
  \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
  \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
}
```

`icAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```
\newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}
```

`\acronymentry`

`\acronymentry{⟨label⟩}`

Display style for the name field in the list of acronyms.

```
\newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}
```

\acronymsort

```
\acronymsort{⟨short⟩}{⟨long⟩}
```

Default sort format for acronyms.

```
\newcommand*{\acronymsort}[2]{#1}
```

\setacronymstyle

```
\setacronymstyle{⟨style name⟩}
```

```
\newcommand*{\setacronymstyle}[1]{%
  \ifcsundef{@glsacr@dispstyle@#1}
  {%
    \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
  }%
  {%
    \ifdefempty{\@glsacronymlists}%
    {%
      \DeclareAcronymList{\acronymtype}%
    }%
    {}%
    \SetGenericNewAcronym
    \GlsUseAcrStyleDefs{#1}%
    \@for\@gls@type:=\@glsacronymlists\do{%
      \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
    }%
  }%
}
```

\newacronymstyle

```
\newacronymstyle{⟨style name⟩}{⟨entry format definition⟩}{⟨display
definitions⟩}
```

Defines a new acronym style called ⟨*style name*⟩.

```
\newcommand*{\newacronymstyle}[3]{%
  \ifcsdef{@glsacr@dispstyle@#1}%
  {%
    \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
  }%
  {%
    \csdef{@glsacr@dispstyle@#1}{#2}%
    \csdef{@glsacr@styledefs@#1}{#3}%
  }%
}
```

newacronymstyle Redefines the given acronym style.

```
\newcommand*{\renewacronymstyle}[3]{%
  \ifcsdef{@glsacr@dispstyle@#1}%
  {%
```

```
    \csdef{@glsacr@dispstyle@#1}{#2}%
    \csdef{@glsacr@styledefs@#1}{#3}%
  }%
  {%
    \PackageError{glossaries}{Acronym style ‘#1’ doesn't exist}{}%
  }%
}
```

rEntryDispStyle

```
\newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

UseAcrStyleDefs

```
\newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

`long-short` ⟨*long*⟩ (⟨*short*⟩) acronym style.

```
\newacronymstyle{long-short}%
{%
```

Check for long form in case this is a mixed glossary.

```
  \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
}%
{%
  \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
  \renewcommand*{\genacrfullformat}[2]{%
   \glsentrylong{##1}##2\space
   (\protect\firstacronymfont{\glsentryshort{##1}})%
  }%
  \renewcommand*{\Genacrfullformat}[2]{%
   \Glsentrylong{##1}##2\space
   (\protect\firstacronymfont{\glsentryshort{##1}})%
  }%
  \renewcommand*{\genplacrfullformat}[2]{%
   \glsentrylongpl{##1}##2\space
   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
  }%
  \renewcommand*{\Genplacrfullformat}[2]{%
   \Glsentrylongpl{##1}##2\space
   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
  }%
  \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
  \renewcommand*{\acronymsort}[2]{##1}%
  \renewcommand*{\acronymfont}[1]{##1}%
  \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
}
```

`long-sp-short` Similar to the previous style but allows the space between the long and short form to be customized.

```
\newacronymstyle{long-sp-short}%
{%
```

Check for long form in case this is a mixed glossary.

```
  \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
}%
{%
  \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
  \renewcommand*{\genacrfullformat}[2]{%
   \glsentrylong{##1}##2\glsacspace{##1}%
   (\protect\firstacronymfont{\glsentryshort{##1}})%
  }%
  \renewcommand*{\Genacrfullformat}[2]{%
   \Glsentrylong{##1}##2\glsacspace{##1}%
   (\protect\firstacronymfont{\glsentryshort{##1}})%
  }%
  \renewcommand*{\genplacrfullformat}[2]{%
   \glsentrylongpl{##1}##2\glsacspace{##1}%
   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
  }%
  \renewcommand*{\Genplacrfullformat}[2]{%
   \Glsentrylongpl{##1}##2\glsacspace{##1}%
   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
  }%
  \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
  \renewcommand*{\acronymsort}[2]{##1}%
  \renewcommand*{\acronymfont}[1]{##1}%
  \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
}
```

`\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.

```
\newcommand*{\glsacspace}[1]{%
  \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
  \ifdim\dimen@<3em~\else\space\fi
}
```

`short-long` ⟨*short*⟩ (⟨*long*⟩) acronym style.

```
\newacronymstyle{short-long}%
{%
```

Check for long form in case this is a mixed glossary.

```
  \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
}%
{%
  \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
  \renewcommand*{\genacrfullformat}[2]{%
   \protect\firstacronymfont{\glsentryshort{##1}}##2\space
   (\glsentrylong{##1})%
  }%
  \renewcommand*{\Genacrfullformat}[2]{%
   \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
   (\glsentrylong{##1})%
  }%
  \renewcommand*{\genplacrfullformat}[2]{%
   \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
   (\glsentrylongpl{##1})%
  }%
  \renewcommand*{\Genplacrfullformat}[2]{%
   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
   (\glsentrylongpl{##1})%
  }%

  \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
  \renewcommand*{\acronymsort}[2]{##1}%
  \renewcommand*{\acronymfont}[1]{##1}%
  \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
}
```

`long-sc-short` ⟨*long*⟩ (`\textsc{`⟨*short*⟩`}`) acronym style.

```
\newacronymstyle{long-sc-short}%
{%
  \GlsUseAcrEntryDispStyle{long-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-short}%
  \renewcommand{\acronymfont}[1]{\textsc{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
}
```

`long-sm-short` ⟨*long*⟩ (`\textsmaller{`⟨*short*⟩`}`) acronym style.

```
\newacronymstyle{long-sm-short}%
{%
  \GlsUseAcrEntryDispStyle{long-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-short}%
  \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
}
```

`sc-short-long` ⟨*short*⟩ (`\textsc{`⟨*long*⟩`}`) acronym style.

```
\newacronymstyle{sc-short-long}%
{%
  \GlsUseAcrEntryDispStyle{short-long}%
}%
{%
```

```
  \GlsUseAcrStyleDefs{short-long}%
  \renewcommand{\acronymfont}[1]{\textsc{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
}
```

sm-short-long ⟨*short*⟩ (`\textsmaller{`⟨*long*⟩`}`) acronym style.

```
\newacronymstyle{sm-short-long}%
{%
  \GlsUseAcrEntryDispStyle{short-long}%
}%
{%
  \GlsUseAcrStyleDefs{short-long}%
  \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
}
```

long-short-desc ⟨*long*⟩ (`{`⟨*short*⟩`}`) acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{long-short-desc}%
{%
  \GlsUseAcrEntryDispStyle{long-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-short}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

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`.

```
\newacronymstyle{long-sp-short-desc}%
{%
  \GlsUseAcrEntryDispStyle{long-sp-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-sp-short}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\glsacspace{##1}(\acronymfont{\glsentryshort{##1}})}%
}
```

g-sc-short-desc ⟨*long*⟩ (`\textsc{`⟨*short*⟩`}`) acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{long-sc-short-desc}%
{%
```

```
  \GlsUseAcrEntryDispStyle{long-sc-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-sc-short}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

`g-sm-short-desc` ⟨*long*⟩ (`\textsmaller{`⟨*short*⟩`}`) acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{long-sm-short-desc}%
{%
  \GlsUseAcrEntryDispStyle{long-sm-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-sm-short}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

`short-long-desc` ⟨*short*⟩ (`{`⟨*long*⟩`}`) acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{short-long-desc}%
{%
  \GlsUseAcrEntryDispStyle{short-long}%
}%
{%
  \GlsUseAcrStyleDefs{short-long}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

`short-long-desc` ⟨*long*⟩ (`\textsc{`⟨*short*⟩`}`) acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{sc-short-long-desc}%
{%
  \GlsUseAcrEntryDispStyle{sc-short-long}%
}%
{%
  \GlsUseAcrStyleDefs{sc-short-long}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
```

```
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

short-long-desc ⟨*long*⟩ (\textsmaller{⟨*short*⟩}) acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{sm-short-long-desc}%
{%
  \GlsUseAcrEntryDispStyle{sm-short-long}%
}%
{%
  \GlsUseAcrStyleDefs{sm-short-long}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

dua ⟨*long*⟩ only acronym style.

```
\newacronymstyle{dua}%
{%
```

Check for long form in case this is a mixed glossary.

```
  \ifdefempty\glscustomtext
  {%
    \ifglshaslong{\glslabel}%
    {%
      \glsifplural
      {%
```

Plural form:

```
        \glscapscase
        {%
```

Plural form, don't adjust case:

```
          \glsentrylongpl{\glslabel}\glsinsert
        }%
        {%
```

Plural form, make first letter upper case:

```
          \Glsentrylongpl{\glslabel}\glsinsert
        }%
        {%
```

Plural form, all caps:

```
          \mfirstucMakeUppercase
            {\glsentrylongpl{\glslabel}\glsinsert}%
        }%
      }%
      {%
```

Singular form

```
          \glscapscase
          {%
```

Singular form, don't adjust case:

```
            \glsentrylong{\glslabel}\glsinsert
          }%
          {%
```

Subsequent singular form, make first letter upper case:

```
            \Glsentrylong{\glslabel}\glsinsert
          }%
          {%
```

Subsequent singular form, all caps:

```
            \mfirstucMakeUppercase
              {\glsentrylong{\glslabel}\glsinsert}%
          }%
        }%
      }%
      {%
```

Not an acronym:

```
        \glsgenentryfmt
      }%
    }%
    {\glscustomtext\glsinsert}%
  }%
  {%
    \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

    \renewcommand*{\acrfullfmt}[3]{%
      \glslink[##1]{##2}{\glsentrylong{##2}##3\space
        (\acronymfont{\glsentryshort{##2}})}}%
    \renewcommand*{\Acrfullfmt}[3]{%
      \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
        (\acronymfont{\glsentryshort{##2}})}}%
    \renewcommand*{\ACRfullfmt}[3]{%
      \glslink[##1]{##2}{%
        \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
        (\acronymfont{\glsentryshort{##2}})}}}%

    \renewcommand*{\acrfullplfmt}[3]{%
      \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
        (\acronymfont{\glsentryshortpl{##2}})}}%

    \renewcommand*{\Acrfullplfmt}[3]{%
      \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
        (\acronymfont{\glsentryshortpl{##2}})}}%
    \renewcommand*{\ACRfullplfmt}[3]{%
      \glslink[##1]{##2}{%
        \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
        (\acronymfont{\glsentryshortpl{##2}})}}}%
    \renewcommand*{\glsentryfull}[1]{%
      \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
    }%
    \renewcommand*{\Glsentryfull}[1]{%
      \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
    }%
    \renewcommand*{\glsentryfullpl}[1]{%
      \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
    }%
    \renewcommand*{\Glsentryfullpl}[1]{%
      \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
    }%
    \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
    \renewcommand*{\acronymsort}[2]{##1}%
    \renewcommand*{\acronymfont}[1]{##1}%
    \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
  }
```

`dua-desc` ⟨*long*⟩ only acronym style with user-supplied description.

```
\newacronymstyle{dua-desc}%
{%
  \GlsUseAcrEntryDispStyle{dua}%
}%
{%
  \GlsUseAcrStyleDefs{dua}%
  \renewcommand*{\GenericAcronymFields}{}%

  \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
  \renewcommand*{\acronymsort}[2]{##2}%
}%
```

`footnote` ⟨*short*⟩`\footnote{`⟨*long*⟩`}` acronym style.

```
\newacronymstyle{footnote}%
{%
```

Check for long form in case this is a mixed glossary.

```
  \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
}%
{%
  \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
  \glshyperfirstfalse
  \renewcommand*{\genacrfullformat}[2]{%
   \protect\firstacronymfont{\glsentryshort{##1}}##2%
   \protect\footnote{\glsentrylong{##1}}%
  }%
  \renewcommand*{\Genacrfullformat}[2]{%
   \firstacronymfont{\Glsentryshort{##1}}##2%
   \protect\footnote{\glsentrylong{##1}}%
  }%
  \renewcommand*{\genplacrfullformat}[2]{%
   \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
   \protect\footnote{\glsentrylongpl{##1}}%
  }%
  \renewcommand*{\Genplacrfullformat}[2]{%
   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
   \protect\footnote{\glsentrylongpl{##1}}%
  }%
  \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
  \renewcommand*{\acronymsort}[2]{##1}%
  \renewcommand*{\acronymfont}[1]{##1}%
  \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
```

Don't use footnotes for `\acrfull`:

```
 \renewcommand*{\acrfullfmt}[3]{%
   \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
     (\glsentrylong{##2})}}%
 \renewcommand*{\Acrfullfmt}[3]{%
   \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
     (\glsentrylong{##2})}}%
 \renewcommand*{\ACRfullfmt}[3]{%
   \glslink[##1]{##2}{%
     \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
     (\glsentrylong{##2})}}}%
 \renewcommand*{\acrfullplfmt}[3]{%
   \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
     (\glsentrylongpl{##2})}}%
 \renewcommand*{\Acrfullplfmt}[3]{%
   \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
     (\glsentrylongpl{##2})}}%
 \renewcommand*{\ACRfullplfmt}[3]{%
   \glslink[##1]{##2}{%
     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
     (\glsentrylongpl{##2})}}}%
```

Similarly for `\glsentryfull` etc:

```
 \renewcommand*{\glsentryfull}[1]{%
    \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
 \renewcommand*{\Glsentryfull}[1]{%
    \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
 \renewcommand*{\glsentryfullpl}[1]{%
    \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
 \renewcommand*{\Glsentryfullpl}[1]{%
    \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
}
```

`footnote-sc` `\textsc{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style.

```
\newacronymstyle{footnote-sc}%
{%
  \GlsUseAcrEntryDispStyle{footnote}%
}%
{%
  \GlsUseAcrStyleDefs{footnote}%
  \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
  \renewcommand{\acronymfont}[1]{\textsc{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
}%
```

`footnote-sm` `\textsmaller{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style.

```
\newacronymstyle{footnote-sm}%
{%
  \GlsUseAcrEntryDispStyle{footnote}%
}%
{%
  \GlsUseAcrStyleDefs{footnote}%
  \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
  \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
}%
```

`footnote-desc` ⟨*short*⟩`\footnote{`⟨*long*⟩`}` acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{footnote-desc}%
{%
  \GlsUseAcrEntryDispStyle{footnote}%
}%
{%
  \GlsUseAcrStyleDefs{footnote}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

`ootnote-sc-desc` `\textsc{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{footnote-sc-desc}%
{%
  \GlsUseAcrEntryDispStyle{footnote-sc}%
}%
{%
  \GlsUseAcrStyleDefs{footnote-sc}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

ootnote-sm-desc `\textsmaller{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style that has an accompanying description (which the user needs to supply).

```
\newacronymstyle{footnote-sm-desc}%
{%
  \GlsUseAcrEntryDispStyle{footnote-sm}%
}%
{%
  \GlsUseAcrStyleDefs{footnote-sm}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
}
```

AcronymSynonyms

```
\newcommand*{\DefineAcronymSynonyms}{%
```

Short form

\acs

```
  \let\acs\acrshort
```

First letter uppercase short form

\Acs

```
  \let\Acs\Acrshort
```

Plural short form

\acsp

```
  \let\acsp\acrshortpl
```

First letter uppercase plural short form

\Acsp

```
  \let\Acsp\Acrshortpl
```

Long form

\acl

```
  \let\acl\acrlong
```

Plural long form

\aclp

```
  \let\aclp\acrlongpl
```

First letter upper case long form

\Acl

```
  \let\Acl\Acrlong
```

First letter upper case plural long form

\Aclp

```
  \let\Aclp\Acrlongpl
```

Full form

\acf

```
  \let\acf\acrfull
```

Plural full form

\acfp

```
  \let\acfp\acrfullpl
```

First letter upper case full form

\Acf

```
  \let\Acf\Acrfull
```

First letter upper case plural full form

\Acfp

```
  \let\Acfp\Acrfullpl
```

Standard form

\ac

```
  \let\ac\gls
```

First upper case standard form

\Ac

```
  \let\Ac\Gls
```

Standard plural form

\acp

```
  \let\acp\glspl
```

Standard first letter upper case plural form

\Acp

```
  \let\Acp\Glspl
}
```

Define synonyms if required

```
\ifglsacrshortcuts
  \DefineAcronymSynonyms
\fi
```

These commands for setting the style are now deprecated but are kept for backward compatibility.

nymDisplayStyle Sets the default acronym display style for given glossary.

```
\newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
  \defglsentryfmt[#1]{\glsgenentryfmt}%
}
```

ltNewAcronymDef Sets up the acronym definition for the default style. The information is provided by the tokens \glslabeltok, \glsshorttok, \glslongtok and \glskeylisttok.

```
\newcommand*{\DefaultNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\the\glsshorttok},%
      sort={\the\glsshorttok},%
      text={\the\glsshorttok},%
      first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
      plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
                                 {\noexpand\expandonce\noexpand\@glo@shortpl}},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\the\glslongtok},%
      descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@descplural\gls@assign@descplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@descplural##1##2{%
    \@@gls@expand@field{##1}{descplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

ultAcronymStyle Set up the default acronym style:

```
\newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetDefaultAcronymDisplayStyle{\@gls@type}%
  }%
```

Set up the definition of \newacronym:

```
  \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).

```
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetDefaultAcronymDisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \DefaultNewAcronymDef
  }%
  \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
}
```

\acrfootnote Used by the footnote acronym styles.

```
\newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```
\newcommand*{\acrlinkfootnote}[3]{%
  \footnote{\glslink[#1]{#2}{#3}}%
}
```

rnolinkfootnote

```
\newcommand*{\acrnolinkfootnote}[3]{%
  \footnote{#3}%
}
```

nymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
\newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
  \defglsentryfmt[#1]{%

    \ifdefempty\glscustomtext
    {%
      \ifglsused{\glslabel}%
      {%
        \acronymfont{\glsgenentryfmt}%
      }%
      {%
        \firstacronymfont{\glsgenentryfmt}%
        \ifglshassymbol{\glslabel}%
        {%
          \expandafter\protect\expandafter\acrfootnote\expandafter
           {\@gls@link@opts}{\@gls@link@label}%
           {%
            \glsifplural
              {\glsentrysymbolplural{\glslabel}}%
              {\glsentrysymbol{\glslabel}}%
           }%
        }%
      }%
    }%
    {\glscustomtext\glsinsert}%
  }%
}
```

teNewAcronymDef

```
\newcommand*{\DescriptionFootnoteNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      sort={\the\glsshorttok},%
      first={\the\glsshorttok},%
      firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      text={\the\glsshorttok},%
      plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      symbol={\the\glslongtok},%
      symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
```

```
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

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.

```
\newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
  \renewcommand{\newacronym}[4][]{%
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \DescriptionFootnoteNewAcronymDef
  }%
```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
  }%
```

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.

```
  \ifglsacrsmallcaps
    \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
    \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
  \else
    \ifglsacrsmaller
      \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
    \fi
  \fi
```

Check for package option clash

```
  \ifglsacrdua
    \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
    can’t both be set}{}%
  \fi
}%
```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```
\newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
  \defglsentryfmt[#1]{\glsgenentryfmt}%
}
```

UANewAcronymDef

```
\newcommand*{\DescriptionDUANewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\the\glslongtok},%
      sort={\the\glslongtok},
      text={\the\glslongtok},%
      first={\the\glslongtok},%
      plural={\noexpand\expandonce\noexpand\@glo@longpl},%
      firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      symbol={\the\glsshorttok},%
      symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

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.

```
\newcommand*{\SetDescriptionDUAAcronymStyle}{%
  \ifglsacrsmallcaps
    \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
    can’t both be set}{}%
  \else
    \ifglsacrsmaller
      \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
      can’t both be set}{}%
    \fi
  \fi
  \renewcommand{\newacronym}[4][]{%
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \DescriptionDUANewAcronymDef
  }%
```

Set display.

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
  }%
}%
```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```
\newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
  \defglsentryfmt[#1]{%

    \ifdefempty\glscustomtext
    {%
      \ifglsused{\glslabel}%
      {%
```

Move the inserted text outside of \acronymfont

```
        \let\gls@org@insert\glsinsert
        \let\glsinsert\@empty
        \acronymfont{\glsgenentryfmt}\gls@org@insert
      }%
      {%
        \glsgenentryfmt
        \ifglshassymbol{\glslabel}%
          {%
             \glsifplural
             {%
               \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
             }%
             {%
               \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
             }%
             \space(\protect\firstacronymfont
             {\glscapscase
              {\@glo@symbol}
              {\@glo@symbol}
              {\mfirstucMakeUppercase{\@glo@symbol}}})%
          }%
          {}%
      }%
    }%
    {\glscustomtext\glsinsert}%
  }%
}
```

onNewAcronymDef

```
\newcommand*{\DescriptionNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand
        \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
      sort={\the\glsshorttok},%
      first={\the\glslongtok},%
      firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      text={\the\glsshorttok},%
      plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      symbol={\noexpand\@glo@text},%
      symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      \the\glskeylisttok}%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \def\gls@assign@firstpl##1##2{%
```

```
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

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 `\acrnameformat` to allow the user to override the way the name is displayed in the list of acronyms.

```
\newcommand*{\SetDescriptionAcronymStyle}{%
  \renewcommand{\newacronym}[4][]{%
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetDescriptionAcronymDisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \DescriptionNewAcronymDef
  }%
```

Set display.

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetDescriptionAcronymDisplayStyle{\@gls@type}%
  }%
```

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.

```
  \ifglsacrsmallcaps
    \renewcommand{\acronymfont}[1]{\textsc{##1}}
    \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
  \else
    \ifglsacrsmaller
      \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
    \fi
  \fi
}%
```

nymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```
\newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
  \defglsentryfmt[#1]{%
    \ifdefempty\glscustomtext
    {%
```

Move the inserted text outside of \acronymfont

```
      \let\gls@org@insert\glsinsert
      \let\glsinsert\@empty
      \ifglsused{\glslabel}%
      {%
        \acronymfont{\glsgenentryfmt}\gls@org@insert
      }%
      {%
        \firstacronymfont{\glsgenentryfmt}\gls@org@insert
        \ifglshaslong{\glslabel}%
        {%
          \expandafter\protect\expandafter\acrfootnote\expandafter
           {\@gls@link@opts}{\@gls@link@label}%
           {%
            \glsifplural
              {\glsentrylongpl{\glslabel}}%
              {\glsentrylong{\glslabel}}%
           }%
        }%
        {}%
      }%
    }%
    {\glscustomtext\glsinsert}%
  }%
}
```

teNewAcronymDef

```
\newcommand*{\FootnoteNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      sort={\the\glsshorttok},%
      text={\the\glsshorttok},%
      plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      first={\the\glsshorttok},%
      firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\the\glslongtok},%
      descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@descplural\gls@assign@descplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@descplural##1##2{%
    \@@gls@expand@field{##1}{descplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@descplural\@org@gls@assign@descplural
}
```

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.

```
\newcommand*{\SetFootnoteAcronymStyle}{%
  \renewcommand{\newacronym}[4][]{%
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetFootnoteAcronymDisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \FootnoteNewAcronymDef
  }%
```

Set display

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetFootnoteAcronymDisplayStyle{\@gls@type}%
  }%
```

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.

```
  \ifglsacrsmallcaps
```

```
    \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
    \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
  \else
    \ifglsacrsmaller
      \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
    \fi
  \fi
```

Check for option clash

```
  \ifglsacrdua
    \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
    can’t both be set}{}%
  \fi
}%
```

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.

```
\DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
  \protected@edef\gls@tmp{#1}%
  \ifdefempty\gls@tmp
  {}%
  {%
    \ifx\gls@tmp\@gls@default@value
    \else
      \space (#2{#1})%
    \fi
  }%
}
```

nymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
\newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
  \defglsentryfmt[#1]{%

    \ifdefempty\glscustomtext
    {%
```

Move the inserted text outside of `\acronymfont`

```
      \let\gls@org@insert\glsinsert
      \let\glsinsert\@empty
      \ifglsused{\glslabel}%
      {%
        \acronymfont{\glsgenentryfmt}\gls@org@insert
      }%
      {%
        \glsgenentryfmt
        \ifglshassymbol{\glslabel}%
        {%
          \glsifplural
          {%
```

```
            \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
          }%
          {%
            \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
          }%
          \space
            (\glscapscase
            {\firstacronymfont{\@glo@symbol}}%
            {\firstacronymfont{\@glo@symbol}}%
            {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
        }%
        {}%
      }%
    }%
    {\glscustomtext\glsinsert}%
  }%
}
```

llNewAcronymDef

```
\newcommand*{\SmallNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      sort={\the\glsshorttok},%
      text={\the\glsshorttok},%
```

Default to the short plural.

```
      plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      first={\the\glslongtok},%
```

Default to the long plural.

```
      firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\noexpand\@glo@first},%
      descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      symbol={\the\glsshorttok},%
```

Default to the short plural.

```
      symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@descplural\gls@assign@descplural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@descplural##1##2{%
    \@@gls@expand@field{##1}{descplural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@descplural\@org@gls@assign@descplural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

`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.

```
\newcommand*{\SetSmallAcronymStyle}{%
  \renewcommand{\newacronym}[4][]{%
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetSmallAcronymDisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \SmallNewAcronymDef
  }%
```

Change the display since first only contains long form.

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetSmallAcronymDisplayStyle{\@gls@type}%
  }%
```

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.

```
  \ifglsacrsmallcaps
    \renewcommand*{\acronymfont}[1]{\textsc{##1}}
    \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
  \else
    \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
  \fi
```

check for option clash

```
  \ifglsacrdua
    \ifglsacrsmallcaps
      \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
      can’t both be set}{}%
    \else
      \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
      can’t both be set}{}%
    \fi
  \fi
}%
```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```
\newcommand*{\SetDUADisplayStyle}[1]{%
  \defglsentryfmt[#1]{\glsgenentryfmt}%
}
```

UANewAcronymDef

```
\newcommand*{\DUANewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\the\glsshorttok},%
      text={\the\glslongtok},%
      first={\the\glslongtok},%
      plural={\noexpand\expandonce\noexpand\@glo@longpl},%
      firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\the\glslongtok},%
      descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
      symbol={\the\glsshorttok},%
      symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \let\@org@gls@assign@descplural\gls@assign@descplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \def\gls@assign@descplural##1##2{%
    \@@gls@expand@field{##1}{descplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
  \let\gls@assign@descplural\@org@gls@assign@descplural
}
```

`\SetDUAStyle` Always expand acronyms.

```
\newcommand*{\SetDUAStyle}{%
  \renewcommand{\newacronym}[4][]{%
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetDUADisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \DUANewAcronymDef
  }%
```

Set the display

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetDUADisplayStyle{\@gls@type}%
  }%
}
```

`SetAcronymStyle`

```
\newcommand*{\SetAcronymStyle}{%
  \SetDefaultAcronymStyle
  \ifglsacrdescription
    \ifglsacrfootnote
      \SetDescriptionFootnoteAcronymStyle
    \else
      \ifglsacrdua
        \SetDescriptionDUAAcronymStyle
      \else
        \SetDescriptionAcronymStyle
      \fi
    \fi
  \else
    \ifglsacrfootnote
      \SetFootnoteAcronymStyle
    \else
      \ifthenelse{\boolean{glsacrsmallcaps}\OR
        \boolean{glsacrsmaller}}%
      {%
        \SetSmallAcronymStyle
      }%
      {%
        \ifglsacrdua
          \SetDUAStyle
        \fi
      }%
    \fi
  \fi
}
```

Set the acronym style according to the package options

```
\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.

`tomDisplayStyle` Sets the acronym display style.

```
\newcommand*{\SetCustomDisplayStyle}[1]{%
  \defglsentryfmt[#1]{\glsgenentryfmt}%
}
```

`omAcronymFields`

```
\newcommand*{\CustomAcronymFields}{%
  name={\the\glsshorttok},%
  description={\the\glslongtok},%
  first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
  firstplural={\acrfullformat
    {\noexpand\glsentrylongpl{\the\glslabeltok}}%
    {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

  text={\the\glsshorttok},%
  plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
}
```

`omNewAcronymDef`

```
\newcommand*{\CustomNewAcronymDef}{%
  \protected@edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
```

```
      type=\acronymtype,%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      user1={\the\glsshorttok},%
      user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
      user3={\the\glslongtok},%
      user4={\the\glslongtok\noexpand\acrpluralsuffix},%
      \CustomAcronymFields,%
      \the\glskeylisttok
    }%
  }%
  \@do@newglossaryentry
}
```

\SetCustomStyle

```
\newcommand*{\SetCustomStyle}{%
  \renewcommand{\newacronym}[4][]{%
    \ifx\@glsacronymlists\@empty
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
      \SetCustomDisplayStyle{\@glo@type}%
    \fi
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \CustomNewAcronymDef
  }%
```

Set the display

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetCustomDisplayStyle{\@gls@type}%
  }%
}
```

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.

```
\RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
\@gls@loadlist
```

The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
\@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.

```
\@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
\@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.

```
\ifx\@glossary@default@style\relax
\else
  \setglossarystyle{\@glossary@default@style}
\fi
```

1.20 Debugging Commands

\showgloparent

```
\showgloparent{⟨label⟩}
```

```
\newcommand*{\showgloparent}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
}
```

\showglolevel

```
\showglolevel{⟨label⟩}
```

```
\newcommand*{\showglolevel}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
}
```

\showglotext

```
\showglotext{⟨label⟩}
```

```
\newcommand*{\showglotext}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
}
```

\showgloplural

```
\showgloplural{⟨label⟩}
```

```
\newcommand*{\showgloplural}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
}
```

\showglofirst

```
\showglofirst{⟨label⟩}
```

```
\newcommand*{\showglofirst}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@first\endcsname
}
```

\showglofirstpl

```
\showglofirstpl{⟨label⟩}
```

```
\newcommand*{\showglofirstpl}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpl\endcsname
}
```

\showglotype

```
\showglotype{⟨label⟩}
```

```
\newcommand*{\showglotype}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@type\endcsname
}
```

\showglocounter

```
\showglocounter{⟨label⟩}
```

```
\newcommand*{\showglocounter}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@counter\endcsname
}
```

\showglouseri

```
\showglouseri{⟨label⟩}
```

```
\newcommand*{\showglouseri}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@useri\endcsname
}
```

\showglouserii

```
\showglouserii{⟨label⟩}
```

```
\newcommand*{\showglouserii}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname
}
```

\showglouseriii

```
\showglouseriii{⟨label⟩}
```

```
\newcommand*{\showglouseriii}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@useriii\endcsname
}
```

\showglouseriv

```
\showglouseriv{⟨label⟩}
```

```
\newcommand*{\showglouseriv}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@useriv\endcsname
}
```

\showglouserv

```
\showglouserv{⟨label⟩}
```

```
\newcommand*{\showglouserv}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@userv\endcsname
}
```

\showglouservi

```
\showglouservi{⟨label⟩}
```

```
\newcommand*{\showglouservi}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
}
```

\showgloname

```
\showgloname{⟨label⟩}
```

```
\newcommand*{\showgloname}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@name\endcsname
}
```

\showglodesc

```
\showglodesc{⟨label⟩}
```

```
\newcommand*{\showglodesc}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
}
```

howglodescplural

\showglodescplural{⟨*label*⟩}

```
\newcommand*{\showglodescplural}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
}
```

\showglosort

\showglosort{⟨*label*⟩}

```
\newcommand*{\showglosort}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
}
```

\showglosymbol

\showglosymbol{⟨*label*⟩}

```
\newcommand*{\showglosymbol}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
}
```

wglosymbolplural

\showglosymbolplural{⟨*label*⟩}

```
\newcommand*{\showglosymbolplural}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
}
```

\showgloshort

\showgloshort{⟨*label*⟩}

```
\newcommand*{\showgloshort}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
}
```

\showglolong

\showglolong{⟨*label*⟩}

```
\newcommand*{\showglolong}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
}
```

\showgloindex

```
\showgloindex{⟨label⟩}
```

```
\newcommand*{\showgloindex}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
}
```

\showgloflag

```
\showgloflag{⟨label⟩}
```

```
\newcommand*{\showgloflag}[1]{%
  \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
}
```

\showgloloclist

```
\showgloloclist{⟨label⟩}
```

```
\newcommand*{\showgloloclist}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
}
```

\showglofield

```
\showglofield{⟨label⟩}{⟨field⟩}
```

```
\newcommand*{\showglofield}[2]{%
 \csshow{glo@\glsdetoklabel{#1}@#2}%
}
```

showacronymlists

```
\showacronymlists
```

Show list of glossaries that have been flagged as a list of acronyms.

```
\newcommand*{\showacronymlists}{%
   \show\@glsacronymlists
}
```

\showglossaries

```
\showglossaries
```

Show list of defined glossaries.

```
\newcommand*{\showglossaries}{%
```

```
  \show\@glo@types
}
```

\showglossaryin

```
\showglossaryin{⟨glossary-label⟩}
```

Show the 'in' extension for the given glossary.

```
\newcommand*{\showglossaryin}[1]{%
  \expandafter\show\csname @glotype@#1@in\endcsname
}
```

\showglossaryout

```
\showglossaryout{⟨glossary-label⟩}
```

Show the 'out' extension for the given glossary.

```
\newcommand*{\showglossaryout}[1]{%
  \expandafter\show\csname @glotype@#1@out\endcsname
}
```

howglossarytitle

```
\showglossarytitle{⟨glossary-label⟩}
```

Show the title for the given glossary.

```
\newcommand*{\showglossarytitle}[1]{%
  \expandafter\show\csname @glotype@#1@title\endcsname
}
```

wglossarycounter

```
\showglossarycounter{⟨glossary-label⟩}
```

Show the counter for the given glossary.

```
\newcommand*{\showglossarycounter}[1]{%
  \expandafter\show\csname @glotype@#1@counter\endcsname
}
```

wglossaryentries

```
\showglossaryentries{⟨glossary-label⟩}
```

Show the list of entry labels for the given glossary.

```
\newcommand*{\showglossaryentries}[1]{%
  \expandafter\show\csname glolist@#1\endcsname
}
```

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.

```
\csname ifglscompatible-2.07\endcsname
  \RequirePackage{glossaries-compatible-207}
\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.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-prefix}[2018/06/01 v4.40 (NLCT)]
```

Pass all options to glossaries:

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
\ProcessOptions
```

Load glossaries:

```
\RequirePackage{glossaries}
```

Add the new keys:

```
\define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
\define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
\define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
\define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
\appto\@gls@keymap{,%
   {prefixfirst}{prefixfirst},%
   {prefixfirstplural}{prefixfirstplural},%
   {prefix}{prefix},%
   {prefixplural}{prefixplural}%
 }
```

Set the default values:

```
\appto\@newglossaryentryprehook{%
  \def\@glo@entryprefix{}%
  \def\@glo@entryprefixplural{}%
  \let\@glo@entryprefixfirst\@gls@default@value
  \let\@glo@entryprefixfirstplural\@gls@default@value
}
```

Set the assignment code:

```
\appto\@newglossaryentryposthook{%
  \gls@assign@field{}{\@glo@label}{prefix}{\@glo@entryprefix}%
  \gls@assign@field{}{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%
```

If prefixfirst has not been supplied, make it the same as prefix.

```
  \expandafter\gls@assign@field\expandafter
    {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
    {\@glo@entryprefixfirst}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```
  \expandafter\gls@assign@field\expandafter
    {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
    {prefixfirstplural}{\@glo@entryprefixfirstplural}%
}
```

Define commands to access these fields:

ntryprefixfirst

```
\newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}
```

efixfirstplural

```
\newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}
```

\glsentryprefix

```
\newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}
```

tryprefixplural

```
\newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}
```

Now for the initial upper case variants:

ntryprefixfirst

```
\newrobustcmd*{\Glsentryprefixfirst}[1]{%
  \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
  \xmakefirstuc\@glo@text
}
```

efixfirstplural

```
\newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
  \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
  \xmakefirstuc\@glo@text
}
```

\Glsentryprefix

```
\newrobustcmd*{\Glsentryprefix}[1]{%
  \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
  \xmakefirstuc\@glo@text
}
```

tryprefixplural

```
\newrobustcmd*{\Glsentryprefixplural}[1]{%
  \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
  \xmakefirstuc\@glo@text
}
```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
\newcommand*{\ifglshasprefix}[3]{%
  \ifcsempty{glo@#1@prefix}%
  {#3}%
  {#2}%
}
```

hasprefixplural

```
\newcommand*{\ifglshasprefixplural}[3]{%
  \ifcsempty{glo@#1@prefixplural}%
  {#3}%
  {#2}%
}
```

shasprefixfirst

```
\newcommand*{\ifglshasprefixfirst}[3]{%
  \ifcsempty{glo@#1@prefixfirst}%
  {#3}%
  {#2}%
}
```

efixfirstplural

```
\newcommand*{\ifglshasprefixfirstplural}[3]{%
  \ifcsempty{glo@#1@prefixfirstplural}%
  {#3}%
  {#2}%
}
```

Define commands that insert the prefix before commands like `\gls`:

\pgls

```
\newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
\newcommand*{\@pgls}[2][]{%
  \new@ifnextchar[%
  {\@pgls@{#1}{#2}}%
  {\@pgls@{#1}{#2}[]}%
}
```

\@pgls@ Read in the final optional argument:

```
\def\@pgls@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \ifglsused{#2}%
    {%
      \glsentryprefix{#2}%
    }%
    {%
      \glsentryprefixfirst{#2}%
    }%
    \@gls@{#1}{#2}[#3]%
  }%
}
```

Similarly for the plural version:

\pglspl

```
\newrobustcmd{\pglspl}{\@gls@hyp@opt\@pglspl}
```

\@pglspl Unstarred version.

```
\newcommand*{\@pglspl}[2][]{%
  \new@ifnextchar[%
  {\@pglspl@{#1}{#2}}%
  {\@pglspl@{#1}{#2}[]}%
}
```

\@pglspl@ Read in the final optional argument:

```
\def\@pglspl@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \ifglsused{#2}%
    {%
      \glsentryprefixplural{#2}%
    }%
    {%
      \glsentryprefixfirstplural{#2}%
    }%
    \@glspl@{#1}{#2}[#3]%
  }%
}
```

Now for the first letter upper case versions:

\Pgls

```
\newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}
```

\@Pgls Unstarred version.

```
\newcommand*{\@Pgls}[2][]{%
  \new@ifnextchar[%
  {\@Pgls@{#1}{#2}}%
  {\@Pgls@{#1}{#2}[]}%
}
```

\@Pgls@ Read in the final optional argument:

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

```
  \glsdoifexists{#2}%
  {%
    \ifglsused{#2}%
    {%
      \ifglshasprefix{#2}%
      {%
        \Glsentryprefix{#2}%
        \@gls@{#1}{#2}[#3]%
      }%
      {\@Gls@{#1}{#2}[#3]}%
    }%
    {%
      \ifglshasprefixfirst{#2}%
      {%
        \Glsentryprefixfirst{#2}%
        \@gls@{#1}{#2}[#3]%
      }%
      {\@Gls@{#1}{#2}[#3]}%
    }%
  }%
}
```

Similarly for the plural version:

\Pglspl

```
\newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}
```

\@Pglspl Unstarred version.

```
\newcommand*{\@Pglspl}[2][]{%
  \new@ifnextchar[%
  {\@Pglspl@{#1}{#2}}%
  {\@Pglspl@{#1}{#2}[]}%
}
```

\@Pglspl@ Read in the final optional argument:

```
\def\@Pglspl@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \ifglsused{#2}%
    {%
      \ifglshasprefixplural{#2}%
      {%
        \Glsentryprefixplural{#2}%
        \@glspl@{#1}{#2}[#3]%
      }%
      {\@Glspl@{#1}{#2}[#3]}%
    }%
    {%
      \ifglshasprefixfirstplural{#2}%
```

```
        {%
          \Glsentryprefixfirstplural{#2}%
          \@glspl@{#1}{#2}[#3]%
        }%
        {\@Glspl@{#1}{#2}[#3]}%
      }%
  }%
}
```

Finally the all upper case versions:

\PGLS

```
\newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}
```

\@PGLS Unstarred version.

```
\newcommand*{\@PGLS}[2][]{%
  \new@ifnextchar[%
  {\@PGLS@{#1}{#2}}%
  {\@PGLS@{#1}{#2}[]}%
}
```

\@PGLS@ Read in the final optional argument:

```
\def\@PGLS@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \ifglsused{#2}%
    {%
      \mfirstucMakeUppercase{\glsentryprefix{#2}}%
    }%
    {%
      \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
    }%
    \@GLS@{#1}{#2}[#3]%
  }%
}
```

Plural version:

\PGLSpl

```
\newrobustcmd{\PGLSpl}{\@gls@hyp@opt\@PGLSpl}
```

\@PGLSpl Unstarred version.

```
\newcommand*{\@PGLSpl}[2][]{%
  \new@ifnextchar[%
  {\@PGLSpl@{#1}{#2}}%
  {\@PGLSpl@{#1}{#2}[]}%
}
```

\@PGLSpl@ Read in the final optional argument:

```
\def\@PGLSpl@#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \ifglsused{#2}%
    {%
      \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
    }%
    {%
      \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
    }%
    \@GLSpl@{#1}{#2}[#3]%
  }%
}
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
\ProvidesPackage{glossary-hypernav}[2018/06/01 v4.40 (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

```
\newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
  \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
  \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

avhyperlinkname Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
\newcommand*{\glsnavhyperlinkname}[2]{glsn:#1@#2}
```

`\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.

snavhypertarget

```
\newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  \@glsnavhypertarget{#1}{#2}{#3}%
}
```

The actual code is now in an internal command that doesn't have an optional argument, which makes it easier to save and restore the original behaviour.

snavhypertarget

```
\newcommand*{\@glsnavhypertarget}[3]{%
```

Add this group to the aux file for re-run check.

```
  \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
  \@glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
  \expandafter\let
     \expandafter\@gls@list\csname @gls@hypergrouplist@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
  \@for\@gls@elem:=\@gls@list\do{%
    \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
  \if@endfor
  \else
```

This group was not included in the list, so issue a warning.

```
    \GlossariesWarningNoLine{Navigation panel
       for glossary type '#1'^^Jmissing group '#2'}%
    \gdef\gls@hypergrouprerun{%
      \GlossariesWarningNoLine{Navigation panel
      has changed. Rerun LaTeX}}%
  \fi
}
```

hypergrouprerun Give a warning at the end if re-run required

```
\let\gls@hypergrouprerun\relax
\AtEndDocument{\gls@hypergrouprerun}
```

@gls@hypergroup This adds to (or creates) the command `\@gls@hypergrouplist@`⟨*glossary type*⟩ which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
\newcommand*{\@gls@hypergroup}[2]{%
\@ifundefined{@gls@hypergrouplist@#1}{%
   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
}{%
   \expandafter\let\expandafter\@gls@tmp
      \csname @gls@hypergrouplist@#1\endcsname
   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
      \@gls@tmp,#2}%
}%
}
```

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. (In earlier verions 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. Version 1.14 changed this to only use labels for groups that are present.) Now for the whole navigation bit:

\glsnavigation

```
\newcommand*{\glsnavigation}{%
  \def\@gls@between{}%
  \ifcsundef{@gls@hypergrouplist@\@glo@type}%
  {%
    \def\@gls@list{}%
  }%
  {%
    \expandafter\let\expandafter\@gls@list
      \csname @gls@hypergrouplist@\@glo@type\endcsname
  }%
  \@for\@gls@tmp:=\@gls@list\do{%
    \@gls@between

    \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
    \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
    \let\@gls@between\glshypernavsep
  }%
}
```

\glshypernavsep Separator for the hyper navigation bar.

```
\newcommand*{\glshypernavsep}{\space\textbar\space}
```

The \glssymbolnav produces a simple navigation set of links for just the symbol and number groups. This used to be used at the start of \glsnavigation. This command is no longer needed.

\glssymbolnav

```
\newcommand*{\glssymbolnav}{%
 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
 \glshypernavsep
 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
 \glshypernavsep
}
```

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.

```
\ProvidesPackage{glossary-inline}[2018/06/01 v4.40 (NLCT)]
```

inline Define the inline style.

```
\newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by \glossentry)

```
  \renewenvironment{theglossary}%
    {%
```

```
      \def\gls@inlinesep{}%
      \def\gls@inlinesubsep{}%
      \def\gls@inlinepostchild{}%
    }%
    {\glspostinline}%
```

No header:

```
\renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

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

Just display separator followed by name and description:

```
  \renewcommand{\glossentry}[2]{%
    \glsinlinedopostchild
    \gls@inlinesep
    \glsentryitem{##1}%
    \glsinlinenameformat{##1}{%
      \glossentryname{##1}%
    }%
   \ifglsdescsuppressed{##1}%
   {%
      \glsinlineemptydescformat
      {%
         \glossentrysymbol{##1}%
      }%
      {%
        ##2%
      }%
    }%
    {%
      \ifglshasdesc{##1}%
      {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
      {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
    }%
    \ifglshaschildren{##1}%
    {%
       \glsresetsubentrycounter
       \glsinlineparentchildseparator
       \def\gls@inlinesubsep{}%
       \def\gls@inlinepostchild{\glsinlinepostchild}%
    }%
    {}%
    \def\gls@inlinesep{\glsinlineseparator}%
  }%
```

Sub-entries display description:

```
  \renewcommand{\subglossentry}[3]{%
    \gls@inlinesubsep%
    \glsinlinesubnameformat{##2}{%
```

```
        \glossentryname{##2}}%
    \glssubentryitem{##2}%
    \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
    \def\gls@inlinesubsep{\glsinlinesubseparator}%
  }%
```

Nothing special between groups:

```
  \renewcommand*{\glsgroupskip}{}%
}
```

linedopostchild

```
\newcommand*{\glsinlinedopostchild}{%
    \gls@inlinepostchild
    \def\gls@inlinepostchild{}%
}
```

inlineseparator Separator to use between entries.

```
\newcommand*{\glsinlineseparator}{;\space}
```

inesubseparator Separator to use between sub-entries.

```
\newcommand*{\glsinlinesubseparator}{,\space}
```

tchildseparator Separator to use between parent and children.

```
\newcommand*{\glsinlineparentchildseparator}{:\space}
```

inlinepostchild Hook to use between child and next entry

```
\newcommand*{\glsinlinepostchild}{}
```

\glspostinline Terminator for inline glossary.

```
\newcommand*{\glspostinline}{\glspostdescription\space}
```

nlinenameformat Formats the name of the entry (first argument label, second argument name):

```
\newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

nlinedescformat Formats the entry's description, symbol and location list:

```
\newcommand*{\glsinlinedescformat}[3]{\space#1}
```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```
\newcommand*{\glsinlineemptydescformat}[2]{}
```

nesubnameformat Formats the name of the subentry (first argument label, second argument name):

```
\newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

nesubdescformat Formats the subentry's description, symbol and location list:

```
\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.

```
\ProvidesPackage{glossary-list}[2018/06/01 v4.40 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
\providecommand{\indexspace}{%
  \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
}
```

`tgroupheaderfmt` Provide a way of adjusting the format of the group headings.

```
\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.

```
\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.

```
\newglossarystyle{list}{%
```

Use description environment:

```
  \renewenvironment{theglossary}%
    {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

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

Main (level 0) entries start a new item in the list:

```
  \renewcommand*{\glossentry}[2]{%
    \item[\glsentryitem{##1}%
          \glstarget{##1}{\glossentryname{##1}}]
       \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
  \renewcommand*{\subglossentry}[3]{%
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}\space
    \glossentrydesc{##2}\glspostdescription\space ##3.}%
```

Add vertical space between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
}
```

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

```
\newglossarystyle{listgroup}{%
```

Base it on the list style:

```
  \setglossarystyle{list}%
```

Each group has a heading:

```
  \renewcommand*{\glsgroupheading}[1]{%
    \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.

```
\newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
  \setglossarystyle{list}%
```

Add navigation links at the start of the environment.

```
  \renewcommand*{\glossaryheader}{%
    \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a hypertarget:

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

`altlist` The altlist glossary style is like the list style, but places the description on a new line. Subentries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
\newglossarystyle{altlist}{%
```

Base it on the list style:

```
  \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
  \renewcommand*{\glossentry}[2]{%
    \item[\glsentryitem{##1}%
      \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.

```
      \mbox{}\par\nobreak\@afterheading
      \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
  \renewcommand{\subglossentry}[3]{%
    \par
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
}
```

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

```
\newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
  \setglossarystyle{altlist}%
```

Each group has a heading:

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

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

```
\newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
  \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment.

```
  \renewcommand*{\glossaryheader}{%
    \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a hypertarget:

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

```
\newglossarystyle{listdotted}{%
```

Base it on the list style:

```
  \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
  \renewcommand*{\glossentry}[2]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
      \glsentryitem{##1}%
      \glstarget{##1}{\glossentryname{##1}}%
      \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
\renewcommand*{\subglossentry}[3]{%
  \item[]\makebox[\glslistdottedwidth][l]{%
  \glssubentryitem{##2}%
  \glstarget{##2}{\glossentryname{##2}}%
  \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
}
```

listdottedwidth

```
\newlength\glslistdottedwidth
\setlength{\glslistdottedwidth}{.5\hsize}
```

sublistdotted This style is similar to the glostylelistdotted style, except that the main entries just have the name displayed.

```
\newglossarystyle{sublistdotted}{%
```

Base it on the listdotted style:

```
  \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
  \renewcommand*{\glossentry}[2]{%
    \item[\glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
}
```

3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the longtable environment in the glossary.

```
\ProvidesPackage{glossary-long}[2018/06/01 v4.40 (NLCT)]
```

Requires the package:

```
\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`.)

```
\@ifundefined{glsdescwidth}{%
  \newlength\glsdescwidth
  \setlength{\glsdescwidth}{0.6\hsize}
}{}
```

lspagelistwidth This is a length that governs the width of the page list column.

```
\@ifundefined{glspagelistwidth}{%
  \newlength\glspagelistwidth
  \setlength{\glspagelistwidth}{0.1\hsize}
}{}
```

`long` The long glossary style command which uses the longtable environment:

```
\newglossarystyle{long}{%
```

Use longtable with two columns:

```
  \renewenvironment{theglossary}%
     {\begin{longtable}{lp{\glsdescwidth}}}%
     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
  \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

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

Main (level 0) entries displayed in a row:

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
  }%
```

Sub entries displayed on the following row without the name:

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
     ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
  \fi
}
```

`longborder` The longborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
  \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
  \renewenvironment{theglossary}{%
    \begin{longtable}{|l|p{\glsdescwidth}|}}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
  \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

`longheader` The longheader style is like the long style but with a header:

```
\newglossarystyle{longheader}{%
```

Base it on the glostylelong style:

```
\setglossarystyle{long}%
```

Set the table's header:

```
\renewcommand*{\glossaryheader}{%
  \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
}
```

ongheaderborder The longheaderborder style is like the long style but with a header and border:

```
\newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
\setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
\renewcommand*{\glossaryheader}{%
  \hline\bfseries \entryname & \bfseries
  \descriptionname\tabularnewline\hline
  \endhead
  \hline\endfoot}%
}
```

long3col The long3col style is like long but with 3 columns

```
\newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
\renewenvironment{theglossary}%
  {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
  {\end{longtable}}%
```

No table header:

```
\renewcommand*{\glossaryheader}{}%
```

No headings between groups:

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

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
\renewcommand{\glossentry}[2]{%
  \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
  \glossentrydesc{##1} & ##2\tabularnewline
}%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
\renewcommand{\subglossentry}[3]{%
   &
   \glssubentryitem{##2}%
   \glstarget{##2}{\strut}\glossentrydesc{##2} &
   ##3\tabularnewline
}%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
  \fi
}
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
\newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
  \setglossarystyle{long3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}%
    {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
  \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

long3colheader The long3colheader style is like long3col but with a header row:

```
\newglossarystyle{long3colheader}{%
```

Base it on the glostylelong3col style:

```
  \setglossarystyle{long3col}%
```

Set the table's header:

```
  \renewcommand*{\glossaryheader}{%
    \bfseries\entryname&\bfseries\descriptionname&
    \bfseries\pagelistname\tabularnewline\endhead}%
}
```

colheaderborder The long3colheaderborder style is like the above but with a border

```
\newglossarystyle{long3colheaderborder}{%
```

Base it on the glostylelong3colborder style:

```
  \setglossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
  \renewcommand*{\glossaryheader}{%
    \hline
    \bfseries\entryname&\bfseries\descriptionname&
    \bfseries\pagelistname\tabularnewline\hline\endhead
    \hline\endfoot}%
}
```

long4col The long4col style has four columns where the third column contains the value of the associated symbol key.

```
\newglossarystyle{long4col}{%
```

Use a longtable with 4 columns:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{llll}}%
    {\end{longtable}}%
```

No table header:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
  \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):

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1} &
    \glossentrysymbol{##1} &
    ##2\tabularnewline
  }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2} &
     \glossentrysymbol{##2} & ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
  \fi
}
```

long4colheader The long4colheader style is like long4col but with a header row.

```
\newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
  \setglossarystyle{long4col}%
```

Table has a header:

```
  \renewcommand*{\glossaryheader}{%
    \bfseries\entryname&\bfseries\descriptionname&
    \bfseries \symbolname&
    \bfseries\pagelistname\tabularnewline\endhead}%
}
```

long4colborder The long4colborder style is like long4col but with a border.

```
\newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
  \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{|l|l|l|l|}}%
    {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
  \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

colheaderborder The long4colheaderborder style is like the above but with a border.

```
\newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```
  \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{|l|l|l|l|}}%
    {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
  \renewcommand*{\glossaryheader}{%
    \hline\bfseries\entryname&\bfseries\descriptionname&
    \bfseries \symbolname&
    \bfseries\pagelistname\tabularnewline\hline\endhead
    \hline\endfoot}%
}
```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```
\newglossarystyle{altlong4col}{%
```

Base it on the glostylelong4col style:

```
   \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
    {\end{longtable}}%
}
```

tlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
\newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
  \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
    {\end{longtable}}%
}
```

tlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
\newglossarystyle{altlong4colborder}{%
```

Base it on the glostylelong4colborder style:

```
   \setglossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
    {\end{longtable}}%
}
```

colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
\newglossarystyle{altlong4colheaderborder}{%
```

Base it on the glostylelong4colheaderborder style:

```
  \setglossarystyle{long4colheaderborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
    {\end{longtable}}%
}
```

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

```
\ProvidesPackage{glossary-longbooktabs}[2018/06/01 v4.40 (NLCT)]
```

Requires booktabs package:

```
\RequirePackage{booktabs}
```

and the base packages for long styles:

```
\RequirePackage{glossary-long}
\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.

```
\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.

```
  \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
  \setglossarystyle{long}%
```

Add a header with rules.

```
  \renewcommand*{\glossaryheader}{%
    \toprule \bfseries \entryname & \bfseries
   \descriptionname\tabularnewline\midrule\endhead
    \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.

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

`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.

```
\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.

```
  \glspatchLToutput
```

Use the long3col style as a base.

```
  \setglossarystyle{long3col}%
```

Add a header with rules.

```
  \renewcommand*{\glossaryheader}{%
    \toprule \bfseries \entryname &
    \bfseries \descriptionname &
    \bfseries \pagelistname
    \tabularnewline\midrule\endhead
    \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.

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

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.

```
\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.

```
  \glspatchLToutput
```

Use the long4col style as a base.

```
  \setglossarystyle{long4col}%
```

Add a header with rules.

```
  \renewcommand*{\glossaryheader}{%
    \toprule \bfseries \entryname &
    \bfseries \descriptionname &
    \bfseries \symbolname &
    \bfseries \pagelistname
    \tabularnewline\midrule\endhead
    \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.

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

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.

```
\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.

```
  \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
  \setglossarystyle{long4col-booktabs}%
```

Change the column specifications:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
    {\end{longtable}}%
}
```

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.

```
\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.

```
  \glspatchLToutput
```

Use the long-booktabs style as a base.

```
  \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
  \renewenvironment{theglossary}%
     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
     {\end{longtable}}%
}
```

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.

```
\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.

```
  \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
  \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
       >{\raggedright}p{\glspagelistwidth}}}%
    {\end{longtable}}%
}
```

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.

```
\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.

```
  \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
\setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
       >{\raggedright}p{\glspagelistwidth}}}%
    {\end{longtable}}%
}
```

sLTpenaltycheck

```
\newcommand*{\glsLTpenaltycheck}{%
 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
}
```

enaltygroupskip

```
\newcommand{\glspenaltygroupskip}{%
  \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring `\LT@output` for the user.

```
\let\@gls@org@LT@output\LT@output
\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

```
\newcommand*{\glspatchLToutput}{%
 \renewcommand*{\LT@output}{%
   \ifnum\outputpenalty <-\@Mi
     \ifnum\outputpenalty > -\LT@end@pen
       \LT@err{floats and marginpars not allowed in a longtable}\@ehc
     \else
       \setbox\z@\vbox{\unvbox\@cclv}%
       \ifdim \ht\LT@lastfoot>\ht\LT@foot
         \dimen@\pagegoal
         \advance\dimen@-\ht\LT@lastfoot
         \ifdim\dimen@<\ht\z@
           \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
           \@makecol
           \@outputpage
           \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%
         \fi
       \fi
       \global\@colroom\@colht
       \global\vsize\@colht
       {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
     \fi
   \else
```

```
      \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
      \@makecol
      \@outputpage
      \global\vsize\@colroom
      \copy\LT@head
      \glsLTpenaltycheck
      \nobreak
    \fi
  }%
}
```

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.

```
\ProvidesPackage{glossary-longragged}[2018/06/01 v4.40 (NLCT)]
```

Requires the package:

```
\RequirePackage{array}
```

Requires the package:

```
\RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
\@ifundefined{glsdescwidth}{%
  \newlength\glsdescwidth
  \setlength{\glsdescwidth}{0.6\hsize}
}{}
```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
\@ifundefined{glspagelistwidth}{%
  \newlength\glspagelistwidth
  \setlength{\glspagelistwidth}{0.1\hsize}
}{}
```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
\newglossarystyle{longragged}{%
```

Use longtable with two columns:

```
  \renewenvironment{theglossary}%
     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
  \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

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

Main (level 0) entries displayed in a row:

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1}\glspostdescription\space ##2%
    \tabularnewline
  }%
```

Sub entries displayed on the following row without the name:

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2}%
     \glspostdescription\space ##3%
     \tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
  \fi
}
```

ongraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{longraggedborder}{%
```

Base it on the glostylelongragged style:

```
  \setglossarystyle{longragged}%
```

Use longtable with two columns with vertical lines between each column:

```
  \renewenvironment{theglossary}{%
    \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}}%
    {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
  \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

ongraggedheader The longraggedheader style is like the longragged style but with a header:

```
\newglossarystyle{longraggedheader}{%
```

Base it on the glostylelongragged style:

```
  \setglossarystyle{longragged}%
```

Set the table's header:

```
  \renewcommand*{\glossaryheader}{%
    \bfseries \entryname & \bfseries \descriptionname
    \tabularnewline\endhead}%
}
```

gedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
\newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```
  \setglossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```
  \renewcommand*{\glossaryheader}{%
    \hline\bfseries \entryname & \bfseries \descriptionname
    \tabularnewline\hline
    \endhead
    \hline\endfoot}%
}
```

longragged3col The longragged3col style is like longragged but with 3 columns

```
\newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
       >{\raggedright}p{\glspagelistwidth}}}%
    {\end{longtable}}%
```

No table header:

```
  \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

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

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1} & ##2\tabularnewline
  }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2} &
     ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
  \fi
}
```

agged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
\newglossarystyle{longragged3colborder}{%
```

Base it on the glostylelongragged3col style:

```
  \setglossarystyle{longragged3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
  \renewenvironment{theglossary}%
    {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|%
      >{\raggedright}p{\glspagelistwidth}|}}%
    {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
  \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

agged3colheader The longragged3colheader style is like longragged3col but with a header row:

```
\newglossarystyle{longragged3colheader}{%
```

Base it on the glostylelongragged3col style:

```
  \setglossarystyle{longragged3col}%
```

Set the table's header:

```
  \renewcommand*{\glossaryheader}{%
    \bfseries\entryname&\bfseries\descriptionname&
    \bfseries\pagelistname\tabularnewline\endhead}%
}
```

colheaderborder The longragged3colheaderborder style is like the above but with a border

```
\newglossarystyle{longragged3colheaderborder}{%
```

Base it on the glostylelongragged3colborder style:

```
  \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
  \renewcommand*{\glossaryheader}{%
    \hline
    \bfseries\entryname&\bfseries\descriptionname&
    \bfseries\pagelistname\tabularnewline\hline\endhead
    \hline\endfoot}%
}
```

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.

```
\newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
\renewenvironment{theglossary}%
  {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
     >{\raggedright}p{\glspagelistwidth}}}%
  {\end{longtable}}%
```

No table header:

```
\renewcommand*{\glossaryheader}{}%
```

No group headings:

```
\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):

```
\renewcommand{\glossentry}[2]{%
  \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
  \glossentrydesc{##1} & \glossentrysymbol{##1} &
  ##2\tabularnewline
}%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
\renewcommand{\subglossentry}[3]{%
   &
   \glssubentryitem{##2}%
   \glstarget{##2}{\strut}\glossentrydesc{##2} &
   \glossentrysymbol{##2} & ##3\tabularnewline
}%
```

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)

```
\ifglsnogroupskip
  \renewcommand*{\glsgroupskip}{}%
\else
  \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
\fi
}
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
\newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
\setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
\renewenvironment{theglossary}%
  {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
    >{\raggedright}p{\glspagelistwidth}}}%
  {\end{longtable}}%
```

Table has a header:

```
\renewcommand*{\glossaryheader}{%
  \bfseries\entryname&\bfseries\descriptionname&
  \bfseries \symbolname&
  \bfseries\pagelistname\tabularnewline\endhead}%
}
```

agged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
\newglossarystyle{altlongragged4colborder}{%
```

Base it on the glostylealtlongragged4col style:

```
 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
\renewenvironment{theglossary}%
  {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
    >{\raggedright}p{\glspagelistwidth}|}}%
  {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
\renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

colheaderborder The altlongragged4colheaderborder style is like the above but with a header as well as a border.

```
\newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the glostylealtlongragged4col style:

```
\setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
\renewenvironment{theglossary}%
  {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
     >{\raggedright}p{\glspagelistwidth}|}}%
  {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
\renewcommand*{\glossaryheader}{%
  \hline\bfseries\entryname&\bfseries\descriptionname&
  \bfseries \symbolname&
  \bfseries\pagelistname\tabularnewline\hline\endhead
    \hline\endfoot}%
}
```

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.

```
\ProvidesPackage{glossary-mcols}[2018/06/01 v4.40 (NLCT)]
```

Required packages:

```
\RequirePackage{multicol}
\RequirePackage{glossary-tree}
```

\indexspace The are a few classes that don't define \indexspace, so provide a definition if it hasn't been defined.

```
\providecommand{\indexspace}{%
  \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
}
```

\glsmcols Define macro in which to store the number of columns. (Defaults to 2.)

```
\newcommand*{\glsmcols}{2}
```

mcolindex Multi-column index style. Same as the index, but puts the glossary in multiple columns. (Ideally the glossary title should go in the optional argument of multicols, but the title isn't part of the glossary style.)

```
\newglossarystyle{mcolindex}{%
  \setglossarystyle{index}%
  \renewenvironment{theglossary}%
    {%

     \begin{multicols}{\glsmcols}
     \setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%

     \let\item\glstreeitem
     \let\subitem\glstreesubitem
     \let\subsubitem\glstreesubsubitem
    }%
    {\end{multicols}}%
}
```

mcolindexgroup As mcolindex but has headings:

```
\newglossarystyle{mcolindexgroup}{%
 \setglossarystyle{mcolindex}%
 \renewcommand*{\glsgroupheading}[1]{%
    \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\indexspace}%
}
```

indexhypergroup The mcolindexhypergroup style is like the mcolindexgroup style but has hyper navigation.

```
\newglossarystyle{mcolindexhypergroup}{%
```

Base it on the glostylemcolindex style:

```
  \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
  \renewcommand*{\glossaryheader}{%
    \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.

```
  \renewcommand*{\glsgroupheading}[1]{%
    \item\glstreegroupheaderfmt
      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
    \indexspace}%
}
```

colindexspannav Similar to mcolindexhypergroup, but puts the navigation line in the optional argument of multicols.

```
\newglossarystyle{mcolindexspannav}{%
  \setglossarystyle{index}%
  \renewenvironment{theglossary}%
    {%
     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
     \setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%

     \let\item\glstreeitem}%
    {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
  \renewcommand*{\glsgroupheading}[1]{%
    \item\glstreegroupheaderfmt
      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
    \indexspace}%
}
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
\newglossarystyle{mcoltree}{%
  \setglossarystyle{tree}%
  \renewenvironment{theglossary}%
  {%
     \begin{multicols}{\glsmcols}
     \setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%
  }%
  {\end{multicols}}%
}
```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```
\newglossarystyle{mcoltreegroup}{%
```

Base it on the glostylemcoltree style:

```
  \setglossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

ltreehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{mcoltreehypergroup}{%
```

Base it on the glostylemcoltree style:

```
  \setglossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

mcoltreespannav Similar to the mcoltreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
\newglossarystyle{mcoltreespannav}{%
  \setglossarystyle{tree}%
  \renewenvironment{theglossary}%
  {%

     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
     \setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%
  }%
  {\end{multicols}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
\newglossarystyle{mcoltreenoname}{%
  \setglossarystyle{treenoname}%
  \renewenvironment{theglossary}%
  {%
```

```
    \begin{multicols}{\glsmcols}
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
  }%
  {\end{multicols}}%
}
```

treenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```
\newglossarystyle{mcoltreenonamegroup}{%
```

Base it on the glostylemcoltreenoname style:

```
  \setglossarystyle{mcoltreenoname}%
```

Give each group a heading:

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

onamehypergroup The mcoltreenonamehypergroup style is like the mcoltreenonamegroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{mcoltreenonamehypergroup}{%
```

Base it on the glostylemcoltreenoname style:

```
  \setglossarystyle{mcoltreenoname}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

eenonamespannav Similar to the mcoltreenonamehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
\newglossarystyle{mcoltreenonamespannav}{%
  \setglossarystyle{treenoname}%
  \renewenvironment{theglossary}%
  {%
    \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
  }%
  {\end{multicols}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

mcolalttree Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```
\newglossarystyle{mcolalttree}{%
  \setglossarystyle{alttree}%
  \renewenvironment{theglossary}%
  {%
     \begin{multicols}{\glsmcols}
     \def\@gls@prevlevel{-1}%
     \mbox{}\par
  }%
  {\par\end{multicols}}%
}
```

colalttreegroup Like the mcolalttree style but the glossary groups have headings.

```
\newglossarystyle{mcolalttreegroup}{%
```

Base it on the glostylemcolalttree style:

```
  \setglossarystyle{mcolalttree}%
```

Give each group a heading.

```
  \renewcommand{\glsgroupheading}[1]{\par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

ttreehypergroup The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{mcolalttreehypergroup}{%
```

Base it on the glostylemcolalttree style:

```
  \setglossarystyle{mcolalttree}%
```

Put the navigation links in the header

```
  \renewcommand*{\glossaryheader}{%
    \par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

lalttreespannav Similar to the mcolalttreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
\newglossarystyle{mcolalttreespannav}{%
  \setglossarystyle{alttree}%
  \renewenvironment{theglossary}%
  {%
     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
     \def\@gls@prevlevel{-1}%
     \mbox{}\par
  }%
  {\par\end{multicols}}%
```

Put a hypertarget at the start of each group

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
\ProvidesPackage{glossary-super}[2018/06/01 v4.40 (NLCT)]
```

Requires the package:

```
\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.

```
\@ifundefined{glsdescwidth}{%
  \newlength\glsdescwidth
  \setlength{\glsdescwidth}{0.6\hsize}
}{}
```

lspagelistwidth This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```
\@ifundefined{glspagelistwidth}{%
  \newlength\glspagelistwidth
  \setlength{\glspagelistwidth}{0.1\hsize}
}{}
```

super The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
\newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{}\tabletail{}%
     \begin{supertabular}{lp{\glsdescwidth}}}%
    {\end{supertabular}}%
```

Do nothing at the start of the table:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

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

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
  }%
```

Sub entries put in a row (no name, description and page list in second column):

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
     ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{& \tabularnewline}%
  \fi
}
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{superborder}{%
```

Base it on the glostylesuper style:

```
  \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline}\tabletail{\hline}%
    \begin{supertabular}{|l|p{\glsdescwidth}|}}%
   {\end{supertabular}}%
}
```

superheader The superheader style is like the super style, but with a header:

```
\newglossarystyle{superheader}{%
```

Base it on the glostylesuper style:

```
  \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
\renewenvironment{theglossary}%
  {\tablehead{\bfseries \entryname &
   \bfseries\descriptionname\tabularnewline}%
   \tabletail{}%
   \begin{supertabular}{lp{\glsdescwidth}}}%
  {\end{supertabular}}%
}
```

perheaderborder The superheaderborder style is like the super style but with a header and border:

```
\newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
  \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline\bfseries \entryname &
       \bfseries \descriptionname\tabularnewline\hline}%
     \tabletail{\hline}
     \begin{supertabular}{|l|p{\glsdescwidth}|}}%
    {\end{supertabular}}%
}
```

super3col The super3col style is like the super style, but with 3 columns:

```
\newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{}\tabletail{}%
     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
    {\end{supertabular}}%
```

Do nothing at the start of the table:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

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

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1} & ##2\tabularnewline
  }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2} &
     ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
  \fi
}
```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
\newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
  \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline}\tabletail{\hline}%
     \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

super3colheader The super3colheader style is like the super3col style but with a header row:

```
\newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```
  \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
       \bfseries\pagelistname\tabularnewline}\tabletail{}%
     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
    {\end{supertabular}}%
}
```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
\newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
  \setglossarystyle{super3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline
        \bfseries\entryname&\bfseries\descriptionname&
        \bfseries\pagelistname\tabularnewline\hline}%
     \tabletail{\hline}%
     \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

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.

```
\newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{}\tabletail{}%
     \begin{supertabular}{llll}}{%
     \end{supertabular}}%
```

Do nothing at the start of the table:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
  \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:

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1} &
    \glossentrysymbol{##1} & ##2\tabularnewline
  }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2} &
     \glossentrysymbol{##2} & ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{& & & \tabularnewline}%
  \fi
}
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
\newglossarystyle{super4colheader}{%
```

Base it on the glostylesuper4col style:

```
  \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
        \bfseries\symbolname &
        \bfseries\pagelistname\tabularnewline}%
     \tabletail{}%
     \begin{supertabular}{llll}}%
    {\end{supertabular}}%
}
```

super4colborder The super4colborder style is like the super4col but with a border.

```
\newglossarystyle{super4colborder}{%
```

Base it on the glostylesuper4col style:

```
  \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline}\tabletail{\hline}%
     \begin{supertabular}{|l|l|l|l|}}%
    {\end{supertabular}}%
}
```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```
\newglossarystyle{super4colheaderborder}{%
```

Base it on the glostylesuper4col style:

```
  \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:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
        \bfseries\symbolname &
```

```
          \bfseries\pagelistname\tabularnewline\hline}%
     \tabletail{\hline}%
     \begin{supertabular}{|l|l|l|l|}}%
    {\end{supertabular}}%
}
```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```
\newglossarystyle{altsuper4col}{%
```

Base it on the glostylesuper4col style:

```
  \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{}\tabletail{}%
     \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
    {\end{supertabular}}%
}
```

super4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```
\newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylesuper4colheader style:

```
  \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
      \bfseries\symbolname &
      \bfseries\pagelistname\tabularnewline}\tabletail{}%
     \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
    {\end{supertabular}}%
}
```

super4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```
\newglossarystyle{altsuper4colborder}{%
```

Base it on the glostylesuper4colborder style:

```
  \setglossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline}\tabletail{\hline}%
     \begin{supertabular}%
       {|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```
\newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the glostylesuper4colheaderborder style:

```
\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:

```
\renewenvironment{theglossary}%
  {\tablehead{\hline
     \bfseries\entryname &
     \bfseries\descriptionname &
     \bfseries\symbolname &
     \bfseries\pagelistname\tabularnewline\hline}%
   \tabletail{\hline}%
   \begin{supertabular}%
     {|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
  {\end{supertabular}}%
}
```

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.

```
\ProvidesPackage{glossary-superragged}[2018/06/01 v4.40 (NLCT)]
```

Requires the package:

```
\RequirePackage{array}
```

Requires the package:

```
\RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
\@ifundefined{glsdescwidth}{%
  \newlength\glsdescwidth
  \setlength{\glsdescwidth}{0.6\hsize}
}{}
```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
\@ifundefined{glspagelistwidth}{%
  \newlength\glspagelistwidth
  \setlength{\glspagelistwidth}{0.1\hsize}
}{}
```

`superragged` The superragged glossary style uses the supertabular environment.

```
\newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
\renewenvironment{theglossary}%
  {\tablehead{}\tabletail{}%
   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
  {\end{supertabular}}%
```

Do nothing at the start of the table:

```
\renewcommand*{\glossaryheader}{}%
```

No group headings:

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

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
\renewcommand{\glossentry}[2]{%
  \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
  \glossentrydesc{##1}\glspostdescription\space ##2%
  \tabularnewline
}%
```

Sub entries put in a row (no name, description and page list in second column):

```
\renewcommand{\subglossentry}[3]{%
   &
   \glssubentryitem{##2}%
   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
   ##3%
   \tabularnewline
}%
```

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)

```
\ifglsnogroupskip
  \renewcommand*{\glsgroupskip}{}%
\else
  \renewcommand*{\glsgroupskip}{& \tabularnewline}%
\fi
}
```

perraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{superraggedborder}{%
```

Base it on the glostylesuperragged style:

```
\setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
\renewenvironment{theglossary}%
  {\tablehead{\hline}\tabletail{\hline}%
   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
  {\end{supertabular}}%
}
```

perraggedheader The superraggedheader style is like the super style, but with a header:

```
\newglossarystyle{superraggedheader}{%
```

Base it on the glostylesuperragged style:

```
  \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
\renewenvironment{theglossary}%
  {\tablehead{\bfseries \entryname & \bfseries \descriptionname
      \tabularnewline}%
   \tabletail{}%
   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
  {\end{supertabular}}%
}
```

gedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
\newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostylesuper style:

```
  \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline\bfseries \entryname &
       \bfseries \descriptionname\tabularnewline\hline}%
     \tabletail{\hline}
     \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
    {\end{supertabular}}%
}
```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```
\newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{}\tabletail{}%
     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
        >{\raggedright}p{\glspagelistwidth}}}%
    {\end{supertabular}}%
```

Do nothing at the start of the table:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

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

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1} &
    ##2\tabularnewline
  }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2} &
     ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
  \fi
}
```

aged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
\newglossarystyle{superragged3colborder}{%
```

Base it on the glostylesuperragged3col style:

```
  \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline}\tabletail{\hline}%
     \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
       >{\raggedright}p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

aged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
\newglossarystyle{superragged3colheader}{%
```

Base it on the glostylesuperragged3col style:

```
  \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
       \bfseries\pagelistname\tabularnewline}\tabletail{}%
     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
       >{\raggedright}p{\glspagelistwidth}}}%
    {\end{supertabular}}%
}
```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
\newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
  \setglossarystyle{superragged3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline
        \bfseries\entryname&\bfseries\descriptionname&
        \bfseries\pagelistname\tabularnewline\hline}%
     \tabletail{\hline}%
     \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
       >{\raggedright}p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
\newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{}\tabletail{}%
     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
       >{\raggedright}p{\glspagelistwidth}}}%
    {\end{supertabular}}%
```

Do nothing at the start of the table:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
  \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:

```
  \renewcommand{\glossentry}[2]{%
    \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
    \glossentrydesc{##1} &
    \glossentrysymbol{##1} & ##2\tabularnewline
  }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
  \renewcommand{\subglossentry}[3]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}\glossentrydesc{##2} &
     \glossentrysymbol{##2} & ##3\tabularnewline
  }%
```

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)

```
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{& & & \tabularnewline}%
  \fi
}
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
\newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
  \setglossarystyle{altsuperragged4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
      \bfseries\symbolname &
      \bfseries\pagelistname\tabularnewline}\tabletail{}%
     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
       >{\raggedright}p{\glspagelistwidth}}}%
    {\end{supertabular}}%
}
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
\newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
  \setglossarystyle{altsuper4col}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline}\tabletail{\hline}%
     \begin{supertabular}%
       {|l|>{\raggedright}p{\glsdescwidth}|l|%
         >{\raggedright}p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
\newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
  \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:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline
       \bfseries\entryname &
       \bfseries\descriptionname &
       \bfseries\symbolname &
       \bfseries\pagelistname\tabularnewline\hline}%
     \tabletail{\hline}%
     \begin{supertabular}%
       {|l|>{\raggedright}p{\glsdescwidth}|l|%
          >{\raggedright}p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

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.

```
\ProvidesPackage{glossary-tree}[2018/06/01 v4.40 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
\providecommand{\indexspace}{%
  \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
}
```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glsnamefont`.) This command was previously also used to format the group headings.

```
\newcommand*{\glstreenamefmt}[1]{\textbf{#1}}
```

`egroupheaderfmt` 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.

```
\newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}
```

`eenavigationfmt` Format used to display the navigation header in the tree styles.

```
\newcommand*{\glstreenavigationfmt}[1]{\glstreenamefmt{#1}}
```

Allow the user to adjust the index style without disturbing the index.

`\glstreeitem` Top level item used in index style.

```
\ifdef\@idxitem
{\newcommand{\glstreeitem}{\@idxitem}}
{\newcommand{\glstreeitem}{\par\hangindent40\p@}}
```

\glstreesubitem Level 1 item used in index style.

```
\ifdef\subitem
{\let\glstreesubitem\subitem}
{\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

streesubsubitem Level 1 item used in index style.

```
\ifdef\subsubitem
{\let\glstreesubsubitem\subsubitem}
{\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

\glstreepredesc Allow the user to adjust the space before the description (except for the alttree style).

```
\newcommand{\glstreepredesc}{\space}
```

reechildpredesc Allow the user to adjust the space before the description for sub-entries (except for the treenoname and alttree style).

```
\newcommand{\glstreechildpredesc}{\space}
```

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.

```
\newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define \item to be the same as that used by theindex:

```
  \renewenvironment{theglossary}%
    {\setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%
     \let\item\glstreeitem
     \let\subitem\glstreesubitem
     \let\subsubitem\glstreesubsubitem
    }%

    {\par}%
```

Do nothing at the start of the environment:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
  \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.

```
  \renewcommand*{\glossentry}[2]{%
     \item\glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
     \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
  }%
```

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.

```
 \renewcommand{\subglossentry}[3]{%
   \ifcase##1\relax
     % level 0
     \item
   \or
     % level 1
     \subitem
     \glssubentryitem{##2}%
   \else
     % all other levels
     \subsubitem
   \fi
   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
 }%
```

Vertical gap between groups is the same as that used by indices:

```
 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The indexgroup style is like the index style but has headings.

```
\newglossarystyle{indexgroup}{%
```

Base it on the glostyleindex style:

```
 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

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

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```
\newglossarystyle{indexhypergroup}{%
```

Base it on the glostyleindex style:

```
 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```
 \renewcommand*{\glossaryheader}{%
   \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.

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

```
      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
    \indexspace}%
}
```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```
\newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
  \renewenvironment{theglossary}%
    {\setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}}%
    {}%
```

Do nothing at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

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

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
  \renewcommand{\glossentry}[2]{%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
    \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
    \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
  }%
```

Sub entries: level ⟨*n*⟩ is indented by ⟨*n*⟩ times `\glstreeindent`. The name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
  \renewcommand{\subglossentry}[3]{%
    \hangindent##1\glstreeindent\relax
    \parindent##1\glstreeindent\relax
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
    \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
    \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
  }%
```

Vertical gap between groups is the same as that used by indices:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```
\newglossarystyle{treegroup}{%
```

Base it on the glostyletree style:

```
  \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par
    \indexspace}%
}
```

treehypergroup The treehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{treehypergroup}{%
```

Base it on the glostyletree style:

```
  \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \glstreegroupheaderfmt
      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

\glstreeindent Length governing left indent for each level of the tree style.

```
\newlength\glstreeindent
\setlength{\glstreeindent}{10pt}
```

treenoname The treenoname glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
\newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
  \renewenvironment{theglossary}%
    {\setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}}%
    {}%
```

No header:

```
  \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
  \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.

```
  \renewcommand{\glossentry}[2]{%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
    \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
    \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
  }%
```

Sub entries: level ⟨*n*⟩ is indented by ⟨*n*⟩ times `\glstreeindent`. The name and symbol are omitted. The description followed by the page list are displayed.

```
  \renewcommand{\subglossentry}[3]{%
    \hangindent##1\glstreeindent\relax
    \parindent##1\glstreeindent\relax
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \glstarget{##2}{\strut}%
    \glossentrydesc{##2}\glspostdescription\space##3\par
  }%
```

Vertical gap between groups is the same as that used by indices:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
}
```

`treenonamegroup` Like the treenoname style but the glossary groups have headings.

```
\newglossarystyle{treenonamegroup}{%
```

Base it on the glostyletreenoname style:

```
  \setglossarystyle{treenoname}%
```

Give each group a heading:

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\glstreegroupheaderfmt
      {\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

`onamehypergroup` The treenonamehypergroup style is like the treenonamegroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{treenonamehypergroup}{%
```

Base it on the glostyletreenoname style:

```
  \setglossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \glstreegroupheaderfmt
      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

esttoplevelname Find the widest name over all parentless entries in the given glossary or glossaries.

```
\newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglshasparent{\@glo@label}%
      {}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \letcs{\@glswidestname}{glo@\glsdetoklabel{\@glo@label}@name}%
        \fi
      }%
    }%
  }%
}
```

\glssetwidest \glssetwidest[⟨*level*⟩]{⟨*text*⟩} sets the widest text for the given level. It is used by the alttree glossary styles to determine the indentation of each level.

```
\newcommand*{\glssetwidest}[2][0]{%
  \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
    #2}%
}
```

\@glswidestname Initialise \@glswidestname.

```
\newcommand*{\@glswidestname}{}
```

\glstreenamebox Used by the alttree style to create the box for the name and associated information.

```
\newcommand*{\glstreenamebox}[2]{%
  \makebox[#1][l]{#2}%
}
```

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.

```
\newglossarystyle{alttree}{%
```

Redefine theglossary environment.

```
  \renewenvironment{theglossary}%
    {\def\@gls@prevlevel{-1}%
     \mbox{}\par}%
    {\par}%
```

Set the header and group headers to nothing.

```
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
\renewcommand{\glossentry}[2]{%
  \ifnum\@gls@prevlevel=0\relax
  \else
```

Find out how big the indentation should be by measuring the widest entry.

```
     \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
  \fi
```

Set the hangindent and paragraph indent.

```
  \hangindent\glstreeindent
  \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
  \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
  \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
  \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
  \def\@gls@prevlevel{0}%
}%
```

Redefine the way sub-entries are displayed.

```
\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.

```
  \ifnum##1=1\relax
    \glssubentryitem{##2}%
  \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
  \ifnum\@gls@prevlevel=##1\relax
  \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
    \@ifundefined{@glswidestname\romannumeral##1}{%
      \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}}{%
      \settowidth{\gls@tmplen}{\glstreenamefmt{%
         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
    \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
          \setlength\glstreeindent\gls@tmplen
          \addtolength\glstreeindent\parindent
          \parindent\glstreeindent
        \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`.

```
          \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
            \settowidth{\glstreeindent}{\glstreenamefmt{%
              \@glswidestname\space}}}{%
            \settowidth{\glstreeindent}{\glstreenamefmt{%
              \csname @glswidestname\romannumeral\@gls@prevlevel
                \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
          \addtolength\parindent{-\glstreeindent}%
          \setlength\glstreeindent\parindent
        \fi
      \fi
```

Set the hanging indentation.

```
      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
      \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
        \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
      \def\@gls@prevlevel{##1}%
    }%
```

Vertical gap between groups is the same as that used by indices:

```
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
}
```

`alttreegroup` Like the alttree style but the glossary groups have headings.

```
\newglossarystyle{alttreegroup}{%
```

Base it on the glostylealttree style:

```
    \setglossarystyle{alttree}%
```

Give each group a heading.

```
    \renewcommand{\glsgroupheading}[1]{\par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
      \par\indexspace}%
  }
```

ttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{alttreehypergroup}{%
```

Base it on the glostylealttree style:

```
  \setglossarystyle{alttree}%
```

Put the navigation links in the header

```
  \renewcommand*{\glossaryheader}{%
    \par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glstreegroupheaderfmt
     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \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.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-compatible-207}[2018/06/01 v4.40 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
\ifglsxindy
  \renewcommand*\GlsAddXdyAttribute[1]{%
  \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
  \expandafter\toks@\expandafter{\@xdylocref}%
  \edef\@xdylocref{\the\toks@ ^^J%
  (markup-locref
  :open \string"\string~n\string\setentrycounter
    {\noexpand\glscounter}%
    \expandafter\string\csname#1\endcsname
    \expandafter\@gobble\string\{\string" ^^J
  :close \string"\expandafter\@gobble\string\}\string" ^^J
  :attr \string"#1\string")}}
```

Only has an effect before \writeist:

```
\fi
```

sAddXdyCounters

```
\renewcommand*\GlsAddXdyCounters[1]{%
  \GlossariesWarning{\string\GlsAddXdyCounters\space not available
    in compatibility mode.}%
}
```

Add predefined attributes

```
  \GlsAddXdyAttribute{glsnumberformat}
  \GlsAddXdyAttribute{textrm}
  \GlsAddXdyAttribute{textsf}
  \GlsAddXdyAttribute{texttt}
  \GlsAddXdyAttribute{textbf}
  \GlsAddXdyAttribute{textmd}
  \GlsAddXdyAttribute{textit}
  \GlsAddXdyAttribute{textup}
  \GlsAddXdyAttribute{textsl}
```

```
\GlsAddXdyAttribute{textsc}
\GlsAddXdyAttribute{emph}
\GlsAddXdyAttribute{glshypernumber}
\GlsAddXdyAttribute{hyperrm}
\GlsAddXdyAttribute{hypersf}
\GlsAddXdyAttribute{hypertt}
\GlsAddXdyAttribute{hyperbf}
\GlsAddXdyAttribute{hypermd}
\GlsAddXdyAttribute{hyperit}
\GlsAddXdyAttribute{hyperup}
\GlsAddXdyAttribute{hypersl}
\GlsAddXdyAttribute{hypersc}
\GlsAddXdyAttribute{hyperemph}
```

`sAddXdyLocation` Restore v2.07 definition:

```
\ifglsxindy
  \renewcommand*{\GlsAddXdyLocation}[2]{%
    \edef\@xdyuserlocationdefs{%
      \@xdyuserlocationdefs ^^J%
      (define-location-class \string"#1\string"^^J\space\space
      \space(#2))
    }%
    \edef\@xdyuserlocationnames{%
      \@xdyuserlocationnames^^J\space\space\space
      \string"#1\string"}%
  }
\fi
```

`\@do@wrglossary`

```
\renewcommand{\@do@wrglossary}[1]{%
```

Determine whether to use `xindy` or `makeindex` syntax

```
\ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```
  \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
  \def\@glo@range{}%
  \expandafter\if\@glo@prefix(\relax
    \def\@glo@range{:open-range}%
  \else
    \expandafter\if\@glo@prefix)\relax
      \def\@glo@range{:close-range}%
    \fi
  \fi
```

Get the location and escape any special characters

```
  \protected@edef\@glslocref{\theglsentrycounter}%
  \@gls@checkmkidxchars\@glslocref
```

Write to the glossary file using `xindy` syntax.

```
  \glossary[\csname glo@#1@type\endcsname]{%
  (indexentry :tkey (\csname glo@#1@index\endcsname)
    :locref \string"\@glslocref\string" %
    :attr \string"\@glo@suffix\string" \@glo@range
  )
  }%
\else
```

Convert the format information into the format required for `makeindex`

```
  \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
```

Write to the glossary file using `makeindex` syntax.

```
  \glossary[\csname glo@#1@type\endcsname]{%
  \string\glossaryentry{\csname glo@#1@index\endcsname
    \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
\fi
}
```

`t@glo@numformat` Only had 3 arguments in v2.07

```
\def\@set@glo@numformat#1#2#3{%
  \expandafter\@glo@check@mkidxrangechar#3\@nil
  \protected@edef#1{%
    \@glo@prefix setentrycounter[]{#2}%
    \expandafter\string\csname\@glo@suffix\endcsname
  }%
  \@gls@checkmkidxchars#1%
}
```

`\writeist` Redefine `\writeist` back to the way it was in v2.07, but change `\istfile` to `\glswrite`.

```
\ifglsxindy
  \def\writeist{%
    \openout\glswrite=\istfilename
    \write\glswrite{;; xindy style file created by the glossaries
      package in compatible-2.07 mode}%
    \write\glswrite{;; for document '\jobname' on
      \the\year-\the\month-\the\day}%
    \write\glswrite{^^J; required styles^^J}
    \@for\@xdystyle:=\@xdyrequiredstyles\do{%
       \ifx\@xdystyle\@empty
       \else
         \protected@write\glswrite{}{(require
           \string"\@xdystyle.xdy\string")}%
       \fi
    }%
    \write\glswrite{^^J%
       ; list of allowed attributes (number formats)^^J}%
    \write\glswrite{(define-attributes ((\@xdyattributes)))}%
    \write\glswrite{^^J; user defined alphabets^^J}%
    \write\glswrite{\@xdyuseralphabets}%
    \write\glswrite{^^J; location class definitions^^J}%
    \protected@edef\@gls@roman{\@roman{0\string"
      \string"roman-numbers-lowercase\string" :sep \string"}}%
    \@onelevel@sanitize\@gls@roman
    \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
       :sep \string"}%
    \@onelevel@sanitize\@tmp
    \ifx\@tmp\@gls@roman
       \write\glswrite{(define-location-class
         \string"roman-page-numbers\string"^^J\space\space\space
         (\string"roman-numbers-lowercase\string")
         :min-range-length \@glsminrange)}%
    \else
       \write\glswrite{(define-location-class
         \string"roman-page-numbers\string"^^J\space\space\space
         (:sep "\@gls@roman")
         :min-range-length \@glsminrange)}%
    \fi
    \write\glswrite{(define-location-class
      \string"Roman-page-numbers\string"^^J\space\space\space
      (\string"roman-numbers-uppercase\string")
         :min-range-length \@glsminrange)}%
    \write\glswrite{(define-location-class
      \string"arabic-page-numbers\string"^^J\space\space\space
      (\string"arabic-numbers\string")
         :min-range-length \@glsminrange)}%
    \write\glswrite{(define-location-class
      \string"alpha-page-numbers\string"^^J\space\space\space
      (\string"alpha\string")
         :min-range-length \@glsminrange)}%
    \write\glswrite{(define-location-class
      \string"Alpha-page-numbers\string"^^J\space\space\space
      (\string"ALPHA\string")
         :min-range-length \@glsminrange)}%
    \write\glswrite{(define-location-class
      \string"Appendix-page-numbers\string"^^J\space\space\space
      (\string"ALPHA\string"
       :sep \string"\@glsAlphacompositor\string"
       \string"arabic-numbers\string")
         :min-range-length \@glsminrange)}%
    \write\glswrite{(define-location-class
      \string"arabic-section-numbers\string"^^J\space\space\space
      (\string"arabic-numbers\string"
       :sep \string"\glscompositor\string"
       \string"arabic-numbers\string")
         :min-range-length \@glsminrange)}%
    \write\glswrite{^^J; user defined location classes}%
    \write\glswrite{\@xdyuserlocationdefs}%
    \write\glswrite{^^J; define cross-reference class^^J}%
    \write\glswrite{(define-crossref-class \string"see\string"
      :unverified )}%
```

```
\write\glswrite{(markup-crossref-list
   :class \string"see\string"^^J\space\space\space
   :open \string"\string\glsseeformat\string"
   :close \string"{}\string")}%
\write\glswrite{^^J; define the order of the location classes}%
\write\glswrite{(define-location-class-order
   (\@xdylocationclassorder))}%
\write\glswrite{^^J; define the glossary markup^^J}%
\write\glswrite{(markup-index^^J\space\space\space
  :open \string"\string
  \glossarysection[\string\glossarytoctitle]{\string
  \glossarytitle}\string\glossarypreamble\string~n\string\begin
  {theglossary}\string\glossaryheader\string~n\string" ^^J\space
  \space\space:close \string"\expandafter\@gobble
    \string\%\string~n\string
    \end{theglossary}\string\glossarypostamble
    \string~n\string" ^^J\space\space\space
  :tree)}%
\write\glswrite{(markup-letter-group-list
  :sep \string"\string\glsgroupskip\string~n\string")}%
\write\glswrite{(markup-indexentry
  :open \string"\string\relax \string\glsresetentrylist
     \string~n\string")}%
\write\glswrite{(markup-locclass-list :open
 \string"\glsopenbrace\string\glossaryentrynumbers
   \glsopenbrace\string\relax\space \string"^^J\space\space\space
 :sep \string", \string"
 :close \string"\glsclosebrace\glsclosebrace\string")}%
\write\glswrite{(markup-locref-list
 :sep \string"\string\delimN\space\string")}%
\write\glswrite{(markup-range
 :sep \string"\string\delimR\space\string")}%
\@onelevel@sanitize\gls@suffixF
\@onelevel@sanitize\gls@suffixFF
\ifx\gls@suffixF\@empty
\else
  \write\glswrite{(markup-range
  :close "\gls@suffixF" :length 1 :ignore-end)}%
\fi
\ifx\gls@suffixFF\@empty
\else
  \write\glswrite{(markup-range
  :close "\gls@suffixFF" :length 2 :ignore-end)}%
\fi
\write\glswrite{^^J; define format to use for locations^^J}%
\write\glswrite{\@xdylocref}%
\write\glswrite{^^J; define letter group list format^^J}%
\write\glswrite{(markup-letter-group-list
 :sep \string"\string\glsgroupskip\string~n\string")}%
```

```
    \write\glswrite{^^J; letter group headings^^J}%
    \write\glswrite{(markup-letter-group
      :open-head \string"\string\glsgroupheading
      \glsopenbrace\string"^^J\space\space\space
      :close-head \string"\glsclosebrace\string")}%
    \write\glswrite{^^J; additional letter groups^^J}%
    \write\glswrite{\@xdylettergroups}%
    \write\glswrite{^^J; additional sort rules^^J}
    \write\glswrite{\@xdysortrules}%
  \noist}
\else
  \edef\@gls@actualchar{\string?}
  \edef\@gls@encapchar{\string|}
  \edef\@gls@levelchar{\string!}
  \edef\@gls@quotechar{\string"}
  \def\writeist{\relax
    \openout\glswrite=\istfilename
    \write\glswrite{\expandafter\@gobble\string\% makeindex style file
      created by the glossaries package}
    \write\glswrite{\expandafter\@gobble\string\% for document
      '\jobname' on \the\year-\the\month-\the\day}
    \write\glswrite{actual '\@gls@actualchar'}
    \write\glswrite{encap '\@gls@encapchar'}
    \write\glswrite{level '\@gls@levelchar'}
    \write\glswrite{quote '\@gls@quotechar'}
    \write\glswrite{keyword \string"\string\\glossaryentry\string"}
    \write\glswrite{preamble \string"\string\\glossarysection[\string
      \\glossarytoctitle]{\string\\glossarytitle}\string
      \\glossarypreamble\string\n\string\\begin{theglossary}\string
      \\glossaryheader\string\n\string"}
    \write\glswrite{postamble \string"\string\%\string\n\string
      \\end{theglossary}\string\\glossarypostamble\string\n
      \string"}
    \write\glswrite{group_skip \string"\string\\glsgroupskip\string\n
      \string"}
    \write\glswrite{item_0 \string"\string\%\string\n\string"}
    \write\glswrite{item_1 \string"\string\%\string\n\string"}
    \write\glswrite{item_2 \string"\string\%\string\n\string"}
    \write\glswrite{item_01 \string"\string\%\string\n\string"}
    \write\glswrite{item_x1
      \string"\string\\relax \string\\glsresetentrylist\string\n
      \string"}
    \write\glswrite{item_12 \string"\string\%\string\n\string"}
    \write\glswrite{item_x2
      \string"\string\\relax \string\\glsresetentrylist\string\n
      \string"}
    \write\glswrite{delim_0 \string"\string\{\string
      \\glossaryentrynumbers\string\{\string\\relax \string"}
    \write\glswrite{delim_1 \string"\string\{\string
      \\glossaryentrynumbers\string\{\string\\relax \string"}
    \write\glswrite{delim_2 \string"\string\{\string
      \\glossaryentrynumbers\string\{\string\\relax \string"}
    \write\glswrite{delim_t \string"\string\}\string\}\string"}
    \write\glswrite{delim_n \string"\string\\delimN \string"}
    \write\glswrite{delim_r \string"\string\\delimR \string"}
    \write\glswrite{headings_flag 1}
    \write\glswrite{heading_prefix
       \string"\string\\glsgroupheading\string\{\string"}
    \write\glswrite{heading_suffix
       \string"\string\}\string\\relax
       \string\\glsresetentrylist \string"}
    \write\glswrite{symhead_positive \string"glssymbols\string"}
    \write\glswrite{numhead_positive \string"glsnumbers\string"}
    \write\glswrite{page_compositor \string"\glscompositor\string"}
    \@gls@escbsdq\gls@suffixF
    \@gls@escbsdq\gls@suffixFF
    \ifx\gls@suffixF\@empty
    \else
      \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
    \fi
    \ifx\gls@suffixFF\@empty
    \else
      \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
    \fi
    \noist
   }
\fi
```

\noist

```
\renewcommand*{\noist}{\let\writeist\relax}
```

4.2 glossaries-compatible-307

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-compatible-307}[2018/06/01 v4.40 (NLCT)]
```

Compatibility macros for predefined glossary styles:

atglossarystyle Defines a compatibility glossary style.

```
\newcommand{\compatglossarystyle}[2]{%
  \ifcsundef{@glscompstyle@#1}%
  {%
    \csdef{@glscompstyle@#1}{#2}%
  }%
  {%
    \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
  }%
}
```

Backward compatible inline style.

```
\compatglossarystyle{inline}{%
  \renewcommand{\glossaryentryfield}[5]{%
    \glsinlinedopostchild
    \gls@inlinesep
    \def\glo@desc{##3}%
    \def\@no@post@desc{\nopostdesc}%
    \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
    \ifx\glo@desc\@no@post@desc
      \glsinlineemptydescformat{##4}{##5}%
    \else
      \ifstrempty{##3}%
      {\glsinlineemptydescformat{##4}{##5}}%
      {\glsinlinedescformat{##3}{##4}{##5}}%
    \fi
    \ifglshaschildren{##1}%
    {%
       \glsresetsubentrycounter
       \glsinlineparentchildseparator
       \def\gls@inlinesubsep{}%
       \def\gls@inlinepostchild{\glsinlinepostchild}%
    }%
    {}%
    \def\gls@inlinesep{\glsinlineseparator}%
  }%
```

Sub-entries display description:

```
  \renewcommand{\glossarysubentryfield}[6]{%
    \gls@inlinesubsep%
    \glsinlinesubnameformat{##2}{##3}%
    \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
    \def\gls@inlinesubsep{\glsinlinesubseparator}%
  }%
}
```

Backward compatible list style.

```
\compatglossarystyle{list}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \item[\glsentryitem{##1}\glstarget{##1}{##2}]
       ##3\glspostdescription\space ##5}%
```

Sub-entries continue on the same line:

```
  \renewcommand*{\glossarysubentryfield}[6]{%
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
}
```

Backward compatible listgroup style.

```
\compatglossarystyle{listgroup}{%
 \csuse{@glscompstyle@list}%
}%
```

Backward compatible listhypergroup style.

```
\compatglossarystyle{listhypergroup}{%
 \csuse{@glscompstyle@list}%
}%
```

Backward compatible altlist style.

```
\compatglossarystyle{altlist}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
      \mbox{}\par\nobreak\@afterheading
      ##3\glspostdescription\space ##5}%
  \renewcommand{\glossarysubentryfield}[6]{%
    \par
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
}%
```

Backward compatible altlistgroup style.

```
\compatglossarystyle{altlistgroup}{%
 \csuse{@glscompstyle@altlist}%
}%
```

Backward compatible altlisthypergroup style.

```
\compatglossarystyle{altlisthypergroup}{%
 \csuse{@glscompstyle@altlist}%
}%
```

Backward compatible listdotted style.

```
\compatglossarystyle{listdotted}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
      \glsentryitem{##1}\glstarget{##1}{##2}%
      \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
  \renewcommand*{\glossarysubentryfield}[6]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
    \glssubentryitem{##2}%
    \glstarget{##2}{##3}%
    \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
}%
```

Backward compatible sublistdotted style.

```
\compatglossarystyle{sublistdotted}{%
  \csuse{@glscompstyle@listdotted}%
  \renewcommand*{\glossaryentryfield}[5]{%
    \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
}%
```

Backward compatible long style.

```
\compatglossarystyle{long}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
  \renewcommand*{\glossarysubentryfield}[6]{%
```

```
    &
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
}%
```

Backward compatible longborder style.

```
\compatglossarystyle{longborder}{%
 \csuse{@glscompstyle@long}%
}%
```

Backward compatible longheader style.

```
\compatglossarystyle{longheader}{%
 \csuse{@glscompstyle@long}%
}%
```

Backward compatible longheaderborder style.

```
\compatglossarystyle{longheaderborder}{%
 \csuse{@glscompstyle@long}%
}%
```

Backward compatible long3col style.

```
\compatglossarystyle{long3col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
  \renewcommand*{\glossarysubentryfield}[6]{%
    &
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}##4 & ##6\\}%
}%
```

Backward compatible long3colborder style.

```
\compatglossarystyle{long3colborder}{%
 \csuse{@glscompstyle@long3col}%
}%
```

Backward compatible long3colheader style.

```
\compatglossarystyle{long3colheader}{%
 \csuse{@glscompstyle@long3col}%
}%
```

Backward compatible long3colheaderborder style.

```
\compatglossarystyle{long3colheaderborder}{%
 \csuse{@glscompstyle@long3col}%
}%
```

Backward compatible long4col style.

```
\compatglossarystyle{long4col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
  \renewcommand*{\glossarysubentryfield}[6]{%
    &
    \glssubentryitem{##2}%
```

```
      \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
}%
```

Backward compatible long4colheader style.

```
\compatglossarystyle{long4colheader}{%
 \csuse{@glscompstyle@long4col}%
}%
```

Backward compatible long4colborder style.

```
\compatglossarystyle{long4colborder}{%
 \csuse{@glscompstyle@long4col}%
}%
```

Backward compatible long4colheaderborder style.

```
\compatglossarystyle{long4colheaderborder}{%
 \csuse{@glscompstyle@long4col}%
}%
```

Backward compatible altlong4col style.

```
\compatglossarystyle{altlong4col}{%
 \csuse{@glscompstyle@long4col}%
}%
```

Backward compatible altlong4colheader style.

```
\compatglossarystyle{altlong4colheader}{%
 \csuse{@glscompstyle@long4col}%
}%
```

Backward compatible altlong4colborder style.

```
\compatglossarystyle{altlong4colborder}{%
 \csuse{@glscompstyle@long4col}%
}%
```

Backward compatible altlong4colheaderborder style.

```
\compatglossarystyle{altlong4colheaderborder}{%
 \csuse{@glscompstyle@long4col}%
}%
```

Backward compatible long style.

```
\compatglossarystyle{longragged}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
    \tabularnewline}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
    \tabularnewline}%
}%
```

Backward compatible longraggedborder style.

```
\compatglossarystyle{longraggedborder}{%
 \csuse{@glscompstyle@longragged}%
}%
```

Backward compatible longraggedheader style.

```
\compatglossarystyle{longraggedheader}{%
 \csuse{@glscompstyle@longragged}%
}%
```

Backward compatible longraggedheaderborder style.

```
\compatglossarystyle{longraggedheaderborder}{%
 \csuse{@glscompstyle@longragged}%
}%
```

Backward compatible longragged3col style.

```
\compatglossarystyle{longragged3col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
}%
```

Backward compatible longragged3colborder style.

```
\compatglossarystyle{longragged3colborder}{%
 \csuse{@glscompstyle@longragged3col}%
}%
```

Backward compatible longragged3colheader style.

```
\compatglossarystyle{longragged3colheader}{%
 \csuse{@glscompstyle@longragged3col}%
}%
```

Backward compatible longragged3colheaderborder style.

```
\compatglossarystyle{longragged3colheaderborder}{%
 \csuse{@glscompstyle@longragged3col}%
}%
```

Backward compatible altlongragged4col style.

```
\compatglossarystyle{altlongragged4col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
}%
```

Backward compatible altlongragged4colheader style.

```
\compatglossarystyle{altlongragged4colheader}{%
 \csuse{@glscompstyle@altlong4col}%
}%
```

Backward compatible altlongragged4colborder style.

```
\compatglossarystyle{altlongragged4colborder}{%
```

```
 \csuse{@glscompstyle@altlong4col}%
}%
```

Backward compatible altlongragged4colheaderborder style.

```
\compatglossarystyle{altlongragged4colheaderborder}{%
 \csuse{@glscompstyle@altlong4col}%
}%
```

Backward compatible index style.

```
\compatglossarystyle{index}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
      \ifx\relax##4\relax
      \else
        \space(##4)%
      \fi
      \space ##3\glspostdescription \space ##5}%
  \renewcommand*{\glossarysubentryfield}[6]{%
    \ifcase##1\relax
      % level 0
      \item
    \or
      % level 1
      \subitem
      \glssubentryitem{##2}%
    \else
      % all other levels
      \subsubitem
    \fi
    \textbf{\glstarget{##2}{##3}}%
    \ifx\relax##5\relax
    \else
      \space(##5)%
    \fi
    \space##4\glspostdescription\space ##6}%
}%
```

Backward compatible indexgroup style.

```
\compatglossarystyle{indexgroup}{%
 \csuse{@glscompstyle@index}%
}%
```

Backward compatible indexhypergroup style.

```
\compatglossarystyle{indexhypergroup}{%
 \csuse{@glscompstyle@index}%
}%
```

Backward compatible tree style.

```
\compatglossarystyle{tree}{%
  \renewcommand{\glossaryentryfield}[5]{%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
    \ifx\relax##4\relax
    \else
      \space(##4)%
    \fi
    \space ##3\glspostdescription \space ##5\par}%
  \renewcommand{\glossarysubentryfield}[6]{%
    \hangindent##1\glstreeindent\relax
    \parindent##1\glstreeindent\relax
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \textbf{\glstarget{##2}{##3}}%
    \ifx\relax##5\relax
    \else
      \space(##5)%
    \fi
    \space##4\glspostdescription\space ##6\par}%
}%
```

Backward compatible treegroup style.

```
\compatglossarystyle{treegroup}{%
 \csuse{@glscompstyle@tree}%
}%
```

Backward compatible treehypergroup style.

```
\compatglossarystyle{treehypergroup}{%
 \csuse{@glscompstyle@tree}%
}%
```

Backward compatible treenoname style.

```
\compatglossarystyle{treenoname}{%
  \renewcommand{\glossaryentryfield}[5]{%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
    \ifx\relax##4\relax
    \else
      \space(##4)%
    \fi
    \space ##3\glspostdescription \space ##5\par}%
  \renewcommand{\glossarysubentryfield}[6]{%
    \hangindent##1\glstreeindent\relax
    \parindent##1\glstreeindent\relax
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \glstarget{##2}{\strut}%
    ##4\glspostdescription\space ##6\par}%
}%
```

Backward compatible treenonamegroup style.

```
\compatglossarystyle{treenonamegroup}{%
 \csuse{@glscompstyle@treenoname}%
}%
```

Backward compatible treenonamehypergroup style.

```
\compatglossarystyle{treenonamehypergroup}{%
 \csuse{@glscompstyle@treenoname}%
}%
```

Backward compatible alttree style.

```
\compatglossarystyle{alttree}{%
  \renewcommand{\glossaryentryfield}[5]{%
    \ifnum\@gls@prevlevel=0\relax
    \else
       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
      \hangindent\glstreeindent
      \parindent\glstreeindent
    \fi
    \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
       \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
    \ifx\relax##4\relax
    \else
      (##4)\space
    \fi
    ##3\glspostdescription \space ##5\par
    \def\@gls@prevlevel{0}%
  }%
  \renewcommand{\glossarysubentryfield}[6]{%
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \ifnum\@gls@prevlevel=##1\relax
    \else
      \@ifundefined{@glswidestname\romannumeral##1}{%
        \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}}{%
        \settowidth{\gls@tmplen}{\textbf{%
           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
      \ifnum\@gls@prevlevel<##1\relax
         \setlength\glstreeindent\gls@tmplen
         \addtolength\glstreeindent\parindent
         \parindent\glstreeindent
      \else
         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
           \settowidth{\glstreeindent}{\textbf{%
              \@glswidestname\space}}}{%
           \settowidth{\glstreeindent}{\textbf{%
              \csname @glswidestname\romannumeral\@gls@prevlevel
                 \endcsname\space}}}%
         \addtolength\parindent{-\glstreeindent}%
         \setlength\glstreeindent\parindent
      \fi
    \fi
    \hangindent\glstreeindent
    \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
      \textbf{\glstarget{##2}{##3}}}}%
    \ifx##5\relax\relax
    \else
      (##5)\space
    \fi
    ##4\glspostdescription\space ##6\par
    \def\@gls@prevlevel{##1}%
  }%
}%
```

Backward compatible alttreegroup style.

```
\compatglossarystyle{alttreegroup}{%
 \csuse{@glscompstyle@alttree}%
}%
```

Backward compatible alttreehypergroup style.

```
\compatglossarystyle{alttreehypergroup}{%
 \csuse{@glscompstyle@alttree}%
}%
```

Backward compatible mcolindex style.

```
\compatglossarystyle{mcolindex}{%
 \csuse{@glscompstyle@index}%
}%
```

Backward compatible mcolindexgroup style.

```
\compatglossarystyle{mcolindexgroup}{%
 \csuse{@glscompstyle@index}%
}%
```

Backward compatible mcolindexhypergroup style.

```
\compatglossarystyle{mcolindexhypergroup}{%
 \csuse{@glscompstyle@index}%
}%
```

Backward compatible mcoltree style.

```
\compatglossarystyle{mcoltree}{%
 \csuse{@glscompstyle@tree}%
}%
```

Backward compatible mcoltreegroup style.

```
\compatglossarystyle{mcolindextreegroup}{%
 \csuse{@glscompstyle@tree}%
}%
```

Backward compatible mcoltreehypergroup style.

```
\compatglossarystyle{mcolindextreehypergroup}{%
```

```
  \csuse{@glscompstyle@tree}%
}%
```

Backward compatible mcoltreenoname style.

```
\compatglossarystyle{mcoltreenoname}{%
  \csuse{@glscompstyle@tree}%
}%
```

Backward compatible mcoltreenonamegroup style.

```
\compatglossarystyle{mcoltreenonamegroup}{%
  \csuse{@glscompstyle@tree}%
}%
```

Backward compatible mcoltreenonamehypergroup style.

```
\compatglossarystyle{mcoltreenonamehypergroup}{%
  \csuse{@glscompstyle@tree}%
}%
```

Backward compatible mcolalttree style.

```
\compatglossarystyle{mcolalttree}{%
  \csuse{@glscompstyle@alttree}%
}%
```

Backward compatible mcolalttreegroup style.

```
\compatglossarystyle{mcolalttreegroup}{%
  \csuse{@glscompstyle@alttree}%
}%
```

Backward compatible mcolalttreehypergroup style.

```
\compatglossarystyle{mcolalttreehypergroup}{%
  \csuse{@glscompstyle@alttree}%
}%
```

Backward compatible superragged style.

```
\compatglossarystyle{superragged}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
      \tabularnewline}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
     \tabularnewline}%
}%
```

Backward compatible superraggedborder style.

```
\compatglossarystyle{superraggedborder}{%
  \csuse{@glscompstyle@superragged}%
}%
```

Backward compatible superraggedheader style.

```
\compatglossarystyle{superraggedheader}{%
  \csuse{@glscompstyle@superragged}%
}%
```

Backward compatible superraggedheaderborder style.

```
\compatglossarystyle{superraggedheaderborder}{%
 \csuse{@glscompstyle@superragged}%
}%
```

Backward compatible superragged3col style.

```
\compatglossarystyle{superragged3col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
}%
```

Backward compatible superragged3colborder style.

```
\compatglossarystyle{superragged3colborder}{%
 \csuse{@glscompstyle@superragged3col}%
}%
```

Backward compatible superragged3colheader style.

```
\compatglossarystyle{superragged3colheader}{%
 \csuse{@glscompstyle@superragged3col}%
}%
```

Backward compatible superragged3colheaderborder style.

```
\compatglossarystyle{superragged3colheaderborder}{%
 \csuse{@glscompstyle@superragged3col}%
}%
```

Backward compatible altsuperragged4col style.

```
\compatglossarystyle{altsuperragged4col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
}%
```

Backward compatible altsuperragged4colheader style.

```
\compatglossarystyle{altsuperragged4colheader}{%
 \csuse{@glscompstyle@altsuperragged4col}%
}%
```

Backward compatible altsuperragged4colborder style.

```
\compatglossarystyle{altsuperragged4colborder}{%
 \csuse{@glscompstyle@altsuperragged4col}%
}%
```

Backward compatible altsuperragged4colheaderborder style.

```
\compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
 \csuse{@glscompstyle@altsuperragged4col}%
}%
```

Backward compatible super style.

```
\compatglossarystyle{super}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
}%
```

Backward compatible superborder style.

```
\compatglossarystyle{superborder}{%
 \csuse{@glscompstyle@super}%
}%
```

Backward compatible superheader style.

```
\compatglossarystyle{superheader}{%
 \csuse{@glscompstyle@super}%
}%
```

Backward compatible superheaderborder style.

```
\compatglossarystyle{superheaderborder}{%
 \csuse{@glscompstyle@super}%
}%
```

Backward compatible super3col style.

```
\compatglossarystyle{super3col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##6\\}%
}%
```

Backward compatible super3colborder style.

```
\compatglossarystyle{super3colborder}{%
 \csuse{@glscompstyle@super3col}%
}%
```

Backward compatible super3colheader style.

```
\compatglossarystyle{super3colheader}{%
 \csuse{@glscompstyle@super3col}%
}%
```

Backward compatible super3colheaderborder style.

```
\compatglossarystyle{super3colheaderborder}{%
 \csuse{@glscompstyle@super3col}%
}%
```

Backward compatible super4col style.

```
\compatglossarystyle{super4col}{%
  \renewcommand*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
}%
```

Backward compatible super4colheader style.

```
\compatglossarystyle{super4colheader}{%
 \csuse{@glscompstyle@super4col}%
}%
```

Backward compatible super4colborder style.

```
\compatglossarystyle{super4colborder}{%
 \csuse{@glscompstyle@super4col}%
}%
```

Backward compatible super4colheaderborder style.

```
\compatglossarystyle{super4colheaderborder}{%
 \csuse{@glscompstyle@super4col}%
}%
```

Backward compatible altsuper4col style.

```
\compatglossarystyle{altsuper4col}{%
 \csuse{@glscompstyle@super4col}%
}%
```

Backward compatible altsuper4colheader style.

```
\compatglossarystyle{altsuper4colheader}{%
 \csuse{@glscompstyle@super4col}%
}%
```

Backward compatible altsuper4colborder style.

```
\compatglossarystyle{altsuper4colborder}{%
 \csuse{@glscompstyle@super4col}%
}%
```

Backward compatible altsuper4colheaderborder style.

```
\compatglossarystyle{altsuper4colheaderborder}{%
 \csuse{@glscompstyle@super4col}%
}%
```

5 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibilty support in glossary entries. See the documentation for further details about accessibility support.

```
\NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
\ProvidesPackage{glossaries-accsupp}[2018/06/01 v4.40 (NLCT)
  Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
\ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
\@ifpackageloaded{glossaries-extra}
{%
```

If the accsupp option was used, `\@glsxtr@doaccsupp` will have been set, otherwise it will be empty.

```
  \ifx\@glsxtr@doaccsupp\empty
   \GlossariesWarning{The ‘glossaries-accsupp’
   package has been loaded\MessageBreak
   after the ‘glossaries-extra’ package. This\MessageBreak
   can cause a failure to integrate both packages. \MessageBreak
   Either use the ‘accsupp’ option when you load\MessageBreak
   ‘glossaries-extra’ or load ‘glossaries-accsupp’\MessageBreak
   before loading ‘glossaries-extra’}%
  \fi
}
{}
```

tibleglossentry Override style compatibility macros:

```
\def\compatibleglossentry#1#2{%
  \toks@{#2}%
  \protected@edef\@do@glossentry{%
    \noexpand\accsuppglossaryentryfield{#1}%
    {\noexpand\glsnamefont
       {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
    {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
    {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
    {\the\toks@}%
  }%
  \@do@glossentry
}
```

lesubglossentry

```
\def\compatiblesubglossentry#1#2#3{%
  \toks@{#3}%
  \protected@edef\@do@subglossentry{%
    \noexpand\accsuppglossarysubentryfield{\number#1}%
    {#2}%
    {\noexpand\glsnamefont
      {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
    {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
    {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
    {\the\toks@}%
  }%
  \@do@subglossentry
}
```

Required packages:

```
\RequirePackage{glossaries}
\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:

```
\define@key{glossentry}{access}{%
  \def\@glo@access{#1}%
}
```

textaccess The replacement text corresponding to the text key:

```
\define@key{glossentry}{textaccess}{%
  \def\@glo@textaccess{#1}%
}
```

firstaccess The replacement text corresponding to the first key:

```
\define@key{glossentry}{firstaccess}{%
  \def\@glo@firstaccess{#1}%
}
```

pluralaccess The replacement text corresponding to the plural key:

```
\define@key{glossentry}{pluralaccess}{%
  \def\@glo@pluralaccess{#1}%
}
```

rstpluralaccess The replacement text corresponding to the firstplural key:

```
\define@key{glossentry}{firstpluralaccess}{%
  \def\@glo@firstpluralaccess{#1}%
}
```

symbolaccess The replacement text corresponding to the symbol key:

```
\define@key{glossentry}{symbolaccess}{%
  \def\@glo@symbolaccess{#1}%
}
```

bolpluralaccess The replacement text corresponding to the symbolplural key:

```
\define@key{glossentry}{symbolpluralaccess}{%
  \def\@glo@symbolpluralaccess{#1}%
}
```

scriptionaccess The replacement text corresponding to the description key:

```
\define@key{glossentry}{descriptionaccess}{%
  \def\@glo@descaccess{#1}%
}
```

ionpluralaccess The replacement text corresponding to the descriptionplural key:

```
\define@key{glossentry}{descriptionpluralaccess}{%
  \def\@glo@descpluralaccess{#1}%
}
```

shortaccess The replacement text corresponding to the short key:

```
\define@key{glossentry}{shortaccess}{%
  \def\@glo@shortaccess{#1}%
}
```

ortpluralaccess The replacement text corresponding to the shortplural key:

```
\define@key{glossentry}{shortpluralaccess}{%
  \def\@glo@shortpluralaccess{#1}%
}
```

longaccess The replacement text corresponding to the long key:

```
\define@key{glossentry}{longaccess}{%
  \def\@glo@longaccess{#1}%
}
```

ongpluralaccess The replacement text corresponding to the longplural key:

```
\define@key{glossentry}{longpluralaccess}{%
  \def\@glo@longpluralaccess{#1}%
}
```

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`:

```
\appto\@gls@keymap{,%
  {access}{access},%
  {textaccess}{textaccess},%
  {firstaccess}{firstaccess},%
  {pluralaccess}{pluralaccess},%
  {firstpluralaccess}{firstpluralaccess},%
  {symbolaccess}{symbolaccess},%
  {symbolpluralaccess}{symbolpluralaccess},%
  {descaccess}{descaccess},%
  {descpluralaccess}{descpluralaccess},%
  {shortaccess}{shortaccess},%
  {shortpluralaccess}{shortpluralaccess},%
  {longaccess}{longaccess},%
  {longpluralaccess}{longpluralaccess}%
}
```

`\@gls@noaccess` Indicates that no replacement text has been provided.

```
\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):

```
\let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
\renewcommand*{\@newglossaryentryprehook}{%
  \@gls@oldnewglossaryentryprehook
  \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```
  \def\@glo@textaccess{\@glo@access}%
  \def\@glo@firstaccess{\@glo@access}%
  \def\@glo@pluralaccess{\@glo@textaccess}%
  \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
  \def\@glo@symbolaccess{\relax}%
  \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
  \def\@glo@descaccess{\relax}%
  \def\@glo@descpluralaccess{\@glo@descaccess}%
  \def\@glo@shortaccess{\relax}%
  \def\@glo@shortpluralaccess{\@glo@shortaccess}%
  \def\@glo@longaccess{\relax}%
  \def\@glo@longpluralaccess{\@glo@longaccess}%
}
```

Add to the end hook:

```
\let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
\renewcommand*{\@newglossaryentryposthook}{%
  \@gls@oldnewglossaryentryposthook
```

Store the access information:

```
  \expandafter
    \protected@xdef\csname glo@\@glo@label @access\endcsname{%
      \@glo@access}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
      \@glo@textaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
      \@glo@firstaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
      \@glo@pluralaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
      \@glo@firstpluralaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
      \@glo@symbolaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
      \@glo@symbolpluralaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
      \@glo@descaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
      \@glo@descpluralaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
      \@glo@shortaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
      \@glo@shortpluralaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
      \@glo@longaccess}%
  \expandafter
    \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
      \@glo@longpluralaccess}%
}
```

5.2 Accessing Replacement Text

\glsentryaccess Get the value of the access key for the entry with the given label:

```
\newcommand*{\glsentryaccess}[1]{%
  \@gls@entry@field{#1}{access}%
}
```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
\newcommand*{\glsentrytextaccess}[1]{%
  \@gls@entry@field{#1}{textaccess}%
}
```

ntryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
\newcommand*{\glsentryfirstaccess}[1]{%
  \@gls@entry@field{#1}{firstaccess}%
}
```

trypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrypluralaccess}[1]{%
  \@gls@entry@field{#1}{pluralaccess}%
}
```

rstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentryfirstpluralaccess}[1]{%
  \csname glo@#1@firstpluralaccess\endcsname
}
```

trysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
\newcommand*{\glsentrysymbolaccess}[1]{%
  \@gls@entry@field{#1}{symbolaccess}%
}
```

bolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrysymbolpluralaccess}[1]{%
  \@gls@entry@field{#1}{symbolpluralaccess}%
}
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
\newcommand*{\glsentrydescaccess}[1]{%
  \@gls@entry@field{#1}{descaccess}%
}
```

escpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrydescpluralaccess}[1]{%
  \@gls@entry@field{#1}{descaccess}%
}
```

ntryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
\newcommand*{\glsentryshortaccess}[1]{%
  \@gls@entry@field{#1}{shortaccess}%
}
```

ortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentryshortpluralaccess}[1]{%
  \@gls@entry@field{#1}{shortpluralaccess}%
}
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
\newcommand*{\glsentrylongaccess}[1]{%
  \@gls@entry@field{#1}{longaccess}%
}
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrylongpluralaccess}[1]{%
  \@gls@entry@field{#1}{longpluralaccess}%
}
```

\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.)

```
\newcommand*{\glsaccsupp}[2]{%
  \BeginAccSupp{ActualText={#1}}#2\EndAccSupp{}%
}
```

\xglsaccsupp Fully expands replacement text before calling `\glsaccsupp`

```
\newcommand*{\xglsaccsupp}[2]{%
   \protected@edef\@gls@replacementtext{#1}%
   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
}
```

@access@display

```
\newcommand*{\@gls@access@display}[2]{%
  \protected@edef\@glo@access{#2}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
\DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentryaccess{#2}}%
}
```

xtaccessdisplay As above but for the textaccess replacement text.

```
\DeclareRobustCommand*{\glstextaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
}
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
\DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
}
```

staccessdisplay As above but for the firstaccess replacement text.

```
\DeclareRobustCommand*{\glsfirstaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
}
```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```
\DeclareRobustCommand*{\glsfirstpluralaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
}
```

olaccessdisplay As above but for the symbolaccess replacement text.

```
\DeclareRobustCommand*{\glssymbolaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
}
```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```
\DeclareRobustCommand*{\glssymbolpluralaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
}
```

onaccessdisplay As above but for the descriptionaccess replacement text.

```
\DeclareRobustCommand*{\glsdescriptionaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
}
```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
\DeclareRobustCommand*{\glsdescriptionpluralaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
}
```

rtaccessdisplay As above but for the shortaccess replacement text.

```
\DeclareRobustCommand*{\glsshortaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
}
```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```
\DeclareRobustCommand*{\glsshortpluralaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
}
```

ngaccessdisplay As above but for the longaccess replacement text.

```
\DeclareRobustCommand*{\glslongaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
}
```

alaccessdisplay As above but for the longpluralaccess replacement text.

```
\DeclareRobustCommand*{\glslongpluralaccessdisplay}[2]{%
  \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
}
```

lsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
\DeclareRobustCommand*{\glsaccessdisplay}[3]{%
  \@ifundefined{gls#1accessdisplay}%
  {%
    \PackageError{glossaries-accsupp}{No accessibility support
     for key ‘#1’}{}%
  }%
  {%
    \csname gls#1accessdisplay\endcsname{#2}{#3}%
  }%
}
```

efault@entryfmt Redefine the default entry format to use accessibility information

```
\renewcommand*{\@@gls@default@entryfmt}[2]{%
  \ifdefempty\glscustomtext
  {%
    \glsifplural
    {%
```

Plural form

```
      \glscapscase
      {%
```

Don't adjust case

```
        \ifglsused\glslabel
        {%
```

Subsequent use

```
          #2{\glspluralaccessdisplay
               {\glsentryplural{\glslabel}}{\glslabel}}%
            {\glsdescriptionpluralaccessdisplay
               {\glsentrydescplural{\glslabel}}{\glslabel}}%
            {\glssymbolpluralaccessdisplay
               {\glsentrysymbolplural{\glslabel}}{\glslabel}}
            {\glsinsert}%
        }%
        {%
```

First use

```
          #1{\glsfirstpluralaccessdisplay
               {\glsentryfirstplural{\glslabel}}{\glslabel}}%
            {\glsdescriptionpluralaccessdisplay
               {\glsentrydescplural{\glslabel}}{\glslabel}}%
            {\glssymbolpluralaccessdisplay
```

```
               {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
            {\glsinsert}%
        }%
      }%
      {%
```

Make first letter upper case

```
        \ifglsused\glslabel
        {%
```

Subsequent use.

```
          #2{\glspluralaccessdisplay
              {\Glsentryplural{\glslabel}}{\glslabel}}%
            {\glsdescriptionpluralaccessdisplay
              {\glsentrydescplural{\glslabel}}{\glslabel}}%
            {\glssymbolpluralaccessdisplay
              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
            {\glsinsert}%
        }%
        {%
```

First use

```
          #1{\glsfirstpluralaccessdisplay
               {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
            {\glsdescriptionpluralaccessdisplay
               {\glsentrydescplural{\glslabel}}{\glslabel}}%
            {\glssymbolpluralaccessdisplay
               {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
            {\glsinsert}%
        }%
      }%
      {%
```

Make all upper case

```
        \ifglsused\glslabel
        {%
```

Subsequent use

```
          \MakeUppercase{%
            #2{\glspluralaccessdisplay
                {\glsentryplural{\glslabel}}{\glslabel}}%
              {\glsdescriptionpluralaccessdisplay
                {\glsentrydescplural{\glslabel}}{\glslabel}}%
              {\glssymbolpluralaccessdisplay
                {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
              {\glsinsert}}%
        }%
        {%
```

First use

```
          \MakeUppercase{%
            #1{\glsfirstpluralaccessdisplay
            {\glsentryfirstplural{\glslabel}}{\glslabel}}%
          {\glsdescriptionpluralaccessdisplay
            {\glsentrydescplural{\glslabel}}{\glslabel}}%
          {\glssymbolpluralaccessdisplay
            {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
          {\glsinsert}}%
    }%
  }%
}%
{%
```

Singular form

```
\glscapscase
{%
```

Don't adjust case

```
    \ifglsused\glslabel
    {%
```

Subsequent use

```
      #2{\glstextaccessdisplay
           {\glsentrytext{\glslabel}}{\glslabel}}%
        {\glsdescriptionaccessdisplay
           {\glsentrydesc{\glslabel}}{\glslabel}}%
        {\glssymbolaccessdisplay
           {\glsentrysymbol{\glslabel}}{\glslabel}}%
        {\glsinsert}%
    }%
    {%
```

First use

```
      #1{\glsfirstaccessdisplay
          {\glsentryfirst{\glslabel}}{\glslabel}}%
        {\glsdescriptionaccessdisplay
          {\glsentrydesc{\glslabel}}{\glslabel}}%
        {\glssymbolaccessdisplay
          {\glsentrysymbol{\glslabel}}{\glslabel}}%
        {\glsinsert}%
    }%
  }%
  {%
```

Make first letter upper case

```
    \ifglsused\glslabel
    {%
```

Subsequent use

```
      #2{\glstextaccessdisplay
           {\Glsentrytext{\glslabel}}{\glslabel}}%
        {\glsdescriptionaccessdisplay
           {\glsentrydesc{\glslabel}}{\glslabel}}%
        {\glssymbolaccessdisplay
```

```
                  {\glsentrysymbol{\glslabel}}{\glslabel}}%
               {\glsinsert}%
           }%
           {%
```

First use

```
             #1{\glsfirstaccessdisplay
                 {\Glsentryfirst{\glslabel}}{\glslabel}}%
               {\glsdescriptionaccessdisplay
                 {\glsentrydesc{\glslabel}}{\glslabel}}%
               {\glssymbolaccessdisplay
                 {\glsentrysymbol{\glslabel}}{\glslabel}}%
               {\glsinsert}%
           }%
         }%
         {%
```

Make all upper case

```
           \ifglsused\glslabel
           {%
```

Subsequent use

```
             \MakeUppercase{%
               #2{\glstextaccessdisplay
                   {\glsentrytext{\glslabel}}{\glslabel}}%
                 {\glsdescriptionaccessdisplay
                   {\glsentrydesc{\glslabel}}{\glslabel}}%
                 {\glssymbolaccessdisplay
                   {\glsentrysymbol{\glslabel}}{\glslabel}}%
                 {\glsinsert}}%
           }%
           {%
```

First use

```
             \MakeUppercase{%
               #1{\glsfirstaccessdisplay
                   {\glsentryfirst{\glslabel}}{\glslabel}}%
                 {\glsdescriptionaccessdisplay
                   {\glsentrydesc{\glslabel}}{\glslabel}}%
                 {\glssymbolaccessdisplay
                   {\glsentrysymbol{\glslabel}}{\glslabel}}%
                 {\glsinsert}}%
           }%
         }%
       }%
     }%
     {%
```

Custom text provided in \glsdisp

```
       \ifglsused{\glslabel}%
       {%
```

Subsequent use

```
      #2{\glscustomtext}%
        {\glsdescriptionaccessdisplay
          {\glsentrydesc{\glslabel}}{\glslabel}}%
        {\glssymbolaccessdisplay
          {\glsentrysymbol{\glslabel}}{\glslabel}}%
        {\glsinsert}%
    }%
    {%
```

First use

```
      #1{\glscustomtext}%
        {\glsdescriptionaccessdisplay
          {\glsentrydesc{\glslabel}}{\glslabel}}%
        {\glssymbolaccessdisplay
          {\glsentrysymbol{\glslabel}}{\glslabel}}%
        {\glsinsert}%
    }%
  }%
}
```

\glsgenentryfmt Redefine to use accessibility information.

```
\renewcommand*{\glsgenentryfmt}{%
  \ifdefempty\glscustomtext
  {%
    \glsifplural
    {%
```

Plural form

```
      \glscapscase
      {%
```

Don't adjust case

```
        \ifglsused\glslabel
        {%
```

Subsequent use

```
          \glspluralaccessdisplay
               {\glsentryplural{\glslabel}}{\glslabel}%
          \glsinsert
        }%
        {%
```

First use

```
          \glsfirstpluralaccessdisplay
              {\glsentryfirstplural{\glslabel}}{\glslabel}%
          \glsinsert
        }%
      }%
      {%
```

Make first letter upper case

```
        \ifglsused\glslabel
        {%
```

Subsequent use.

```
          \glspluralaccessdisplay
              {\Glsentryplural{\glslabel}}{\glslabel}%
          \glsinsert
        }%
        {%
```

First use

```
          \glsfirstpluralaccessdisplay
              {\Glsentryfirstplural{\glslabel}}{\glslabel}%
          \glsinsert
        }%
      }%
      {%
```

Make all upper case

```
        \ifglsused\glslabel
        {%
```

Subsequent use

```
           \glspluralaccessdisplay
              {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
              {\glslabel}%
           \mfirstucMakeUppercase{\glsinsert}%
        }%
        {%
```

First use

```
          \glsfirstpluralacessdisplay
             {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
             {\glslabel}%
          \mfirstucMakeUppercase{\glsinsert}%
        }%
      }%
    }%
    {%
```

Singular form

```
      \glscapscase
      {%
```

Don't adjust case

```
        \ifglsused\glslabel
        {%
```

Subsequent use

```
          \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
          \glsinsert
          }%
          {%
```

First use

```
            \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
            \glsinsert
          }%
        }%
        {%
```

Make first letter upper case

```
          \ifglsused\glslabel
          {%
```

Subsequent use

```
             \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
             \glsinsert
          }%
          {%
```

First use

```
            \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
            \glsinsert
          }%
        }%
        {%
```

Make all upper case

```
          \ifglsused\glslabel
          {%
```

Subsequent use

```
            \glstextaccessdisplay
              {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
            \mfirstucMakeUppercase{\glsinsert}%
          }%
          {%
```

First use

```
            \glsfirstaccessdisplay
              {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
            \mfirstucMakeUppercase{\glsinsert}%
          }%
        }%
      }%
    }%
    {%
```

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.

```
      \glscustomtext\glsinsert
    }%
}
```

\glsgenacfmt Redefine to include accessibility information.

```
\renewcommand*{\glsgenacfmt}{%
  \ifdefempty\glscustomtext
  {%
    \ifglsused\glslabel
    {%
```

Subsequent use:

```
      \glsifplural
      {%
```

Subsequent plural form:

```
        \glscapscase
        {%
```

Subsequent plural form, don't adjust case:

```
          \acronymfont
           {\glsshortpluralaccessdisplay
              {\glsentryshortpl{\glslabel}}{\glslabel}}%
          \glsinsert
        }%
        {%
```

Subsequent plural form, make first letter upper case:

```
          \acronymfont
           {\glsshortpluralaccessdisplay
              {\Glsentryshortpl{\glslabel}}{\glslabel}}%
          \glsinsert
        }%
        {%
```

Subsequent plural form, all caps:

```
          \mfirstucMakeUppercase
          {\acronymfont
           {\glsshortpluralaccessdisplay
              {\glsentryshortpl{\glslabel}}{\glslabel}}%
          \glsinsert}%
        }%
      }%
      {%
```

Subsequent singular form

```
        \glscapscase
        {%
```

Subsequent singular form, don't adjust case:

```
          \acronymfont
           {\glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
          \glsinsert
        }%
        {%
```

Subsequent singular form, make first letter upper case:

```
          \acronymfont
           {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
          \glsinsert
        }%
        {%
```

Subsequent singular form, all caps:

```
          \mfirstucMakeUppercase
            {\acronymfont{%
              \glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
             \glsinsert}%
        }%
      }%
    }%
    {%
```

First use:

```
      \glsifplural
      {%
```

First use plural form:

```
        \glscapscase
        {%
```

First use plural form, don't adjust case:

```
          \genplacrfullformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use plural form, make first letter upper case:

```
          \Genplacrfullformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use plural form, all caps:

```
          \mfirstucMakeUppercase
            {\genplacrfullformat{\glslabel}{\glsinsert}}%
        }%
      }%
      {%
```

First use singular form

```
        \glscapscase
        {%
```

First use singular form, don't adjust case:

```
          \genacrfullformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use singular form, make first letter upper case:

```
          \Genacrfullformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use singular form, all caps:

```
          \mfirstucMakeUppercase
           {\genacrfullformat{\glslabel}{\glsinsert}}%
        }%
      }%
    }%
  }%
  {%
```

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.

```
    \glscustomtext
  }%
}
```

enacrfullformat

Redefine to include accessibility information.

```
\renewcommand*{\genacrfullformat}[2]{%
   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
}
```

enacrfullformat

Redefine to include accessibility information.

```
\renewcommand*{\Genacrfullformat}[2]{%
   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
}
```

placrfullformat

Redefine to include accessibility information.

```
\renewcommand*{\genplacrfullformat}[2]{%
   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
   (\glsshortpluralaccessdisplay
      {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
}
```

placrfullformat

Redefine to include accessibility information.

```
\renewcommand*{\Genplacrfullformat}[2]{%
   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
   (\glsshortpluralaccessdisplay
      {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
}
```

\@acrshort

```
\def\@acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax

    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
    }%
```

Call `\@gls@link`

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%

  \glspostlinkhook
}
```

`\@Acrshort`

```
\def\@Acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax

    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
    }%
```

Call `\@gls@link`

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%

  \glspostlinkhook
}
```

`\@ACRshort`

```
\def\@ACRshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax

    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsshortaccessdisplay
          {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
    }%
```

Call `\@gls@link`

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
```

```
  \glspostlinkhook
}
```

\@acrlong

```
\def\@acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax
```

```
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
    }%
```

Call `\@gls@link`

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
```

```
  \glspostlinkhook
}
```

\@Acrlong

```
\def\@Acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax
```

```
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
    }%
```

Call `\@gls@link`

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
```

```
  \glspostlinkhook
}
```

\@ACRlong

```
\def\@ACRlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glslongaccessdisplay{%
        \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
    }%
```

Call `\@gls@link`

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
```

```
  \glspostlinkhook
}
```

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.

```
\renewcommand*{\glossentryname}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
  }%
}
```

```
\renewcommand*{\glossentryname}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
  }%
}
```

```
\renewcommand*{\glossentrydesc}[1]{%
  \glsdoifexists{#1}%
  {%
     \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
  }%
}
```

```
\renewcommand*{\Glossentrydesc}[1]{%
  \glsdoifexists{#1}%
  {%
     \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
  }%
}
```

```
\renewcommand*{\glossentrysymbol}[1]{%
  \glsdoifexists{#1}%
  {%
     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
  }%
}
```

```
\renewcommand*{\Glossentrysymbol}[1]{%
  \glsdoifexists{#1}%
  {%
     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
  }%
}
```

ssaryentryfield

```
\newcommand*{\accsuppglossaryentryfield}[5]{%
  \glossaryentryfield{#1}%
  {\glsnameaccessdisplay{#2}{#1}}%
  {\glsdescriptionaccessdisplay{#3}{#1}}%
  {\glssymbolaccessdisplay{#4}{#1}}{#5}%
}
```

rysubentryfield

```
\newcommand*{\accsuppglossarysubentryfield}[6]{%
  \glossarysubentryfield{#1}{#2}%
  {\glsnameaccessdisplay{#3}{#2}}%
  {\glsdescriptionaccessdisplay{#4}{#2}}%
  {\glssymbolaccessdisplay{#5}{#2}}{#6}%
}
```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short ⟨*long*⟩ (⟨*short*⟩) acronym style.

```
\renewacronymstyle{long-short}%
{%
```

Check for long form in case this is a mixed glossary.

```
  \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
}%
{%
  \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
  \renewcommand*{\genacrfullformat}[2]{%
   \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
   (\glsshortaccessdisplay
      {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
  }%
  \renewcommand*{\Genacrfullformat}[2]{%
   \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
   (\glsshortaccessdisplay
      {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
  }%
  \renewcommand*{\genplacrfullformat}[2]{%
   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
   (\glsshortpluralaccessdisplay
      {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
  }%
  \renewcommand*{\Genplacrfullformat}[2]{%
   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
   (\glsshortpluralaccessdisplay
      {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
  }%
  \renewcommand*{\acronymentry}[1]{%
    \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
  \renewcommand*{\acronymsort}[2]{##1}%
  \renewcommand*{\acronymfont}[1]{##1}%
  \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
 }
```

`short-long` ⟨*short*⟩ (⟨*long*⟩) acronym style.

```
\renewacronymstyle{short-long}%
{%
```

Check for long form in case this is a mixed glossary.

```
 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
}%
{%
 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
 \renewcommand*{\genacrfullformat}[2]{%
  \glsshortaccessdisplay
    {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
  (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
 }%
 \renewcommand*{\Genacrfullformat}[2]{%
  \glsshortaccessdisplay
     {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
  (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
 }%
 \renewcommand*{\genplacrfullformat}[2]{%
  \glsshortpluralaccessdisplay
    {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
  (\glslongpluralaccessdisplay
    {\glsentrylongpl{##1}}{##1})%
 }%
 \renewcommand*{\Genplacrfullformat}[2]{%
  \glsshortpluralaccessdisplay
   {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space
   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
  }%
  \renewcommand*{\acronymentry}[1]{%
    \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
  \renewcommand*{\acronymsort}[2]{##1}%
  \renewcommand*{\acronymfont}[1]{##1}%
  \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
}
```

`long-short-desc` ⟨*long*⟩ ({⟨*short*⟩}) acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{long-short-desc}%
{%
  \GlsUseAcrEntryDispStyle{long-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-short}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

`g-sc-short-desc` ⟨*long*⟩ (\textsc{⟨*short*⟩}) acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{long-sc-short-desc}%
{%
  \GlsUseAcrEntryDispStyle{long-sc-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-sc-short}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

`g-sm-short-desc` ⟨*long*⟩ (\textsmaller{⟨*short*⟩}) acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{long-sm-short-desc}%
{%
  \GlsUseAcrEntryDispStyle{long-sm-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-sm-short}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

`short-long-desc` ⟨*short*⟩ ({⟨*long*⟩}) acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{short-long-desc}%
{%
  \GlsUseAcrEntryDispStyle{short-long}%
}%
{%
  \GlsUseAcrStyleDefs{short-long}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

`short-long-desc` ⟨*long*⟩ (`\textsc{`⟨*short*⟩`}`) acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{sc-short-long-desc}%
{%
  \GlsUseAcrEntryDispStyle{sc-short-long}%
}%
{%
  \GlsUseAcrStyleDefs{sc-short-long}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

`short-long-desc` ⟨*long*⟩ (`\textsmaller{`⟨*short*⟩`}`) acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{sm-short-long-desc}%
{%
  \GlsUseAcrEntryDispStyle{sm-short-long}%
}%
{%
  \GlsUseAcrStyleDefs{sm-short-long}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

dua ⟨*long*⟩ only acronym style.

```
\renewacronymstyle{dua}%
{%
```

Check for long form in case this is a mixed glossary.

```
  \ifdefempty\glscustomtext
  {%
    \ifglshaslong{\glslabel}%
    {%
      \glsifplural
      {%
```

Plural form:

```
        \glscapscase
        {%
```

Plural form, don't adjust case:

```
          \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
          \glsinsert
        }%
        {%
```

Plural form, make first letter upper case:

```
          \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
          \glsinsert
        }%
        {%
```

Plural form, all caps:

```
          \glslongpluralaccessdisplay
            {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
          \mfirstucMakeUppercase{\glsinsert}%
        }%
      }%
      {%
```

Singular form

```
        \glscapscase
        {%
```

Singular form, don't adjust case:

```
          \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
        }%
        {%
```

Subsequent singular form, make first letter upper case:

```
          \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
        }%
        {%
```

Subsequent singular form, all caps:

```
           \glslongaccessdisplay
            {\mfirstucMakeUppercase
               {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
           \mfirstucMakeUppercase{\glsinsert}%
         }%
       }%
     }%
     {%
```

Not an acronym:

```
       \glsgenentryfmt
     }%
   }%
   {\glscustomtext\glsinsert}%
}%
{%
   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
   \renewcommand*{\acrfullfmt}[3]{%
     \glslink[##1]{##2}{%
       \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
       (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
   \renewcommand*{\Acrfullfmt}[3]{%
     \glslink[##1]{##2}{%
       \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
       (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
   \renewcommand*{\ACRfullfmt}[3]{%
     \glslink[##1]{##2}{%
       \glslongaccessdisplay
         {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
       (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}}%
   \renewcommand*{\acrfullplfmt}[3]{%
     \glslink[##1]{##2}{%
       \glslongpluralaccessdisplay
         {\glsentrylongpl{##2}}{##2}##3\space
       (\glsshortpluralaccessdisplay
         {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
   \renewcommand*{\Acrfullplfmt}[3]{%
     \glslink[##1]{##2}{%
       \glslongpluralaccessdisplay
         {\Glsentrylongpl{##2}}{##2}##3\space
       (\glsshortpluralaccessdisplay
         {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
   \renewcommand*{\ACRfullplfmt}[3]{%
     \glslink[##1]{##2}{%
       \glslongpluralaccessdisplay
          {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
       (\glsshortpluralaccessdisplay
          {\acronymfont{\glsentryshortpl{##2}}}{##2})}}}%
   \renewcommand*{\glsentryfull}[1]{%
```

```
  \glslongaccessdisplay{\glsentrylong{##1}}\space
  (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
 }%
 \renewcommand*{\Glsentryfull}[1]{%
  \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
  (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
 }%
 \renewcommand*{\glsentryfullpl}[1]{%
  \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
  (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
 }%
 \renewcommand*{\Glsentryfullpl}[1]{%
  \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
  (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
 }%
 \renewcommand*{\acronymentry}[1]{%
   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
 \renewcommand*{\acronymsort}[2]{##1}%
 \renewcommand*{\acronymfont}[1]{##1}%
 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
}
```

`dua-desc` ⟨*long*⟩ only acronym style with user-supplied description.

```
\renewacronymstyle{dua-desc}%
{%
 \GlsUseAcrEntryDispStyle{dua}%
}%
{%
 \GlsUseAcrStyleDefs{dua}%
 \renewcommand*{\GenericAcronymFields}{}%
 \renewcommand*{\acronymentry}[1]{%
  \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
 \renewcommand*{\acronymsort}[2]{##2}%
}%
```

`footnote` ⟨*short*⟩`\footnote{`⟨*long*⟩`}` acronym style.

```
\renewacronymstyle{footnote}%
{%
```

Check for long form in case this is a mixed glossary.

```
 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
}%
{%
 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
 \glshyperfirstfalse
 \renewcommand*{\genacrfullformat}[2]{%
  \glsshortaccessdisplay
   {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%
  \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
 }%
 \renewcommand*{\Genacrfullformat}[2]{%
  \glsshortaccessdisplay
   {\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
  \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
 }%
 \renewcommand*{\genplacrfullformat}[2]{%
  \glsshortpluralaccessdisplay
   {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2%
  \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
 }%
 \renewcommand*{\Genplacrfullformat}[2]{%
  \glsshortpluralaccessdisplay
   {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2%
  \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
 }%
 \renewcommand*{\acronymentry}[1]{%
  \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
 \renewcommand*{\acronymsort}[2]{##1}%
 \renewcommand*{\acronymfont}[1]{##1}%
 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
```

Don’t use footnotes for \acrfull:

```
\renewcommand*{\acrfullfmt}[3]{%
  \glslink[##1]{##2}{%
    \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}##3\space
    (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
\renewcommand*{\Acrfullfmt}[3]{%
  \glslink[##1]{##2}{%
    \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
    (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
\renewcommand*{\ACRfullfmt}[3]{%
  \glslink[##1]{##2}{%
    \glsshortaccessdisplay
       {\mfirstucMakeUppercase
         {\acronymfont{\glsentryshort{##2}}}{##2}##3\space
    (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
\renewcommand*{\acrfullplfmt}[3]{%
  \glslink[##1]{##2}{%
    \glsshortpluralaccessdisplay
       {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
    (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}%
\renewcommand*{\Acrfullplfmt}[3]{%
  \glslink[##1]{##2}{%
    \glsshortpluralaccessdisplay
      {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
    (\glslongpluralaccessdisplay{\glsentrylongpl{##2}})}}%
\renewcommand*{\ACRfullplfmt}[3]{%
  \glslink[##1]{##2}{%
```

```
      \glsshortpluralaccessdisplay
        {\mfirstucMakeUppercase
           {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
      (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
```

Similarly for `\glsentryfull` etc:

```
\renewcommand*{\glsentryfull}[1]{%
   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
    (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
\renewcommand*{\Glsentryfull}[1]{%
   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
\renewcommand*{\glsentryfullpl}[1]{%
   \glsshortpluralaccessdisplay
     {\acronymfont{\glsentryshortpl{##1}}}{##1}\space
     (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
\renewcommand*{\Glsentryfullpl}[1]{%
   \glsshortpluralaccessdisplay
      {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
}
```

`footnote-sc` `\textsc{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style.

```
\renewacronymstyle{footnote-sc}%
{%
  \GlsUseAcrEntryDispStyle{footnote}%
}%
{%
  \GlsUseAcrStyleDefs{footnote}%
  \renewcommand{\acronymentry}[1]{%
     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
  \renewcommand{\acronymfont}[1]{\textsc{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
}%
```

`footnote-sm` `\textsmaller{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style.

```
\renewacronymstyle{footnote-sm}%
{%
  \GlsUseAcrEntryDispStyle{footnote}%
}%
{%
  \GlsUseAcrStyleDefs{footnote}%
  \renewcommand{\acronymentry}[1]{%
    \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
  \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
}%
```

`footnote-desc` ⟨*short*⟩`\footnote{`⟨*long*⟩`}` acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{footnote-desc}%
{%
  \GlsUseAcrEntryDispStyle{footnote}%
}%
{%
  \GlsUseAcrStyleDefs{footnote}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

ootnote-sc-desc `\textsc{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{footnote-sc-desc}%
{%
  \GlsUseAcrEntryDispStyle{footnote-sc}%
}%
{%
  \GlsUseAcrStyleDefs{footnote-sc}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

ootnote-sm-desc `\textsmaller{`⟨*short*⟩`}\footnote{`⟨*long*⟩`}` acronym style that has an accompanying description (which the user needs to supply).

```
\renewacronymstyle{footnote-sm-desc}%
{%
  \GlsUseAcrEntryDispStyle{footnote-sm}%
}%
{%
  \GlsUseAcrStyleDefs{footnote-sm}%
  \renewcommand*{\GenericAcronymFields}{}%
  \renewcommand*{\acronymsort}[2]{##2}%
  \renewcommand*{\acronymentry}[1]{%
    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
}
```

Use `\newacronymhook` to modify the key list to set the access text to the long version by default.

```
\renewcommand*{\newacronymhook}{%
  \edef\@gls@keylist{shortaccess=\the\glslongtok,%
     \the\glskeylisttok}%
  \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
}
```

ltNewAcronymDef Modify default style to use access text:

```
\renewcommand*{\DefaultNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\the\glsshorttok},%
      description={\the\glslongtok},%
      descriptionaccess=\relax,
      text={\the\glsshorttok},%
      access={\noexpand\@glo@textaccess},%
      sort={\the\glsshorttok},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      shortaccess={\the\glslongtok},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      first={\noexpand\glslongaccessdisplay
        {\the\glslongtok}{\the\glslabeltok}\space
        (\noexpand\glsshortaccessdisplay
          {\the\glsshorttok}{\the\glslabeltok})},%
      plural={\the\glsshorttok\acrpluralsuffix},%
      firstplural={\noexpand\glslongpluralaccessdisplay
        {\noexpand\@glo@longpl}{\the\glslabeltok}\space
        (\noexpand\glsshortpluralaccessdisplay
          {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
      firstaccess=\relax,
      firstpluralaccess=\relax,
      textaccess={\noexpand\@glo@shortaccess},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@descplural\gls@assign@descplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@descplural##1##2{%
    \@@gls@expand@field{##1}{descplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

teNewAcronymDef

```
\renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      sort={\the\glsshorttok},%
      text={\the\glsshorttok},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      shortaccess={\the\glslongtok},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      access={\noexpand\@glo@textaccess},%
      plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      symbol={\the\glslongtok},%
      symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      firstpluralaccess=\relax,
      textaccess={\noexpand\@glo@shortaccess},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

onNewAcronymDef

```
\renewcommand*{\DescriptionNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand
        \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
      access={\noexpand\@glo@textaccess},%
      sort={\the\glsshorttok},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      shortaccess={\the\glslongtok},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      first={\the\glslongtok},%
      firstaccess=\relax,
      firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      text={\the\glsshorttok},%
      textaccess={\the\glslongtok},%
      plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      symbol={\noexpand\@glo@text},%
      symbolaccess={\noexpand\@glo@textaccess},%
      symbolplural={\noexpand\@glo@plural},%
      firstpluralaccess=\relax,
      textaccess={\noexpand\@glo@shortaccess},%
      \the\glskeylisttok}%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

teNewAcronymDef

```
\renewcommand*{\FootnoteNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      sort={\the\glsshorttok},%
      text={\the\glsshorttok},%
      textaccess={\the\glslongtok},%
      access={\noexpand\@glo@textaccess},%
      plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\the\glslongtok},%
      descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@descplural\gls@assign@descplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@descplural##1##2{%
    \@@gls@expand@field{##1}{descplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@descplural\@org@gls@assign@descplural
}
```

llNewAcronymDef

```
\renewcommand*{\SmallNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      access={\noexpand\@glo@symbolaccess},%
      sort={\the\glsshorttok},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      shortaccess={\the\glslongtok},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      text={\noexpand\@glo@short},%
      textaccess={\noexpand\@glo@shortaccess},%
      plural={\noexpand\@glo@shortpl},%
      first={\the\glslongtok},%
      firstaccess=\relax,
      firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\noexpand\@glo@first},%
      descriptionplural={\noexpand\@glo@firstplural},%
      symbol={\the\glsshorttok},%
      symbolaccess={\the\glslongtok},%
      symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      \the\glskeylisttok
    }%
  }%
  \let\@org@gls@assign@firstpl\gls@assign@firstpl
  \let\@org@gls@assign@plural\gls@assign@plural
  \let\@org@gls@assign@descplural\gls@assign@descplural
  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
  \def\gls@assign@firstpl##1##2{%
    \@@gls@expand@field{##1}{firstpl}{##2}%
  }%
  \def\gls@assign@plural##1##2{%
    \@@gls@expand@field{##1}{plural}{##2}%
  }%
  \def\gls@assign@descplural##1##2{%
    \@@gls@expand@field{##1}{descplural}{##2}%
  }%
  \def\gls@assign@symbolplural##1##2{%
    \@@gls@expand@field{##1}{symbolplural}{##2}%
  }%
  \@do@newglossaryentry
  \let\gls@assign@firstpl\@org@gls@assign@firstpl
  \let\gls@assign@plural\@org@gls@assign@plural
  \let\gls@assign@descplural\@org@gls@assign@descplural
  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
}
```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
\newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
\newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
\newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
\newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

owglonameaccess

```
\newcommand*{\showglonameaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@access\endcsname
}
```

owglotextaccess

```
\newcommand*{\showglotextaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
}
```

glopluralaccess

```
\newcommand*{\showglopluralaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
}
```

wglofirstaccess

```
\newcommand*{\showglofirstaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
}
```

rstpluralaccess

```
\newcommand*{\showglofirstpluralaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
}
```

glosymbolaccess

```
\newcommand*{\showglosymbolaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
}
```

bolpluralaccess

```
\newcommand*{\showglosymbolpluralaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
}
```

owglodescaccess

```
\newcommand*{\showglodescaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
}
```

escpluralaccess

```
\newcommand*{\showglodescpluralaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
}
```

wgloshortaccess

```
\newcommand*{\showgloshortaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
}
```

ortpluralaccess

```
\newcommand*{\showgloshortpluralaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
}
```

owglolongaccess

```
\newcommand*{\showglolongaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
}
```

ongpluralaccess

```
\newcommand*{\showglolongpluralaccess}[1]{%
  \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
}
```

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.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-babel}[2018/06/01 v4.40 (NLCT)]
```

Load tracklang to obtain language settings.

```
\RequirePackage{tracklang}
\let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
  \AnyTrackedLanguages
  {%
    \ForEachTrackedDialect{\this@dialect}{%
      \IfTrackedLanguageFileExists{\this@dialect}%
      {glossaries-}% prefix
      {.ldf}%
      {%
        \RequireGlossariesLang{\CurrentTrackedTag}%
      }%
      {%
         \PackageWarningNoLine{glossaries}%
         {No language module detected for ‘\this@dialect’.\MessageBreak
          Language modules need to be installed separately.\MessageBreak
          Please check on CTAN for a bundle called\MessageBreak
         ‘glossaries-\CurrentTrackedLanguage’ or similar}%
      }%
    }%
  }%
  {}%
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-polyglossia}[2018/06/01 v4.40 (NLCT)]
```

Load tracklang to obtain language settings.

```
\RequirePackage{tracklang}
\let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
  \AnyTrackedLanguages
```

```
{%
  \ForEachTrackedDialect{\this@dialect}{%
    \IfTrackedLanguageFileExists{\this@dialect}%
    {glossaries-}% prefix
    {.ldf}%
    {%
      \RequireGlossariesLang{\CurrentTrackedTag}%
    }%
    {%
       \PackageWarningNoLine{glossaries}%
       {No language module detected for ‘\this@dialect’.\MessageBreak
        Language modules need to be installed separately.\MessageBreak
        Please check on CTAN for a bundle called\MessageBreak
       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
    }%
  }%
}%
{}%
```

Glossary

`makeindex` An indexing application. 9, 13, 29, 30, 179

`xindy` An flexible indexing application with multilingual support written in Perl. 9, 13, 29, 30, 179

Change History

1.01 (2007-05-17)

General: Added range facility in format key . . . 114

\writeist: Added spaces after \delimN and \delimR in ist file . . . 161

1.04 (2007-08-03)

General: Added \glstextformat . . . 98

1.05 (2007-08-10)

\glossarysection: added \@mkboth to \glossarysection . . . 41

\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key . . . 82

1.07 (2007-09-13)

\@gls@link: fixed bug caused by \theglsentrycounter setting the page number too soon . . . 112

\glsadd: fixed bug caused by \theglsentrycounter setting the page number too soon . . . 158

1.08 (2007-10-13)

General: Added babel support . . . 35

listgroup: changed listgroup style to use \glsgetgrouptitle . . . 274

altlistgroup: changed altlistgroup style to use \glsgetgrouptitle . . . 275

1.1 (2008-02-22)

\@glossarysection: numbered sections and auto label added . . . 42

\@gls@tmpb: changed \toksdef to \newtoks . . . 116

\@gls@toc: numberline added . . . 44

\@p@glossarysection: numbered sections and auto label added . . . 42

General: amsgen now loaded (\new@ifnextchar needed) . . . 4

translate: translate option added . . . 26

\setglossarysection: new . . . 42

numberedsection: numberedsection package option added . . . 8

numberline: numberline option added . . 7

1.12 (2008-03-08)

\@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol . . . 128

\@Glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol . . . 127

\@glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol . . . 126

General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) . . . 122

descriptionplural: new . . . 64

\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) . . . 82

descriptionplural support added . . . 82

symbolplural support added . . . 82

\Glsentrydescplural: New . . . 151

\glsentrydescplural: New . . . 151

\Glsentrysymbolplural: New . . . 152

\glsentrysymbolplural: New . . . 152

\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . 240

\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . 246

symbolplural: new . . . 65

1.13 (2008-05-10)
General: fixed bug that ignored 3rd parameter 129–137
\ACRfullpl: new 221
\Acrfullpl: new 221
\acrfullpl: new 220
\acrpluralsuffix: New 218
\gls@defglossaryentry: Changed default first value 82
Changed default firstplural value 82
Removed restriction on only using \newglossaryentry in the preamble 87
\newacronym: Removed restriction on only using \newacronym in the preamble 218
1.14 (2008-06-17)
\@gls@hypergroup: new 269
General: added nonumberlist key to \printglossary 205
added numberedsection key to \printglossary 204
\firstacronymfont: new 222
\glsautoprefix: new 7
\glsnavhyperlink: changed \edef to \protected@edef 268
\glsnavhypertarget: added write to aux file 268
\glsnavigation: changed to only use labels for groups that are present . . 270
1.15 (2008-08-15)
\@gls@link: added \glslabel 112
\gls@defglossaryentry: check for \@glo@first in description 86
check for \@glo@text in symbol 86
\gls@hypergrouprerun: new 269
\glsnavhypertarget: added check if rerun required 268
\glssettoctitle: new 34
\printglossary: changed the way the TOC title is set 190
1.16 (2008-08-27)
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 125
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used 128
\@Gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 125
\@Glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 127
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 124
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used 129
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 126
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page 122
\gls@defglossaryentry: Changed def to let 82
1.17 (2008-12-26)
\@@do@esc@wrglossary: new 184
\@do@seeglossary: new 187
\@glo@storeentry: new 88
\@gls@glossary: changed definition to use \index instead of \@index 180
\@glsdefaultplural: new 69
\@glsdefaultsort: new 69
\@glshypernumber: new 215
\@glsnoname: new 68
\@glsnonextpages: new 206
General: added xindy support 29
parent: new 66
see: new 66
\gls@defglossaryentry: added nonumberlist key 82
added parent key 82
added see key 82
Stored main part of entry format when entry is defined 87
\gls@suffixF: new 39
\gls@suffixFF: new 39
\gls@wrglossary: modified to allow for xindy support 181

\glshyperlink: new 157
\glshypernumber: modified to allow material to be attached to location . 215
\glsnavhyperlink: replaced \hyperlink to \@glslink 268
\glsnavhypertarget: replaced \hypertarget to \@glstarget . . . 268
\glssee: new 188
\glsseeformat: new 188
\glsSetSuffixF: new 39
\glsSetSuffixFF: new 39
\ifglsxindy: new 29
\istfilename: added xindy support . . . 38
\newglossarystyle: made \newglossarystyle long 214
\nopostdesc: new 37
nonumberlist: new 66
\printglossary: added check to determine if \printglossary is already defined 190
added print language to aux file 190
order: order package option added 28
\writeist: added xindy support 161
1.18 (2009-01-14)
\@gls@loadlist: new 10
\@gls@loadlong: new 9
\@gls@loadsuper: new 10
\@gls@loadtree: new 10
\gls@defglossaryentry: Changed default value of sort to \@glsdefaultsort 82
moved sort sanitization to \newglossaryentry 86
\glstarget: new 208
\oldacronym: new 217
nolist: new 10
nolong: new 9
sort: moved sanitization to \newglossaryentry 64
nostyles: new 10
nosuper: new 10
notree: new 10
1.19 (2009-03-02)
\glsclearpage: new 43
\glsdisp: new 128
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller 244
\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller 240
\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller 246
\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller 249
2.01 (2009 May 30)
\@gls@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit 113
\forallglossaries: replaced \ifthenelse with \ifx 53
\forglsentries: replaced \ifthenelse with \ifx 53
\glsdefmain: new 16
\glsdescwidth: changed \linewidth to \hsize 276, 298
\glslistdottedwidth: changed \linewidth to \hsize 276
\glspagelistwidth: changed \linewidth to \hsize 276, 298
nomain: added nomain package option . 16
\writeist: removed item_02 - no such makeindex key 165
2.02 (2007-07-13)
\@printglossary: suppressed warning globally rather than locally 193
2.02 (2009-07-13)
\glossarysection: changed \@mkboth to \glossarymark 41
\glsglossarymark: New 41
2.03 (2009-09-23)
\@GLS@: Added check for hyperfirst 125
\@GLSpl: Added check for hyperfirst . . . 128
\@Gls@: Added check for hyperfirst 125
\@Glspl@: Added check for hyperfirst . . 127
\@gls@: Added check for hyperfirst 124
\@gls@@link: new 111
\@gls@link: added \leavevmode 112
Moved entry existence check to avoid duplicate code 112
\@glsdisp: Added check for hyperfirst . 129
\@glspl@: Added check for hyperfirst . . 126
\glsglossarymark: Added check to see if it's already defined 41
hyperfirst: new 27

2.04 (2009-11-10)
- \@GLS@: Changed test to check if glossary type has been identified as a list of acronyms . . . 125
- \@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms . . . 128
- \@Gls@: Changed test to check if glossary type has been identified as a list of acronyms . . . 125
- \@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms . . . 127
- \@glossaryentryfield: new . . . 88
- \@glossarysubentryfield: new . . . 88
- \@gls@: Changed test to check if glossary type has been identified as a list of acronyms . . . 124
- \@glsacronymlists: new . . . 17
- \@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms . . . 129
- \@glspl@: Changed test to check if glossary type has been identified as a list of acronyms . . . 126
- \@newglossaryentryposthook: new . . 87
- \@newglossaryentryprehook: new . . . 87
- acronymlists: new . . . 19
- \DeclareAcronymList: new . . . 18
- \DefineAcronymSynonyms: new . . . 235
- \gls@defglossaryentry: added user1-6 keys . . . 82
- \glsadd: fixed bug that ignored counter 158
- \Glsentryuseri: new . . . 153
- \glsentryuseri: new . . . 153
- \Glsentryuserii: new . . . 154
- \glsentryuserii: new . . . 154
- \Glsentryuseriii: new . . . 154
- \glsentryuseriii: new . . . 154
- \Glsentryuseriv: new . . . 154
- \glsentryuseriv: new . . . 154
- \Glsentryuserv: new . . . 154
- \glsentryuserv: new . . . 154
- \Glsentryuservi: new . . . 155
- \glsentryuservi: new . . . 155
- \ns@newglossary: added check to determine if \gls@⟨*type*⟩@display and \gls@⟨*type*⟩@displayfirst have been defined. . . . 61
- \SetAcronymLists: new . . . 19
- \SetDefaultAcronymDisplayStyle: new . . . 237
- \SetDefaultAcronymStyle: new . . . 238
- \SetDescriptionAcronymDisplayStyle: new . . . 242
- \SetDescriptionDUAAcronymDisplayStyle: new . . . 241
- \SetDescriptionFootnoteAcronymDisplayStyle: new . . . 238
- \SetDUADisplayStyle: new . . . 250
- \SetFootnoteAcronymDisplayStyle: new . . . 245
- \SetSmallAcronymDisplayStyle: new 247

2.05 (2010-02-06)
- \@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco . . . 128
  - Removed spurious brace. Patch provided by Sergiu Dotenco . . . 129
- \writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco . . . 166

2.06 (2010-06-14)
- \altnewglossary: new . . . 62
- \CustomAcronymFields: new . . . 252
- \CustomNewAcronymDef: new . . . 252
- \SetCustomDisplayStyle: new . . . 252
- \SetCustomStyle: new . . . 253

2.07 (2010-07-10)
- General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format) . . . 158

3.0 (2010-07-12)
- \@makeglossary: Added check for savewrites . . . 170
- \gls@wrglossary: modified to take into account savewrites . . . 181

3.0 (2010/03/31)
- \@set@glo@numformat: added 4th argument . . . 114

3.0 (2011-04-02)
- \@@do@esc@wrglossary: added check for hyper location prefix . . . 186
  - modified to use new format . . . 184
- \@@glossarysec: replaced \@ifundefined with \ifcsundef . . . 7
- \@do@seeglossary: Sanitize and escape cross-referencing information . . . 187
- \@gls@counterwithin: new . . . 12

\@gls@ifinlist: new 44
\@gls@link: added
  \@gls@saveentrycounter 113
  added \@gls@setsort 113
\@gls@saveentrycounter: new 113
\@gls@setupsort@def: new 14
\@gls@setupsort@standard: new 13
\@gls@setupsort@use: new 14
\@gls@xdy@locationlist: new 47
\@glslink: replaced \@ifundefined
  with \ifcsundef 122
\@glsnextpages: new 206
\@print@glossary: replaced
  \@ifundefined with \ifcsundef . 193
\@printglossary: added
  \currentglossary 192
  added \glsnextpages 192
  make toctitle default to title 192
\@xdyattributelist: new 44
General: added prefix to hyperlink 216
  etoolbox now loaded 4
  replaced \@ifundefined with
  \ifcsundef 33, 36, 109, 204
\acrfootnote: new 238
\ACRfull: added starred version 220
\Acrfull: added starred version 220
\acrfull: added starred version 219
\ACRfullpl: added starred version . . . 221
\Acrfullpl: added starred version . . . 221
\acrfullpl: added starred version . . . 220
\acrlinkfootnote: new 238
\acrnolinkfootnote: new 238
savewrites: new 30
see: added \@glo@seeautonumberlist 66
seeautonumberlist: new 9
\glossarysection: replaced
  \@ifundefined with \ifcsundef . . 41
\glossarystyle: replaced
  \@ifundefined with \ifcsundef . 213
\gls@codepage: replaced
  \@ifundefined with \ifcsundef . . 29
\gls@defglossaryentry: added
  \@gls@defsort 86
  added short and long keys 83
  replaced \@ifundefined with
  \ifcsundef 83
\gls@doclearpage: replaced
  \@ifundefined with \ifcsundef . . 43
\glsadd: added
  \@gls@saveentrycounter 158
\GlsAddXdyCounters: new 45
\glsentrycounterlabel: new 207
\glsentryitem: new 207
\Glsentrylong: new 155
\glsentrylong: new 155
\Glsentrylongpl: new 155
\glsentrylongpl: new 155
\Glsentryshort: new 155
\glsentryshort: new 155
\Glsentryshortpl: new 155
\glsentryshortpl: new 155
\glsgetgrouptitle: replaced
  \@ifundefined with \ifcsundef . 212
\glsglossarymark: replaced
  \@ifundefined with \ifcsundef . . 41
\glshyperlink: changed default from
  \glsentryname to \glsentrytext 157
\glshypernumber: replaced
  \@ifundefined with \ifcsundef . 215
\glsnumberformat: replaced
  \@ifundefined with \ifcsundef . . 39
\glsrefentry: new 207
\glsresetsubentrycounter: new . . . 206
\glsseeitem: hyperlink uses
  \glsseeitemformat instead of
  \glsentryname 189
\glsseeitemformat: new 189
\glssortnumberfmt: new 13
\glsstepentry: new 207
\glsstepsubentry: new 207
\glssubentrycounterlabel: new . . . 207
\glssubentryitem: new 208
theglossary: replaced \@ifundefined
  with \ifcsundef 208
short: new 68
shortplural: new 68
\ifglossaryexists: replaced
  \@ifundefined with \ifcsundef . . 54
\ifglsentryexists: replaced
  \@ifundefined with \ifcsundef . . 54
\istfile: deprecated 178
glossaryentry: new 11
glossarysubentry: new 12
\newglossaryentry: replaced
  \DeclareRobustCommand with
  \newrobustcmd 71

\newglossarystyle: replaced
\@ifundefined with \ifcsundef . 214
\ns@newglossary: added
\@gls@defsortcount 62
replaced \@ifundefined with
\ifcsundef 61
entrycounter: new 11
\oldacronym: replaced \@ifundefined
with \ifcsundef 217
compatible-2.07: compatible-2.07
option added 30
long: new 68
longplural: new 68
nonumberlist: now boolean 66
sort: new 12
counter: replaced \@ifundefined with
\ifcsundef 65
counterwithin: new 11
\printglossary: replaced
\@ifundefined with \ifcsundef . 190
\SetDescriptionFootnoteAcronymDisplayStyle:
expanded options link options 238
\setentrycounter: added optional
argument 213
\showacronymlists: new 258
\showglocounter: new 255
\showglodesc: new 256
\showglodescplural: new 257
\showglofirst: new 255
\showglofirstpl: new 255
\showgloflag: new 258
\showgloindex: new 258
\showglolevel: new 254
\showgloname: new 256
\showgloparent: new 254
\showgloplural: new 254
\showglosort: new 257
\showglossaries: new 258
\showglossarycounter: new 259
\showglossaryentries: new 259
\showglossaryin: new 259
\showglossaryout: new 259
\showglossarytitle: new 259
\showglosymbol: new 257
\showglosymbolplural: new 257
\showglotext: new 254
\showglotype: new 255
\showglouseri: new 255
\showglouserii: new 255
\showglouseriii: new 256
\showglouseriv: new 256
\showglouserv: new 256
\showglouservi: new 256
subentrycounter: new 12
\writeist: added xindy-only macro
definitions to glossary open tag 163
modified to support new format 161
3.01 (2011-04-12)
\@glswritefiles: added check for
empty glossaries 179
General: made robust 125
\ACRfull: made robust 220
\Acrfull: made robust 219
\acrfull: made robust 219
\acrfullformat: removed
\acronymfont as it should already be
set in the second argument. 219
\ACRfullpl: made robust 221
\Acrfullpl: made robust 221
\acrfullpl: made robust 220
\ACRlong: made robust 146
\Acrlong: made robust 146
\acrlong: made robust 145
\ACRlongpl: made robust 148
\Acrlongpl: made robust 148
\acrlongpl: made robust 147
\ACRshort: made robust 143
\Acrshort: made robust 142
\acrshort: made robust 141
\ACRshortpl: made robust 144
\Acrshortpl: made robust 144
\acrshortpl: made robust 143
\Gls: made robust 124
\glsadd: made robust 158
\glsaddall: made robust 158
\GLSdesc: made robust 134
\Glsdesc: made robust 134
\glsdesc: made robust 133
\GLSdescplural: made robust 135
\Glsdescplural: made robust 134
\glsdescplural: made robust 134
\glsfirst: made robust 130
\GLSfirstplural: made robust 132
\Glsfirstplural: made robust 132
\glsfirstplural: made robust 132
\glslink: made robust 111
\GLSname: made robust 133
\Glsname: made robust 133

\glsname: made robust 133
\GLSpl: made robust 127
\Glspl: made robust 127
\glspl: made robust 126
\GLSplural: made robust 131
\GLSsymbol: made robust 136
\Glssymbol: made robust 135
\glssymbol: made robust 135
\GLSsymbolplural: made robust 136
\Glssymbolplural: made robust 136
\glssymbolplural: made robust 136
\Glstext: made robust 130
\glstext: made robust 129
\GLSuseri: made robust 137
\Glsuseri: made robust 137
\glsuseri: made robust 137
\GLSuserii: made robust 138
\Glsuserii: made robust 138
\glsuserii: made robust 137
\GLSuseriii: made robust 139
\Glsuseriii: made robust 139
\glsuseriii: made robust 138
\GLSuseriv: made robust 140
\Glsuseriv: made robust 139
\glsuseriv: made robust 139
\GLSuserv: made robust 140
\Glsuserv: made robust 140
\glsuserv: made robust 140
\GLSuservi: made robust 141
\Glsuservi: made robust 141
\glsuservi: made robust 141

3.02 (2012-05-19)
\glsnumlistlastsep: new 157
\glsnumlistsep: new 157

3.02 (2012-05-21)
\@@do@@wrglossary: changed \@glslocref to \theglsentrycounter 186
\@do@wrglossary: changed \@do@wr@glossary to test for indexonlyfirst option; put old \@do@wr@glossary code into \@@do@wrglossary 181
\@gls@missingnumberlist: new 69
\@glswritefiles: added check for existence of token in case \makeglossaries has been omitted 179
\@printglossary: add a way to fetch current entry label 192
savenumberlist: new 9
ucmark: new 11
\gls@defglossaryentry: added numberlist element 86
\gls@save@numberlist: new 189
\gls@wrglossary: added check for glossary file defined 181
\glsdisplaynumberlist: new 156
\glsentrycounter: set default value . . 113
\Glsentryfull: fixed bug (replaced \glsentryshortpl with \glsentryshort) 156
\glsentryfullpl: fixed bug (replaced \glsentryshort with \glsentryshortpl) 156
\glsentrynumberlist: new 156
\glsmoveentry: new 88
\glsresetsubentrycounter: new . . . 206
\ifglshaschildren: new 56
\ifglshasparent: new 56
\makeglossaries: added list parser . . 173
indexonlyfirst: new 27
\renewglossarystyle: new 214
\showglossaryentries: fixed misspelt command 259
\SmallNewAcronymDef: fixed broken short and long plural 248

3.03 (2012/09/21)
\@gls@sanitizesort: new 22
\@gls@setupsort@standard: used \@gls@sanitizesort 13
\@printglossary: allow title to override default toctitle 191
General: allow title to set toctitle 204
\glsinlinedescformat: new 272
\glsinlineemptydescformat: new . . 272
\glsinlinenameformat: new 272
\glsinlinepostchild: new 272
\glsinlinesubdescformat: new 272
\glsinlinesubnameformat: new 272
\glspostinline: replaced "." with \glspostdescription 272
list: added check for glsnogroupskip . 274
altlongragged4col: added check for glsnogroupskip 291
altsuperragged4col: added check for glsnogroupskip 310

alttree: added check for glsnogroupskip 319
index: added check for glsnogroupskip 313
nogroupskip: new 11
long: added check for glsnogroupskip . 277
long3col: added check for glsnogroupskip 279
long4col: added check for glsnogroupskip 280
longragged: added check for glsnogroupskip 288
longragged3col: added check for glsnogroupskip 289
nopostdot: new 11
tree: added check for glsnogroupskip . 314
treenoname: added check for glsnogroupskip 316
super: added check for glsnogroupskip 299
super3col: added check for glsnogroupskip 301
super4col: added check for glsnogroupskip 303
superragged: added check for glsnogroupskip 306
superragged3col: added check for glsnogroupskip 308

3.04 (2012-11-11)
altlist: replaced \newline with paragraph break 274

3.04 (2012-11-18)
\@@do@@wrglossary: changed \theglsentrycounter back to \@glslocref 186
\@@do@esc@wrglossary: modified to compensate for possible incorrect page number 185
\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage 115
\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here. . . 193
General: Added check for doc package . . . 4
added datatool-base as a required package 4
added local key 109
\gls@Alphpage: new 182
\gls@alphpage: new 182
\gls@disablepagerefexpansion: new 181
\gls@numberpage: new 182
\gls@protected@pagefmts: new 181
\gls@romanpage: new 182
\glsdefmain: added check for doc package 16
\glsorg@endtheglossary: new 5
\glsorg@theglossary: new 5
\PrintChanges: new 5

3.05 (2013-04-21)
\@@do@esc@wrglossary: add Roman case. Fixed bugs in the else statements 185
\@gls@link: added check for “nohypertypes” 112
mcolalttree: replaced ‘2’ with \glsmcols 297
mcolindex: replaced ‘2’ with \glsmcols 293
mcolindexspannav: replaced ‘2’ with \glsmcols 294
mcoltree: replaced ‘2’ with \glsmcols 294
mcoltreenoname: replaced ‘2’ with \glsmcols 296
mcoltreespannav: replaced ‘2’ with \glsmcols 295
\gls@protected@pagefmts: added Roman to list 181
\gls@Romanpage: new 182
\glsgetgrouplabel: fixed bug (typo in \equal) 213
\nopostdesc: made robust 37

3.05 (2013/04/21)
\@gls@nohyperlist: new 19
\GlsDeclareNoHyperList: new 19
nohypertypes: new 19

3.06 (2013/06/17)
\@xdy@main@language: Changed back to using \languagename 29
\findrootlanguage: Obsoleted 51

3.07 (2013-07-05)
\@gls@link: fixed bug that failed to find entry in list 112
\glossarypreamble: modified to work with \setglossarypreamble 40
\gls@doclearpage: added check for openright 43
\glspostdescription: Added spacefactor code 10

\GlsSetXdyCodePage: Added check for fontspec 52
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty 242
\setglossarypreamble: new 40

3.08a (2013-08-30)
list: updated list style to use \glossentry and \subglossentry 273
listdotted: updated listdotted style to use \glossentry and \subglossentry 275
altlist: updated altlist style to use \glossentry and \subglossentry 274
inline: updated inline style to use \glossentry and \subglossentry 271

3.08a (2013-09-28)
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort 89
updated for \glossentry 89
\@glossaryentryfield: switched to \glossentry 88
\@glossarysubentryfield: switched to \subglossentry 88
General: added nogroupskip key to \printglossary 204
removed definition of \@glossaryentryfield 361
removed definition of \@glossarysubentryfield 361
\compatibleglossentry: new 208
\compatiblesubglossentry: new . . . 210
\glossaryentryfield: deprecated . . . 210
\Glossentrydesc: new 209
\glossentrydesc: new 209
\Glossentryname: new 209
\glossentryname: new 209
\Glossentrysymbol: new 210
\glossentrysymbol: new 209
\gls@assign@desc@field: new 21
\gls@assign@descplural@field: new 21
\gls@assign@field: new 71
\gls@ifnotmeasuring: new 90
\glsaddallunused: new 159
\glsexpandfields: new 71
\glsnoexpandfields: new 71
\glssee: made robust 188
\glsseeformat: made robust 188
\glsseeitem: made robust 189
\glsseelist: made robust 188
\ifglsdescsuppressed: new 57
\ifglshasdesc: new 57
\ifglshassymbol: new 57
altlongragged4col: updated to use \glossentry and \subglossentry 291
alttree: updated to use \glossentry and \subglossentry 318
index: added paragraph break at end of environment 312
updated to use \glossentry and \subglossentry 312
long: updated to use \glossentry and \subglossentry 277
longragged: updated to use \glossentry and \subglossentry 288
longragged3col: updated to use \glossentry and \subglossentry 289
tree: updated to use \glossentry and \subglossentry 314
\setglossarystyle: new 213
\setglossentrycompatibility: new 210
superragged: updated to use \glossentry and \subglossentry 306

3.09a (2013-10-09)
\@gls@assign@symbolplural@field: new 22
\@gls@default@value: new 65
\Glsentrydesc: made robust 151
\Glsentrydescplural: made robust . . 151
\Glsentryfirst: made robust 152
\Glsentryfirstplural: made robust . 153
\Glsentryfull: made robust 156
\Glsentryfullpl: made robust 156
\Glsentrylong: made robust 155
\Glsentrylongpl: made robust 155
\Glsentryname: made robust 150
\Glsentryplural: made robust 152
\Glsentryshort: made robust 155
\Glsentryshortpl: made robust 155
\Glsentrysymbol: made robust 152
\Glsentrysymbolplural: made robust 152
\Glsentrytext: made robust 151
\Glsentryuseri: made robust 153
\Glsentryuserii: made robust 154
\Glsentryuseriii: made robust 154
\Glsentryuseriv: made robust 154
\Glsentryuserv: made robust 154
\Glsentryuservi: made robust 155

\glstextup: new 218
\ifglshassymbol: changed test to check for \@gls@default@symbol 57

3.10a (2013-09-28)
\gls@assign@type@field: new 21

3.10a (2013-10-13)
\@gls@keymap: new 73
\@gls@provide@newglossary: new . . . 60
\@gls@writedef: new 73
\@glsdefaultplural: Obsolete 69
\@glsnodesc: new 69
\@print@glossary: Added providecommand code to aux file 193, 194
\gls@defglossaryentry: Changed to using \@gls@default@value 82
new 81
\glswritedefhook: new 81
\makeglossaries: Added providecommand code to aux file . . 172
\new@glossaryentry: new 72
\ns@newglossary: added \@gls@provide@newglossary 61

3.11a (2013-10-15)
\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext 361
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext 359
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext 360
\@Acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext 359
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert 125
change to using \glsentryfmt style commands 125
removed \MakeUppercase (now moved to \glsentryfmt) 125
\@GLSpl: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert 128
change to using \glsentryfmt style commands 128
removed \MakeUppercase as now dealt with in \glsentryfmt 128
\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert 125
change to using \glsentryfmt style commands 125
removed \makefirstuc (now dealt with in \glsentryfmt) 125
\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert 127
change to using \glsentryfmt style commands 127
removed \makefirstuc (now dealt with in \glsentryfmt) 127
\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext 360
\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext 359
\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert 124
change to using \glsentryfmt style commands 124
\@gls@noexpand@fields: Fixed bug expand replaced with noexpand 70
\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert 128
change to using \glsentryfmt style commands 129
\@glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert 126
change to using \glsentryfmt style commands 126
General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext 142–148
changed to just use \Glsentrydescplural 135
changed to just use \glsentrydescplural 134, 135
changed to just use \Glsentrydesc . 134
changed to just use \glsentrydesc . 134

changed to just use \Glsentryfirstplural 132
changed to just use \glsentryfirstplural 132
changed to just use \Glsentryfirst 131
changed to just use \glsentryfirst 130, 131
changed to just use \Glsentryname . 133
changed to just use \glsentryname . 133
changed to just use \Glsentryplural 131
changed to just use \glsentryplural 131, 132
changed to just use \Glsentrysymbolplural 136
changed to just use \glsentrysymbolplural 136, 137
changed to just use \Glsentrysymbol 135
changed to just use \glsentrysymbol 135, 136
Changed to just use \Glsentrytext . 130
changed to just use \glsentrytext . 129
changed to just use \Glsentryuseriii 139
changed to just use \glsentryuseriii 138, 139
changed to just use \Glsentryuserii 138
changed to just use \glsentryuserii 138
changed to just use \Glsentryuseriv 139
changed to just use \glsentryuseriv 139, 140
changed to just use \Glsentryuseri 137
changed to just use \glsentryuseri 137
changed to just use \Glsentryuservi 141
changed to just use \glsentryuservi 141
changed to just use \Glsentryuserv 140
changed to just use \glsentryuserv 140, 141
Now requires textcase 4
acronymlists: replaced \@addtoacronymlists with \DeclareAcronymList 19
\defglsdisplay: obsoleted 108
\defglsdisplayfirst: obsoleted 108
\defglsentryfmt: new 60
\forglsentries: replaced \ifx with \ifdefempty 53
\gls@assign@desc: new 81
\gls@defglossaryentry: Fixed default counter if none supplied 85
\gls@doentryfmt: new 60
\glsdisplay: obsoleted 107
\glsdisplayfirst: obsoleted 107
\glsgenentryfmt: new 102
\glsgetgrouptitle: Added check in case non-Latin alphabet in use 212
\glsglossarymark: replaced \MakeUppercase with \mfirstucMakeUppercase 41
\glsnavigation: switched to using \@gls@getgrouptitle 270
\ifglshasdesc: replaced \ifdefempty with \ifcsempty 57
\ifglshaslong: new 57
\ifglshasshort: new 57
\ifglshassymbol: replaced \ifdefempty with \ifcsempty 57
\ifglsused: replaced \ifthenelse with \ifbool 54
\longnewglossaryentry: new 81
\ns@newglossary: replaced \glsdisplay and \glsdisplayfirst with \glsentryfmt 61
compatible-3.07: cnew 30
\SetCustomDisplayStyle: updated to use \defglsentryfmt 252
\SetDefaultAcronymDisplayStyle: changed to use \defglsentryfmt . 237
\SetDescriptionAcronymDisplayStyle: updated to use \defglsentryfmt . 242
\SetDescriptionDUAAcronymDisplayStyle: updated to use \defglsentryfmt . 241
\SetDescriptionFootnoteAcronymDisplayStyle: updated to use \defglsentryfmt . 238
\SetDUADisplayStyle: updated to use \defglsentryfmt 250
\SetFootnoteAcronymDisplayStyle: updated to use \defglsentryfmt . 245
\SetSmallAcronymDisplayStyle: updated to use \defglsentryfmt . 247
\setupglossaries: new 32
\showglolong: new 257
\showgloshort: new 257
numbers: new 31
symbols: new 31
3.12a (2013-10-16)
\gls@defglossaryentry: added \glslabel 82

\glsaddkey: new 75
3.13a (2013-11-05)
\@gls@assign@symbol@field: changed to use \glssetnoexpandfield 22
\@gls@assign@symbolplural@field: changed to use \glssetnoexpandfield 22
\@gls@link: removed \relax 113
\@gls@notranslatorhook: new 25
\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort 13
ucmark: added check for memoir 11
see: added \gls@checkseeallowed . . . 66
\glossarysection: changed \glossarymark to \glsglossarymark 41
\glossarystyle: fixed bug caused by using \ifdef instead of \ifcsdef . 214
\gls@assign@desc@field: changed to use \glssetnoexpandfield 21
\gls@assign@descplural@field: changed to use \glssetnoexpandfield 21
\gls@assign@name@field: changed to use \glssetnoexpandfield 21
\gls@assign@type@field: changed to use \glssetexpandfield 21
\gls@checkseeallowed: new 66
\glsaddallunused: set default to \@glo@types 159
\Glsentryfull: changed to use \acrfullformat 156
\glsentryfull: changed to use \acrfullformat 156
\Glsentryfullpl: changed to use \acrfullformat 156
\glsentryfullpl: changed to use \acrfullformat 156
\glsglossarymark: renamed \glossarymark to \glsglossarymark to avoid conflict with memoir 41
\glsprestandardsort: new 13
\glssetexpandfield: new 21
\glssetnoexpandfield: new 21
altsuper4colheader: switched to \tabularnewline 304
altsuper4colheaderborder: switched to \tabularnewline 305
long: switched to \tabularnewline . . 277
long3col: switched to \tabularnewline 278
long3colheader: switched to \tabularnewline 279
long3colheaderborder: switched to \tabularnewline 279
long4col: switched to \tabularnewline 280
long4colheader: switched to \tabularnewline 280
longheader: switched to \tabularnewline 278
longheaderborder: switched to \tabularnewline 278
\SetFootnoteAcronymDisplayStyle: fixed missing argument bug 245
super: switched to \tabularnewline . 299
super3col: switched to \tabularnewline 301
super3colheader: switched to \tabularnewline 301
super4col: switched to \tabularnewline 302
super4colheader: switched to \tabularnewline 303
super4colheaderborder: switched to \tabularnewline 303
superheader: switched to \tabularnewline 300
superheaderborder: switched to \tabularnewline 300
3.14a (2013-11-12)
\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles 179
General: new 261
acronyms: new 17
\gls@defglossaryentry: added check for existence of default glossary 83
set the default for firstplural to be the value of plural 85
xindygloss: new 29
\longprovideglossaryentry: new . . . 81

compatible-2.07: added check for 2.07 before setting 3.07 compatibility 30
notranslate: new 26
\provideglossaryentry: new 72

4.0 (2013-11-14)
\gls@defglossaryentry: added check for first key 85
super: fixed typo in \subglossentry (\glossentrydesc) 299

4.01 (2013-11-16)
General: fixed non-value options so that they can be passed to document class . 8
\CustomAcronymFields: inserted missing comma 252

4.02 (2013-12-05)
\@acrfull: now using \acrfullfmt .. 219
\@gls@indexdef: new 31
\@gls@numbersdef: new 31
\@gls@symbolsdef: new 31
General: Removed \acronymfont . 145–149
\ACRfullfmt: new 220
\Acrfullfmt: new 220
\acrfullfmt: new 219
\ACRfullplfmt: new 221
\Acrfullplfmt: new 221
\acrfullplfmt: new 221
\acronymentry: new 223
sanitize: fixed bug that caused an error here 25
sc-short-long: new 227
sc-short-long-desc: new 229
\Genacrfullformat: new 107
\genacrfullformat: new 107
\GenericAcronymFields: new 223
\Genplacrfullformat: new 107
\genplacrfullformat: new 107
\Glsentryfull: bug fix: added missing \acronymfont 156
\glsentryfull: bug fix: added missing \acronymfont 156
\Glsentryfullpl: bug fix: added missing \acronymfont 156
\glsentryfullpl: bug fix: added missing \acronymfont 156
\glsgenacfmt: new 105
\GlsUseAcrEntryDispStyle: new ... 225
\GlsUseAcrStyleDefs: new 225
short-long: new 226
short-long-desc: new 229
xindynoglsnumbers: new 29
sm-short-long: new 228
sm-short-long-desc: new 230
index: new 31
\newacronymstyle: new 224
long-sc-short: new 227
long-sc-short-desc: new 228
long-short: new 225
long-short-desc: new 228
long-sm-short: new 227
long-sm-short-desc: new 229
long-sp-short-desc: new 228
footnote: new 232
footnote-desc: new 234
footnote-sc: new 233
footnote-sc-desc: new 234
footnote-sm: new 234
footnote-sm-desc: new 235
\setacronymstyle: new 224
\SetDescriptionAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material 242
\SetDescriptionFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material 238
\SetFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material 245
\SetGenericNewAcronym: new 222
\SetSmallAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material 247
dua: new 230
dua-desc: new 232
numberedsection: added nameref option 8

4.02 (2013-13-05)
\makeglossaries: made preamble only 174

4.03 (2014-01-17)
General: changed default to \@empty instead of \relax 30

4.03 (2014-01-20)
\@@do@esc@wrglossary: added \glsdetoklabel 186

\@@do@noesc@wrglossary: added
\glsdetoklabel 183
\@ACRlong: removed \glslabel
(defined in \@gls@link) 361
\@ACRshort: removed \glslabel
(defined in \@gls@link) 359
\@Acrlong: removed \glslabel
(defined in \@gls@link) 360
\@Acrshort: removed \glslabel
(defined in \@gls@link) 359
\@GLS@: removed \glslabel (defined in
\@gls@link) 125
\@GLSpl: removed \glslabel (defined
in \@gls@link) 128
\@Gls@: removed \glslabel (defined in
\@gls@link) 125
\@Gls@entry@field: new 149
\@Glspl@: removed \glslabel (defined
in \@gls@link) 127
\@acrlong: removed \glslabel
(defined in \@gls@link) 360
\@acrshort: removed \glslabel
(defined in \@gls@link) 359
\@gls@: removed \glslabel (defined in
\@gls@link) 124
\@gls@access@display: new 347
\@gls@entry@field: new 149
\@gls@fetchfield: new 74
\@gls@field@link: new 129
\@gls@link: added \glsdetoklabel . 112
moved \@gls@link@opts and
\@gls@link@label to \@gls@link 112
\@gls@writedef: added
\glsdetoklabel 73
\@glsdisp: removed \glslabel
(defined in \@gls@link) 128
\@glspl@: removed \glslabel (defined
in \@gls@link) 126
\@printglossary: added
\glsdetoklabel 192
General: removed \glslabel (defined in
\@gls@link) 142
sc-short-long-desc: redefined to use
accessibility information 365
\compatibleglossentry: added
\glsdetoklabel 341
\compatiblesubglossentry: added
\glsdetoklabel 342
\Genacrfullformat: redefined to use
accessibility information 358
\genacrfullformat: redefined to use
accessibility information 358
\Genplacrfullformat: redefined to use
accessibility information 358
\genplacrfullformat: redefined to use
accessibility information 358
\glossentryname: added
\glsdetoklabel 209
\gls@defglossaryentry: added
\glsdetoklabel 81
replaced #1 with \@glo@label 83
replaced \ifthenelse with
\ifdefequal 84
\glsadd: added \glsdetoklabel 158
\glsaddkey: switched to using
\@gls@field@link 76
\glsdetoklabel: new 54
\glsdisplaynumberlist: added
\glsdetoklabel 157
\glsdoifexistsorwarn: new 55
\glsentryaccess: switched to using
\@gls@entry@field 345
\glsentrydescaccess: switched to
using \@gls@entry@field 346
\glsentrydescpluralaccess: switched
to using \@gls@entry@field 346
\glsentryfirstaccess: switched to
using \@gls@entry@field 346
\glsentryfirstplural: added
\glsdetoklabel 153
\glsentrylongaccess: switched to
using \@gls@entry@field 347
\glsentrylongpluralaccess: switched
to using \@gls@entry@field 347
\glsentrypluralaccess: switched to
using \@gls@entry@field 346
\glsentryshortaccess: switched to
using \@gls@entry@field 346
\glsentryshortpluralaccess:
switched to using
\@gls@entry@field 346
\glsentrysymbolaccess: switched to
using \@gls@entry@field 346
\glsentrysymbolpluralaccess:
switched to using
\@gls@entry@field 346

\glsentrytextaccess: switched to using \@gls@entry@field 346
\glsgenacfmt: redefined to use accessibility information 356
\glsgenentryfmt: redefined to use accessibility information 353
\glshyperlink: added \glsdetoklabel 157
\glslocalreset: added \glsdetoklabel 90
\glslocalunset: added \glsdetoklabel 91
\glsmoveentry: added \glsdetoklabel 88
replaced \ifthenelse with \ifdefequal 88
\glsrefentry: added \glsdetoklabel 207
\glsreset: added \glsdetoklabel . . . 90
\glsseelist: added \expandafter commands 189
\glsstepentry: added \glsdetoklabel 207
\glsstepsubentry: added \glsdetoklabel 207
\glsunset: added \glsdetoklabel . . . 91
short-long: commented spurious EOL 227
redefined to use accessibility information 363
short-long-desc: redefined to use accessibility information 365
\ifglsdescsuppressed: added \glsdetoklabel 57
fixed typo 57
\ifglsentryexists: added \glsdetoklabel 54
\ifglshaschildren: added \glsdetoklabel 56
\ifglshasdesc: added \glsdetoklabel 57
\ifglshasfield: new 58
\ifglshaslong: added \glsdetoklabel 57
\ifglshasparent: added \glsdetoklabel 56
\ifglshasshort: added \glsdetoklabel 57
\ifglshassymbol: added \glsdetoklabel 57
replaced \ifcsempty with \ifdefempty and replaced \ifx with \ifdefequal 57
\ifglsused: added \glsdetoklabel . . 54
sm-short-long-desc: redefined to use accessibility information 365
long-sc-short-desc: redefined to use accessibility information 364
long-short: redefined to use accessibility information 362
long-short-desc: redefined to use accessibility information 364
long-sm-short-desc: redefined to use accessibility information 364
footnote: redefined to use accessibility information 368
footnote-desc: redefined to use accessibility information 370
footnote-sc: redefined to use accessibility information 370
footnote-sc-desc: redefined to use accessibility information 371
footnote-sm: redefined to use accessibility information 370
footnote-sm-desc: redefined to use accessibility information 371
\renewacronymstyle: new 224
\showglocounter: added \glsdetoklabel 255
\showglodesc: added \glsdetoklabel 256
\showglodescaccess: added \glsdetoklabel 377
\showglodescplural: added \glsdetoklabel 257
\showglodescpluralaccess: added \glsdetoklabel 377
\showglofirst: added \glsdetoklabel 255
\showglofirstaccess: added \glsdetoklabel 377
\showglofirstpl: added \glsdetoklabel 255
\showglofirstpluralaccess: added \glsdetoklabel 377
\showgloflag: added \glsdetoklabel 258
\showgloindex: added \glsdetoklabel 258
\showglolevel: added \glsdetoklabel 254

\showglolong: added \glsdetoklabel 257
\showglolongaccess: added \glsdetoklabel 378
\showglolongpluralaccess: added \glsdetoklabel 378
\showgloname: added \glsdetoklabel 256
\showglonameaccess: added \glsdetoklabel 377
\showgloparent: added \glsdetoklabel 254
\showgloplural: added \glsdetoklabel 254
\showglopluralaccess: added \glsdetoklabel 377
\showgloshort: added \glsdetoklabel 257
\showgloshortaccess: added \glsdetoklabel 378
\showgloshortpluralaccess: added \glsdetoklabel 378
\showglosort: added \glsdetoklabel 257
\showglosymbol: added \glsdetoklabel 257
\showglosymbolaccess: added \glsdetoklabel 377
\showglosymbolplural: added \glsdetoklabel 257
\showglosymbolpluralaccess: added \glsdetoklabel 377
\showglotext: added \glsdetoklabel 254
\showglotextaccess: added \glsdetoklabel 377
\showglotype: added \glsdetoklabel 255
\showglouseri: added \glsdetoklabel 255
\showglouserii: added \glsdetoklabel 255
\showglouseriii: added \glsdetoklabel 256
\showglouseriv: added \glsdetoklabel 256
\showglouserv: added \glsdetoklabel 256
\showglouservi: added \glsdetoklabel 256
dua: fixed bug in \acrfullfmt 231
  fixed bug in \Acrfullplfmt 231
  fixed bug in \acrfullplfmt 231
  redefined to use accessibility information 366
dua-desc: commented spurious EOL . . 232
  redefined to use accessibility information 368

4.04 (2014-03-04)
\@gls@getcounterprefix: added warning if no prefix can be formed . 187

4.04 (2014-03-06)
\@@gls@noidx@nosanitizesort: new . 22
\@@gls@noidx@sanitizesort: new . . . 22
\@@gls@nosanitizesort: new 22
\@@gls@sanitizesort: new 22
\@glo@addchildren: new 195
\@glo@do@sortentries: new 195
\@glo@grabfirst: new 200
\@glo@sortedinsert: new 196
\@glo@sortentries: new 194
\@glo@sorthandler@case: new 197
\@glo@sorthandler@letter: new . . . 196
\@glo@sorthandler@nocase: new . . . 197
\@glo@sorthandler@word: new 196
\@glo@sortmacro@case: new 198
\@glo@sortmacro@def: new 198
\@glo@sortmacro@def@do: new 199
\@glo@sortmacro@letter: new 197
\@glo@sortmacro@nocase: new 198
\@glo@sortmacro@standard: new . . . 198
\@glo@sortmacro@use: new 199
\@glo@sortmacro@word: new 197
\@gls@noidx@do: new 201
\@gls@noidx@getgrouptitle: new . . 212
\@gls@noref@warn: new 178
\@gls@reference: new 203
\@gls@warnonglossdefined: new 20
\@gls@warnontheglossdefined: new . 20
\@no@makeglossaries: new 178
\@print@glossary: new 193
\@print@noidx@glossary: new 199
\@printgloss@setsort: new 191
\@printglossary: new 191
General: added sort key to printgloss group 205
\compatibleglossentry: changed \newcommand to \def as is may or may not be defined 341
\compatiblesubglossentry: changed \newcommand to \def as is may or may not be defined 342

\defglsdisplayfirst: fixed unwanted space . . . 108
\glo@grabfirst: new . . . 200
\gls@defglossaryentry: replaced \ifx with \ifdefvoid . . . 87
\glsnoidxdisplayloc: new . . . 203
\glsnoidxdisplayloclisthandler: new . . . 202
\glsnoidxloclist: new . . . 202
\glsnoidxloclisthandler: new . . . 202
\glsnoidxstripaccents: new . . . 23
alttree: moved hangindent and parindent assignments outside level test . . . 318
\makeglossaries: Moved definition of \glswrite to \makeglossaries . . 172
\makenoidxglossaries: new . . . 174
\printglossary: changed to use new \@printglossary . . . 190
\printnoidxglossaries: new . . . 191
\printnoidxglossary: new . . . 190
\showgloloclist: new . . . 258
\warn@noprintglossary: Activate warning in \makeglossaries . . . 190
\writeist: checked for definition of \glswrite . . . 161, 165

4.06 (2014-03-12)
\@GLS@: added \glsifhyper . . . 125
\@GLSpl: added \glsifhyper . . . 128
\@Gls@: added \glsifhyper . . . 125
\@Glspl@: added \glsifhyper . . . 127
\@gls@: added \glsifhyper . . . 124
\@gls@numbersdef: added hook to set toc title . . . 31
\@gls@symbolsdef: added hook to set toc title . . . 31
\@glsdisp: added \glsifhyper . . . 129
\@glspl@: added \glsifhyper . . . 126
General: added \glsifhyper . . . 142–149
acronym: added hook to set toc title . . . 17
acronyms: added hook to set toc title . . . 17
\glsdefmain: added hook to set toc title 16

4.07 (2014-04-04)
\@glossarysection: added optional argument when using unstarred version . . . 42
\@gls@noidx@do: added \global in case it's used in a tabular-like style . . . 201
\Acrfullplfmt: fixed no case change bug . . . 221
\glsletentryfield: new . . . 149

4.08 (2014-07-30)
\@ACRlong: added \do@gls@link@checkfirsthyper 360
\@ACRshort: added \do@gls@link@checkfirsthyper 359
\@Acrlong: added \do@gls@link@checkfirsthyper 360
\@Acrshort: added \do@gls@link@checkfirsthyper 359
\@GLS@: moved \glsifhyper . . . 125
moved check for first use to \@gls@link . . . 125
\@GLSpl: moved \glsifhyper . . . 128
moved check for first use to \@gls@link . . . 128
\@Gls@: moved \glsifhyper . . . 125
moved check for first use to \@gls@link . . . 125
\@Glspl@: moved \glsifhyper . . . 127
moved check for first use to \@gls@link . . . 127
\@acrlong: added \do@gls@link@checkfirsthyper 360
\@acrshort: added \do@gls@link@checkfirsthyper 358
\@closegls: new . . . 171
\@gls@: moved \glsifhyper . . . 124
moved check for first use to \@gls@link . . . 124
\@gls@automake: new . . . 171
\@gls@doautomake: new . . . 30
\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper 129
\@gls@forbidtexext: new . . . 60
\@gls@hyp@opt: new . . . 110
\@gls@link: removed redundancy . . . 112
renamed \gls@type to \glstype . . . 112
\@gls@link@checkfirsthyper: new . 111
\@glsdisp: moved \glsifhyper . . . 129
moved check for first use to \@gls@link . . . 129
\@glspl@: moved \glsifhyper . . . 126
moved check for first use to \@gls@link . . . 126
\@ignored@glossaries: new . . . 63

General: added entrycounter option to printgloss family 205
added nopostdot option to printgloss family 205
added subentrycounter option to printgloss family 205
explicitly initialise hyper key 109
moved \glsifhyper 142–149
removed \@sACRlongpl 148
removed \@sAcrlongpl 148
removed \@sacrlongpl 147
removed \@sACRlong 146
removed \@sAcrlong 146
removed \@sacrlong 145
removed \@sACRshortpl 144
removed \@sAcrshortpl 144
removed \@sacrshortpl 143
removed \@sACRshort 143
removed \@sAcrshort 142
removed \@sacrshort 142
removed \@sgls@link 111
removed \@sGLSdescplural 135
removed \@sGlsdescplural 135
removed \@sglsdescplural 134
removed \@sGLSdesc 134
removed \@sGlsdesc 134
removed \@sglsdesc 133
removed \@sglsdisp 128
removed \@sGLSfirstplural 132
removed \@sGlsfirstplural 132
removed \@sglsfirstplural 132
removed \@sGLSfirst 131
removed \@sGlsfirst 130
removed \@sglsfirst 130
removed \@sGLSname 133
removed \@sGlsname 133
removed \@sglsname 133
removed \@sGLSplural 132
removed \@sGlsplural 131
removed \@sglsplural 131
removed \@sGLSpl 127
removed \@sGlspl 127
removed \@sglspl 126
removed \@sGLSsymbolplural 136
removed \@sGlssymbolplural 136
removed \@sglssymbolplural 136
removed \@sGLSsymbol 136
removed \@sGlssymbol 135
removed \@sglssymbol 135
removed \@sGLStext 130
removed \@sGlstext 130
removed \@sglstext 129
removed \@sGLSuseriii 139
removed \@sGlsuseriii 139
removed \@sglsuseriii 138
removed \@sGLSuserii 138
removed \@sGlsuserii 138
removed \@sglsuserii 138
removed \@sGLSuseriv 140
removed \@sGlsuseriv 139
removed \@sglsuseriv 139
removed \@sGLSuseri 137
removed \@sGlsuseri 137
removed \@sglsuseri 137
removed \@sGLSuservi 141
removed \@sGlsuservi 141
removed \@sglsuservi 141
removed \@sGLSuserv 140
removed \@sGlsuserv 140
removed \@sglsuserv 140
removed \@sGLS 125
removed \@sGls 124
removed \@sgls 123
removed \@thirdofthree (defined in kernel) 123
removed sPGLS 266
removed sPgls 264
removed spgls 263
removed sPGLSpl 266
removed sPglspl 265
removed spglspl 264
\ACRfull: removed \s@ACRfull 220
switched to using \@gls@hyp@opt . . 220
\Acrfull: removed \@sAcrfull 220
switched to using \@gls@hyp@opt . . 219
\acrfull: removed \@sacrfull 219
switched to using \@gls@hyp@opt . . 219
\ACRfullpl: removed \s@ACRfullpl . 221
switched to using \@gls@hyp@opt . . 221
\Acrfullpl: removed \s@Acrfullpl . 221
switched to using \@gls@hyp@opt . . 221
\acrfullpl: removed \s@acrfullpl . 220
switched to using \@gls@hyp@opt . . 220
\ACRlong: switched to using \@gls@hyp@opt 146
\Acrlong: switched to using \@gls@hyp@opt 146

\acrlong: switched to using \@gls@hyp@opt 145
\ACRlongpl: switched to using \@gls@hyp@opt 148
\Acrlongpl: switched to using \@gls@hyp@opt 148
\acrlongpl: switched to using \@gls@hyp@opt 147
\ACRshort: switched to using \@gls@hyp@opt 143
\Acrshort: switched to using \@gls@hyp@opt 142
\acrshort: switched to using \@gls@hyp@opt 141
\ACRshortpl: switched to using \@gls@hyp@opt 144
\Acrshortpl: switched to using \@gls@hyp@opt 144
\acrshortpl: switched to using \@gls@hyp@opt 143
\forallacronyms: new 53
\GLS: switched to using \@gls@hyp@opt 125
\Gls: switched to using \@gls@hyp@opt 124
\gls: switched to using \@gls@hyp@opt 123
\gls@defglossaryentry: added check for ignored glossary 83
\gls@istfilebase: new 38
\glsaddkey: removed \@sGLS@user@⟨*key*⟩ 77
  removed \@sGls@user@⟨*key*⟩ 77
  removed \@sgls@user@⟨*key*⟩ 76
  switched to using \@gls@hyp@opt 76, 77
\GLSdesc: switched to using \@gls@hyp@opt 134
\Glsdesc: switched to using \@gls@hyp@opt 134
\glsdesc: switched to using \@gls@hyp@opt 133
\GLSdescplural: switched to using \@gls@hyp@opt 135
\Glsdescplural: switched to using \@gls@hyp@opt 134
\glsdescplural: switched to using \@gls@hyp@opt 134
\glsdisablehyper: added \KV@glslink@hyperfalse to definition 123
\glsdisp: switched to using \@gls@hyp@opt 128
\glsdohyperlink: new 122
\glsdohypertarget: new 122
\glsenablehyper: added \KV@glslink@hypertrue to definition 123
\GLSfirst: switched to using \@gls@hyp@opt 131
\Glsfirst: switched to using \@gls@hyp@opt 130
\glsfirst: switched to using \@gls@hyp@opt 130
\GLSfirstplural: switched to using \@gls@hyp@opt 132
\Glsfirstplural: switched to using \@gls@hyp@opt 132
\glsfirstplural: switched to using \@gls@hyp@opt 132
\glsifhyper: deprecated 110
\glslink: switched to using \@gls@hyp@opt 111
\glslinkcheckfirsthyperhook: new 112
\glslinkvar: new 110
\GLSname: switched to using \@gls@hyp@opt 133
\Glsname: switched to using \@gls@hyp@opt 133
\glsname: switched to using \@gls@hyp@opt 133
\GLSpl: switched to using \@gls@hyp@opt 127
\Glspl: switched to using \@gls@hyp@opt 127
\glspl: switched to using \@gls@hyp@opt 126
\GLSplural: switched to using \@gls@hyp@opt 131
\Glsplural: switched to using \@gls@hyp@opt 131
\glsplural: switched to using \@gls@hyp@opt 131
\glsspace: new 219
\GLSsymbol: switched to using \@gls@hyp@opt 136
\Glssymbol: switched to using \@gls@hyp@opt 135
\glssymbol: switched to using \@gls@hyp@opt 135
\GLSsymbolplural: switched to using \@gls@hyp@opt 136

\Glssymbolplural: switched to using \@gls@hyp@opt 136
\glssymbolplural: switched to using \@gls@hyp@opt 136
\GLStext: switched to using \@gls@hyp@opt 130
\Glstext: switched to using \@gls@hyp@opt 130
\glstext: switched to using \@gls@hyp@opt 129
\glstreenamefmt: new 311
\GLSuseri: switched to using \@gls@hyp@opt 137
\Glsuseri: switched to using \@gls@hyp@opt 137
\glsuseri: switched to using \@gls@hyp@opt 137
\GLSuserii: switched to using \@gls@hyp@opt 138
\Glsuserii: switched to using \@gls@hyp@opt 138
\glsuserii: switched to using \@gls@hyp@opt 137
\GLSuseriii: switched to using \@gls@hyp@opt 139
\Glsuseriii: switched to using \@gls@hyp@opt 139
\glsuseriii: switched to using \@gls@hyp@opt 138
\GLSuseriv: switched to using \@gls@hyp@opt 140
\Glsuseriv: switched to using \@gls@hyp@opt 139
\glsuseriv: switched to using \@gls@hyp@opt 139
\GLSuserv: switched to using \@gls@hyp@opt 140
\Glsuserv: switched to using \@gls@hyp@opt 140
\glsuserv: switched to using \@gls@hyp@opt 140
\GLSuservi: switched to using \@gls@hyp@opt 141
\Glsuservi: switched to using \@gls@hyp@opt 141
\glsuservi: switched to using \@gls@hyp@opt 141
\ifignoredglossary: new 63
altlongragged4col: fixed bug that displayed description instead of symbol 291
\newglossary: added starred version . . 61
\newignoredglossary: new 63
\ns@newglossary: added
  \@glotype@⟨*name*⟩@log 61
  new 61
\p@gls@hyp@opt: new 110
\PGLS: changed to use \@gls@hyp@opt 266
\Pgls: changed to use \@gls@hyp@opt 264
\pgls: changed to use \@gls@hyp@opt 263
\PGLSpl: changed to use \@gls@hyp@opt 266
\Pglspl: changed to use \@gls@hyp@opt 265
\pglspl: changed to use \@gls@hyp@opt 264
\s@gls@hyp@opt: new 110
\s@newglossary: new 61
automake: new 30

4.09 (2014-08-12)
  \glsaddkey: fixed bug in user commands 76

4.10 (2014-08-27)
  \@Gls@acrentryname: new 150
  \@Gls@entryname: new 150
  \@gls@glossary: Renamed \@glossary to \@gls@glossary 180
  \glspercentchar: new 160
  \glstildechar: new 160
  alttree: moved space after symbol 318, 319

4.11 (2014-09-01)
  \@@do@esc@wrglossary: added hook . 185
  sanitize: none option 25
  \gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary 181
  \glsaddprotectedpagefmt: new 182
  \glsbackslash: new 160

4.12 (2014-11-22)
  \@gls@addpredefinedattributes: Added glsignore attribute 47
  \@gls@adjustmode: new 158
  \@gls@notranslatorhook: removed . . . 25
  \@gls@toc: added \protect to \numberline 44
  \@gls@usetranslator: new 25
  \glsacrpluralsuffix: new 35
  \glsadd: added check for vertical mode 158

\glsaddallunused: replaced @gobble with glsignore 159
\glsifusedtranslatordict: new 26
\glsignore: new 159
\glsupacrpluralsuffix: new 35
\ProvidesGlossariesLang: new 35
\RequireGlossariesLang: new 35
4.13 (2015-02-03)
\indexspace: new 273, 293, 311
4.14 (2015-02-28)
\@@glslocalreset: new 92
\@@glslocalunset: new 91
\@@glsreset: new 92
\@@glsunset: new 91
\@@newglossaryentry@defcounters: new 93
\@cGls: new 96
\@cGls@: new 96
\@cGlspl@: new 97
\@cgls: new 96
\@cgls@: new 96
\@cglspl: new 97
\@cglspl@: new 97
\@gls@entry@count: new 96
\@gls@increment@currcount: new . . . 95
\@gls@local@increment@currcount: new 95
\@gls@write@entrycounts: new 95
\@glslocalreset: new 92
\@glslocalunset: new 91
\@glsreset: new 92
\@glsunset: new 91
\@newglossaryentry@defcounters: new 87
\cGls: new 96
\cgls: new 96
\cGlsformat: new 97
\cglsformat: new 96
\cGlspl: new 97
\cglspl: new 97
\cGlsplformat: new 98
\cglsplformat: new 97
\gls@defdocnewglossaryentry: new . 71
\glsenableentrycount: new 93
\glslocalreset: switched to \@glslocalreset 90
\glslocalunset: switched to \@glslocalunset 91
\glsreset: switched to \@glsreset . . . 90
\glsunset: switched to \@glsunset . . . 91
4.15 (2015-03-16)
General: bug fix replaced \@glo@type with \glstype 148
4.16 (2015-06-18)
\glsaddstoragekey: new 74
4.16 (2015-07-08)
\@ACRlong: added \glspostlinkhook 361
\@ACRshort: added \glspostlinkhook 360
\@Acrlong: added \glspostlinkhook 360
\@Acrshort: added \glspostlinkhook 359
\@GLS@: added \glspostlinkhook . . . 126
\@GLSpl: added \glspostlinkhook . . 128
\@Gls@: added \glspostlinkhook . . . 125
\@Glspl@: added \glspostlinkhook . 127
\@acrlong: added \glspostlinkhook 360
\@acrshort: added \glspostlinkhook 359
\@gls@: added \glspostlinkhook . . . 124
\@gls@@link: added \glspostlinkhook 111
\@gls@field@link: added \glspostlinkhook 129
\@gls@link: moved definition of \glsifhyperon outside of this macro 113
\@glsdisp: added \glspostlinkhook 129
\@glspl@: added \glspostlinkhook . 126
General: added \glspostlinkhook 142–149
\glsacspace: new 226
\glsadd: changed \@do@wrglossary to \@@do@wrglossary 158
\glsfielddef: new 79
\glsfieldedef: new 78
\glsfieldfetch: new 79
\glsfieldgdef: new 78
\glsfieldxdef: new 78
\glsifhyperon: moved definition of \glsifhyperon 112
\glslinkpostsetkeys: new 112
\glspostlinkhook: new 111
\glswriteentry: new 181
\ifglsfieldcseq: new 80
\ifglsfielddefeq: new 80
\ifglsfieldeq: new 79
long-sp-short: new 225
\showglofield: new 258
4.18 (2015-09-09)
General: split mfirstuc into separate bundle 4

4.19 (2015-10-31)
\glstreenamebox: new 317
4.19 (2015-11-22)
\@gls@link@nocheckfirsthyper: new 129
\@gls@preglossaryhook: new 191
\@printglossary: added
\@gls@preglossaryhook 192
\do@glsdisablehyperinlist: new . . 112
\doifglossarynoexistsordo: new . . . 56
\gls@gobbleopt: new 60
\glsdoifexistsordo: new 55
4.20 (2015-11-30)
\@gls@link: added
\@gls@setdefault@glslink@opts 112
added \glsdonohyperlink when hyperlink is suppressed 113
\@gls@setdefault@glslink@opts: new 112
\gls@checkseeallowed@preambleonly: new 66
\glsdonohyperlink: new 122
4.21 (2016-01-24)
\@printglossary: warn if no style has been set 191
General: changed checkfirsthyper assignment 142–148
\glossarystyle: set default style if not already set 214
\glsLTpenaltycheck: new 286
\glspatchLToutput: new 286
\glspenaltygroupskip: new 286
altlong4col-booktabs: new 284
altlongragged4col-booktabs: new . 285
long-booktabs: new 283
long3col-booktabs: new 283
long4col-booktabs: new 284
longragged-booktabs: new 285
longragged3col-booktabs: new 285
\setglossarystyle: set default style if not already set 213
4.22 (2016-04-19)
\@@do@esc@wrglossary: added check for \@arabic 185
added test to allow temporary primitive modifications and added arabic case 185
mcolalttreespannav: new 298
mcolindexspannav: new 294
mcoltreenonamespannav: new 296
mcoltreespannav: new 295
\gls@arabicpage: new 182
\gls@protected@pagefmts: added arabic to list 181
\glsentrytitlecase: new 153
\glsfindwidesttoplevelname: new . 317
\glslistgroupheaderfmt: new 273
\glslistnavigationitem: new 273
\glstreegroupheaderfmt: new 311
\glstreenavigationfmt: new 311
\ifglswrallowprimitivemods: new . 183
list: fixed missing space before description 273
long: fixed typo in \glossentrydesc . 277
super4col: fixed bug in \glossentry . 302
4.23 (2016-04-30)
\glscurrentfieldvalue: new 59
\ifglshasfield: added
\glscurrentfieldvalue 58, 59
altlongragged4col: check for nogroupskip changed 291
altsuperragged4col: check for nogroupskip changed 310
long: check for nogroupskip changed . . 277
long-booktabs: check for nogroupskip changed 283
long3col: check for nogroupskip changed 279
long3col-booktabs: check for nogroupskip changed 284
long4col: check for nogroupskip changed 280
long4col-booktabs: check for nogroupskip changed 284
longragged: check for nogroupskip changed 288
longragged3col: check for nogroupskip changed 289
super: check for nogroupskip changed . 299
super3col: check for nogroupskip changed 301
super4col: check for nogroupskip changed 303
superragged: check for nogroupskip changed 306
superragged3col: check for nogroupskip changed 308
4.24 (2016-05-27)
\@gls@extramakeindexopts: new . . . 170

\@gls@glossary: added check for debug mode . . . 180
\@gls@see@noindex: new . . . 6
debug: new . . . 5
seenoindex: new . . . 6
\glsnomakeindexwarning: new . . . 44
\GlsSetQuote: new . . . 167
\GlsSetWriteIstHook: new . . . 167
4.25 (2016-06-09)
\@gls@enablesavenonumberlist: new 67
\@gls@initnonumberlist: new . . . 67
\@gls@savenonumberlist: new . . . 67
4.26 (2016-10-12)
\@glossary@default@style: added check for classicthesis . . . 8
mcolindex: replaced \@idxitem with \glstreeitem . . . 293
mcolindexspannav: replaced \@idxitem with \glstreeitem . . . 294
\glstreechildpredesc: new . . . 312
\glstreeitem: new . . . 311
\glstreepredesc: new . . . 312
\glstreesubitem: new . . . 312
\glstreesubsubitem: new . . . 312
4.28 (2017-01-07)
\glspatchtabularx: new . . . 90
4.29 (2017-01-19)
\@gls@noidx@do: current letter group assignment made global . . . 202
\@print@noidx@glossary: moved definition of \@gls@currentlettergroup outside of theglossary environment . . . 200
General: added check for \@glsxtr@doaccsupp . . . 341
\glsnavhyperlinkname: new . . . 268
4.30 (2017-06-11)
\@glo@autosee: new . . . 87
\@glo@autoseehook: new . . . 87
\@glo@check@sortallowed: new . . . 13
\@gls@noidx@do: letter group assignment made global . . . 202
\@gls@setupsort@def: added check for register . . . 14
\@gls@setupsort@none: new . . . 15
\@xdycrossrefhook: new . . . 49
\@xdylocationclassorder: bug fix: changed \edef to \def . . . 50
\glosortentrieswarning: new . . . 20
\gls@set@xr@key: new . . . 66
\gls@xr@key: new . . . 66
\GlsAddXdyLocation: bug fix: changed #1 to #2 . . . 50
\glsnoidxstripaccents: added \a . . . 23
added \TH, \dh and \DH . . . 23
4.31 (2017-08-10)
nolist: added check for "list" style . . . 10
4.31 (2017-09-10)
style: changed \renewcommand to \def . 8
4.32 (2017-08-24)
\@glsnavhypertarget: new . . . 268
\@glsshowtarget: new . . . 6
\glsshowtarget: new . . . 6
4.33 (2017-09-20)
\@@do@esc@wrglossary: added \gls@the and \gls@number . . . 185
renamed from \@@do@esc@wrglossary . . . 184
\@@do@noesc@wrglossary: new . . . 183
\@@do@wrglossary: changed to check for esclocations . . . 183
\@gls@missinglang@warn: new . . . 20
\GlsSetXdyFirstLetterAfterDigits: added starred version . . . 160
\GlsSetXdyNumberGroupOrder: new . 160
esclocations: new . . . 9
4.34 (2017-11-03)
mcolalttreespannav: removed spurious space . . . 298
\glsshowtarget: modified to check for math mode and inner . . . 6
4.35 (2017-11-14)
\glsadd: added \@gls@setsort (in case of sort=use) . . . 158
4.36 (2018-03-07)
\@gls@glossary: removed \index . . . 180
4.37 (2018-04-07)
\gls@begindocdefs: new . . . 72
4.38 (2018-05-10)
\@gls@define@glossaryentrycounter: added check for existence of glossaryentry counter . . . 11
new . . . 11
\@gls@define@glossarysubentrycounter: new . . . 12
prepended \currentglossary. to \theHglossarysubentry and removed spurious eol space . . . 12

\glsaccsupp: added braces around actual text argument 347
\glsentrycounterlabel: bug fix: move conditional inside command 207
\GlsEntryCounterLabelPrefix: new 205
\glsentryitem: bug fix: move conditional inside command 207
\glsrefentry: bug fix: move conditional inside command 207
\glsresetsubentrycounter: bug fix: move conditional inside command . 206
\glsstepentry: bug fix: move conditional inside command 207
\glsstepsubentry: bug fix: move conditional inside command 207
\glssubentrycounterlabel: bug fix: move conditional inside command . 207
\glssubentryitem: bug fix: move conditional inside command 208
\showglonameaccess: bug fix: corrected field (was showing text access field) 377

4.40 (2018-06-01)
\istfile: changed \def to \providecommand 178
\makenoidxglossaries: false 175

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

\! 118, 119
\" 23, 116–119, 121
\# 163, 164
\% 160, 166, 325, 326
\& 35, 157
\' 23
\. 10, 23
\= 23
\? 116–118, 168
\@ 72
\@@delimN 216
\@@do@@wrglossary 175, 183, 186
\@@do@esc@wrglossary 183
\@@do@noesc@wrglossary 183
\@@do@wrglossary 158, 181
\@@glo@assign@sortkey 176
\@@glo@list 53
\@@glo@sort 23
\@@glo@type 190, 191
\@@glossarysec 7, 42, 43
\@@glossaryseclabel 8, 42, 43, 204
\@@glossarysecstar 8, 42, 43, 204
\@@gls@checkactual 120
\@@gls@checkbar 119
\@@gls@checkescactual 118
\@@gls@checkescbar 118
\@@gls@checkesclevel 119
\@@gls@checkescquote 117, 169, 170
\@@gls@checklevel 120
\@@gls@checkquote 116, 117, 167, 168
\@@gls@default@entryfmt 99, 108
\@@gls@expand@field . . 21, 70, 71, 75, 76, 237, 239–241, 244, 246, 249–251, 372–376
\@@gls@extramakeindexopts . . 167, 172, 173
\@@gls@fixbraces 188
\@@gls@noexpand@field 21, 69, 70
\@@gls@noidx@no@sanitizesort 22, 23
\@@gls@noidx@nosanitizesort 177
\@@gls@nosanitizesort 22, 177
\@@gls@sanitizesort 22, 177
\@@gls@xdycheckbackslash 121, 122
\@@gls@xdycheckquote 121
\@@glslocalreset 92, 94
\@@glslocalunset 91, 94
\@@glsreset 92, 94
\@@glsunset 91, 94
\@@newglossaryentry@defcounters 93
\@@this@glo@ 54
\@ACRfull 220
\@ACRfullpl 221
\@ACRlong 146, 220
\@ACRlongpl 148, 221
\@ACRshort 143, 220
\@ACRshortpl 144, 145, 221
\@Acrfull 220
\@Acrfullpl 221
\@Acrlong 146, 220
\@Acrlongpl 148, 221
\@Acrshort 142
\@Acrshortpl 144
\@Alph 182, 185
\@GLS 125
\@GLS@ 125, 266
\@GLSdesc 134
\@GLSdesc@ 134
\@GLSdescplural 135
\@GLSdescplural@ 135
\@GLSfirst 131
\@GLSfirst@ 131
\@GLSfirstplural 132
\@GLSfirstplural@ 132
\@GLSname 133
\@GLSname@ 133

\@GLSpl . . . 127
\@GLSpl@ . . . 127, 128, 267
\@GLSplural . . . 131, 132
\@GLSplural@ . . . 132
\@GLSsymbol . . . 136
\@GLSsymbol@ . . . 136
\@GLSsymbolplural . . . 136
\@GLSsymbolplural@ . . . 136, 137
\@GLStext . . . 130
\@GLStext@ . . . 130
\@GLSuseri . . . 137
\@GLSuseri@ . . . 137
\@GLSuserii . . . 138
\@GLSuserii@ . . . 138
\@GLSuseriii . . . 139
\@GLSuseriii@ . . . 139
\@GLSuseriv . . . 140
\@GLSuseriv@ . . . 140
\@GLSuserv . . . 140
\@GLSuserv@ . . . 140, 141
\@GLSuservi . . . 141
\@GLSuservi@ . . . 141
\@Gls . . . 124
\@Gls@ . . . 94, 96, 124, 265
\@Gls@acrentryname . . . 222
\@Gls@entry@field . . . 76, 150–155
\@Gls@entryname . . . 150, 222
\@GlsSetXdyFirstLetterAfterDigits . 160
\@GlsSetXdyNumberGroupOrder . . . 160, 161
\@Glsdesc . . . 134
\@Glsdesc@ . . . 134
\@Glsdescplural . . . 134, 135
\@Glsdescplural@ . . . 135
\@Glsfirst . . . 130
\@Glsfirst@ . . . 130, 131
\@Glsfirstplural . . . 132
\@Glsfirstplural@ . . . 132
\@Glsname . . . 133
\@Glsname@ . . . 133
\@Glspl . . . 127
\@Glspl@ . . . 95, 97, 127, 265, 266
\@Glsplural . . . 131
\@Glsplural@ . . . 131
\@Glssymbol . . . 135
\@Glssymbol@ . . . 135
\@Glssymbolplural . . . 136
\@Glssymbolplural@ . . . 136
\@Glstext . . . 130
\@Glstext@ . . . 130
\@Glsuseri . . . 137
\@Glsuseri@ . . . 137
\@Glsuserii . . . 138
\@Glsuserii@ . . . 138
\@Glsuseriii . . . 139
\@Glsuseriii@ . . . 139
\@Glsuseriv . . . 139
\@Glsuseriv@ . . . 139
\@Glsuserv . . . 140
\@Glsuserv@ . . . 140
\@Glsuservi . . . 141
\@Glsuservi@ . . . 141
\@Mi . . . 286
\@PGLS . . . 266
\@PGLS@ . . . 266
\@PGLSpl . . . 266
\@PGLSpl@ . . . 266
\@Pgls . . . 264
\@Pgls@ . . . 264
\@Pglspl . . . 265
\@Pglspl@ . . . 265
\@Roman . . . 182, 185
\@acrfull . . . 219
\@acrfullpl . . . 220
\@acrlong . . . 145, 219
\@acrlongpl . . . 147, 221
\@acrshort . . . 142, 219, 220
\@acrshortpl . . . 143, 221
\@addtoacronymlists . . . 17, 18
\@after . . . 18
\@afterheading . . . 274, 329
\@alph . . . 182, 185
\@arabic . . . 182, 185
\@auxout . . . 60, 61, 95, 172, 173, 175, 178, 190, 193, 194, 269
\@backslashchar . . . 115, 121, 122
\@before . . . 18
\@bsphack . . . 180
\@cGls . . . 96
\@cGls@ . . . 94, 96
\@cGlspl . . . 97
\@cGlspl@ . . . 95, 97
\@cclv . . . 286, 287
\@cgls . . . 96
\@cgls@ . . . 94, 96
\@cglspl . . . 97
\@cglspl@ . . . 94, 97
\@chapter . . . 33
\@classoptionslist . . . 32

\@closegls 171, 172
\@colht 286
\@colroom 286, 287
\@currentlabelname 8, 204
\@curroptions 32
\@declaredoptions 32
\@delimN 215
\@delimR 215
\@disable@onlypremakeg 173
\@disable@premakecs 34
\@disabled@glsaddxdycounters 46
\@do@addcounter 45
\@do@auxoutstuff 193, 194
\@do@glossentry 209, 341, 342
\@do@gls@getcounterprefix 183, 186
\@do@gls@islistofacronyms 18
\@do@glssee 87
\@do@ifinlist 44, 45
\@do@newglossaryentry 223, 237, 239–241, 243–246, 248–253, 372–376
\@do@seeglossary 175, 188
\@do@subglossentry 210, 342
\@do@wrglossary 113
\@do@writeaux@info 189, 190
\@ehc 286
\@empty 11, 15, 17, 30, 32, 33, 44, 46, 49, 52, 53, 83, 84, 89, 113, 114, 124–128, 142–148, 161, 164, 166, 167, 171, 172, 179, 180, 187, 213, 238, 240, 242, 244–247, 249, 251, 253, 323, 325, 327, 359–361
\@end@fixbraces 188
\@endfortrue 27, 56, 74, 269
\@esphack 181
\@expandtwoargs 32
\@firstofone 23
\@firstofthree 110, 123, 124, 126, 128, 142, 144, 145, 147, 359–361
\@firstoftwo 26, 27, 73, 74, 110, 126–128, 144, 145, 147, 148
\@for 27, 32, 34, 45, 46, 53, 73, 74, 115, 161–163, 173, 174, 182, 189, 194, 195, 224, 238, 240, 242, 244, 246, 249, 251, 253, 269, 270, 323
\@glo@@desc 86
\@glo@@symbol 86
\@glo@access 342, 344, 345, 347
\@glo@addchildren 194, 199
\@glo@assign@sortkey 174, 176, 205
\@glo@autosee 87
\@glo@autoseehook 87
\@glo@check@mkidxrangechar 114, 115, 186, 322, 323
\@glo@check@sortallowed . . 13–15, 174, 177
\@glo@childlist 195
\@glo@counter 66, 82, 85
\@glo@counterprefix 178, 183, 186, 187, 213, 216
\@glo@default@sorttype . . 12, 176, 197, 198
\@glo@defaultcounter 85
\@glo@desc 64, 81–83, 86
\@glo@descaccess 343–345
\@glo@descplural 64, 81, 82
\@glo@descpluralaccess 343–345
\@glo@do@sortentries 194
\@glo@entry 159
\@glo@entryprefix 261
\@glo@entryprefixfirst 261
\@glo@entryprefixfirstplural . . 261, 262
\@glo@entryprefixplural 261
\@glo@esclabel 88, 89
\@glo@etext 99–101
\@glo@first 65, 82, 85, 86, 248, 376
\@glo@firstaccess 342, 344, 345
\@glo@firstplural 65, 82, 85, 376
\@glo@firstpluralaccess 343–345
\@glo@grabfirst 200
\@glo@label 68, 69, 75, 76, 78–80, 82–87, 93, 157, 261, 262, 317, 345
\@glo@list 87
\@glo@long 57, 68, 83, 85
\@glo@longaccess 343–345
\@glo@longpl 68, 83, 85, 237, 239, 241, 243, 246, 248, 250, 372
\@glo@longpluralaccess 343–345
\@glo@name 13, 64, 69, 82, 84–86
\@glo@no@assign@sortkey 174
\@glo@nonumberlist 67
\@glo@numfmt 187, 323
\@glo@parent . . 15, 66, 82, 84, 85, 89, 195, 196
\@glo@plural 64, 82, 84, 85, 374
\@glo@pluralaccess 343–345
\@glo@prefix 9, 66, 67, 82, 89, 114, 115, 186, 322, 323
\@glo@range 186, 322, 323
\@glo@see 66, 82, 87
\@glo@seeautonumberlist 9, 66
\@glo@short 57, 58, 68, 83, 85, 375
\@glo@shortaccess 343–345, 372–375

\@glo@shortpl 68, 83, 85, 237, 239, 241, 243, 245, 248, 250, 372, 375
\@glo@shortpluralaccess 343–345
\@glo@sort 13, 15, 22, 23, 64, 82, 85, 89
\@glo@sortedinsert 195, 196
\@glo@sortentries 197, 198
\@glo@sorthandler@case 198
\@glo@sorthandler@letter 197
\@glo@sorthandler@nocase 198
\@glo@sorthandler@word 197
\@glo@sortinghandler 194, 196
\@glo@sortinglist 194, 196, 198, 199
\@glo@sorttype 176, 199, 201, 205
\@glo@storeentry 13–15
\@glo@suffix 114, 115, 186, 323
\@glo@symbol 57, 65, 82, 86, 243, 248, 344
\@glo@symbolaccess 343–345, 375
\@glo@symbolplural 65, 82, 86
\@glo@symbolpluralaccess 343–345
\@glo@text 64, 82, 85, 86, 124–129, 149–151, 243, 262, 374
\@glo@textaccess . . . 342, 344, 345, 372–375
\@glo@thislabel 88
\@glo@thislettergrp 200–202
\@glo@thisvalue 58, 59
\@glo@tmp 75, 76, 187
\@glo@type 8, 15, 65, 82, 83, 85–87, 158, 159, 173, 179, 191–195, 199, 200, 203, 204, 222, 238, 240, 242, 244, 246, 249, 251, 253, 268, 270
\@glo@types 53, 54, 61, 92, 93, 158, 159, 173, 174, 259, 317
\@glo@useri 67, 82, 85
\@glo@userii 67, 82, 85
\@glo@useriii 68, 82, 85
\@glo@useriv 68, 82, 85
\@glo@userv 68, 82, 85
\@glo@uservi 68, 82, 85
\@glodesc 86
\@glolist@ 83
\@gloname 86
\@glossary@default@style 8, 10, 191, 213, 214, 254
\@glossaryentryfield 89
\@glossarysection 41
\@glossarystyle 191, 192, 204
\@glossarysubentryfield 89
\@gls 123
\@gls@ 94, 96, 123, 264, 265
\@gls@@link 111
\@gls@Hcounter 113, 114
\@gls@ReturnAfterFi 216
\@gls@access@display 347–349
\@gls@actualchar . 89, 118, 120, 165, 166, 326
\@gls@addpredefinedattributes . 161, 170
\@gls@adjustmode 158
\@gls@automake 174
\@gls@between 270
\@gls@body 150
\@gls@checkactual 116, 168
\@gls@checkbar 116, 168
\@gls@checkedmkidx 115–121, 167–169
\@gls@checkescactual 116, 168
\@gls@checkescbar 116, 169
\@gls@checkescquote 116, 168, 169
\@gls@checklevel 116, 169
\@gls@checkmkidxchars 88, 89, 114, 168, 175, 186, 188, 322, 323
\@gls@checkquote 116, 167, 168
\@gls@classI 162
\@gls@classII 162
\@gls@codepage 194
\@gls@counter 109, 112–114, 158, 178, 186, 187, 323
\@gls@counterwithin 11, 12
\@gls@ctr 45
\@gls@currentlettergroup 200–202
\@gls@debugfalse 5
\@gls@debugtrue 5
\@gls@declareoption 9, 10, 16, 17, 20, 21, 26, 29, 31
\@gls@default 98
\@gls@default@value 57–59, 69, 70, 82, 84–86, 247, 261
\@gls@deffile 72, 73
\@gls@define@glossaryentrycounter 12, 33, 205, 206
\@gls@define@glossarysubentrycounter 33, 205, 206
\@gls@defsort 13–15, 86
\@gls@defsortcount 13–15, 62
\@gls@do@acronymsdef 16, 17, 32, 33, 63
\@gls@do@indexdef 31–33, 63
\@gls@do@numbersdef 31–33, 63
\@gls@do@symbolsdef 31, 63
\@gls@do@symbolssdef 32, 33
\@gls@doautomake 30, 174
\@gls@docheckquotedef 167–170

\@gls@docloadedfalse 4
\@gls@docloadedtrue 4
\@gls@dodeflistparser 174
\@gls@doentrycounterdef 32, 33
\@gls@doentrydef 108
\@gls@dolast 188, 189
\@gls@donext 189
\@gls@donext@def 157
\@gls@dosubentrycounterdef 33
\@gls@dothiswrite 171, 172
\@gls@elem 269
\@gls@enablesavenonumberlist 72
\@gls@encapchar . . . .
. . . . 118, 119, 165, 166, 187, 188, 323, 326
\@gls@entry@count 95
\@gls@entry@field . . . .
. . . . 75, 76, 93, 94, 150–156, 345–347
\@gls@escbsdq 116, 166, 327
\@gls@expand@fields 70, 71
\@gls@expandonce 71
\@gls@extramakeindexopts 173
\@gls@fetchfield 58
\@gls@field@link 76, 77, 129–141
\@gls@firsttok 200
\@gls@fixbraces 87
\@gls@forbidtexext 61
\@gls@get@counterprefix 187
\@gls@getbody 150
\@gls@getcounterprefix 183, 186
\@gls@getgrouptitle 175, 212, 270
\@gls@glossary 180
\@gls@gobbleopt 60
\@gls@grptitle 212, 268, 270
\@gls@hyp@opt . . . .
77, 96, 97, 111, 123–148, 219–221, 263–266
\@gls@hyp@opt@cs 110
\@gls@hypergroup 269
\@gls@ifinlist 45
\@gls@ifnotmeasuring 90
\@gls@igtype 63
\@gls@increment@currcount 94
\@gls@indexdef 31
\@gls@initnonumberlist 67, 82
\@gls@islistofacronyms 18
\@gls@keylist 371
\@gls@keymap 67, 73–76, 261, 344
\@gls@label 175, 178, 183, 186, 187
\@gls@langmod 171
\@gls@levelchar . . 89, 119, 120, 165, 166, 326
\@gls@link . . 111, 124–129, 142–149, 359–361
\@gls@link@checkfirsthyper 124–128
\@gls@link@label 112, 239, 245
\@gls@link@nocheckfirsthyper 129, 142–148
\@gls@link@opts 112, 239, 245
\@gls@list 269, 270
\@gls@listsuffix 44
\@gls@loadlist 10, 253
\@gls@loadlong 9, 10, 254
\@gls@loadsuper 10, 254
\@gls@loadtree 10, 254
\@gls@local@increment@currcount 94
\@gls@loclist 176, 177, 201, 202
\@gls@map 73, 74
\@gls@missinglang@warn 20, 36
\@gls@missingnumberlist 86
\@gls@noaccess 347
\@gls@noexpand@fields 71
\@gls@nohyperlist 19, 63, 112
\@gls@noidx@do 200
\@gls@noidx@getgrouptitle 175
\@gls@noidx@sanitizesort 22, 177
\@gls@noidx@setsanitizesort 24, 177
\@gls@noidxloclist@finalsep . . . 176, 177
\@gls@noidxloclist@prev 176, 177, 202, 203
\@gls@noidxloclist@sep 176, 202, 203
\@gls@noref@warn 175, 200
\@gls@numberlink 216
\@gls@numbersdef 31
\@gls@numlist@lastsep 157
\@gls@numlist@nextsep 157
\@gls@numlist@sep 157
\@gls@old@chapter 33
\@gls@oldnewglossaryentryposthook . 344
\@gls@oldnewglossaryentryprehook . . 344
\@gls@onlypremakeg 33, 34
\@gls@order 171, 172
\@gls@org@LT@output 286
\@gls@org@glsnoidxdisplayloc 177
\@gls@org@glsseeformat 177
\@gls@patchtabularx 90
\@gls@preglossaryhook 192
\@gls@prevlevel . 297, 298, 317–320, 335, 336
\@gls@provide@newglossary 61
\@gls@quotechar 117–120, 165–169, 326
\@gls@reference 175, 178
\@gls@removespaces 216
\@gls@renewglossary 171
\@gls@replacementtext 347

\@gls@rest . . . 150
\@gls@restoreat . . . 72, 73
\@gls@roman . . . 48, 323, 324
\@gls@sanitized@tmp . . . 115
\@gls@sanitizedesc . . . 28
\@gls@sanitizesort . . . 13
\@gls@sanitizesymbol . . . 28
\@gls@saveentrycounter . . . 113, 158
\@gls@savenonumberlist . . . 66, 67
\@gls@see@noindex . . . 7, 66
\@gls@setacrstyle . . . 28, 32, 33
\@gls@setcounter . . . 62
\@gls@setdefault@glslink@opts . . . 112
\@gls@setsort . . . 13–15, 113, 158
\@gls@setupshortcuts . . . 32, 33
\@gls@sort . . . 201
\@gls@sort@A . . . 196, 197
\@gls@sort@B . . . 196, 197
\@gls@startswithexpandonce . . . 70
\@gls@storenonumberlist . . . 67, 86
\@gls@symbolsdef . . . 31
\@gls@this . . . 182
\@gls@thisHloc . . . 187
\@gls@thisfield . . . 58, 59
\@gls@thislabel . . . 56, 189, 198, 199
\@gls@thislist . . . 157
\@gls@thisloc . . . 187
\@gls@thisval . . . 74
\@gls@title . . . 41
\@gls@tmp . . . 15, 36, 49, 71, 115, 181, 269, 270
\@gls@tmpb . . . 116–121, 167–169
\@gls@toc . . . 42, 43
\@gls@type . . . 174, 224, 238, 240, 242, 244, 246, 249, 251, 253, 317
\@gls@updatechecked . . . 115, 116, 168, 169
\@gls@usetranslator . . . 26, 27, 36
\@gls@value . . . 70, 153
\@gls@warnonglossdefined . . . 21, 190
\@gls@warnontheglossdefined . . . 21, 208
\@gls@write@entrycounts . . . 95
\@gls@writedef . . . 72
\@gls@writeisthook . . . 165, 167
\@gls@xdy@locationlist . . . 162
\@gls@xdycheckbackslash . . . 115
\@gls@xdycheckquote . . . 115
\@gls@xref . . . 188
\@glsAlphacompositor . . . 39, 49, 324
\@glsHlocref . . . 183, 186
\@glsacronymlists . . 17–19, 53, 222, 224, 238, 240, 242, 244, 246, 249, 251, 253, 258
\@glsaddkey . . . 75, 76
\@glsaddstoragekey . . . 74, 75
\@glsaddxdyattribute . . . 45, 46
\@glsdefaultsort . . . 13
\@glsdesc . . . 133
\@glsdesc@ . . . 133, 134
\@glsdescplural . . . 134
\@glsdescplural@ . . . 134
\@glsdisp . . . 128
\@glsentry . . . 92, 93, 95
\@glsentrytitlecase . . . 153
\@glsfirst . . . 130
\@glsfirst@ . . . 130
\@glsfirstletter . . . 160
\@glsfirstplural . . . 132
\@glsfirstplural@ . . . 132
\@glshypernumber . . . 215
\@glsisacronymlistfalse . . . 18
\@glsisacronymlisttrue . . . 18
\@glslink . . . 113, 123, 157, 268
\@glslocalreset . . . 91, 94
\@glslocalunset . . . 91, 94
\@glslocref . . . 178, 183, 185–187, 322, 323
\@glsminrange . . . 161, 162, 324
\@glsname . . . 133
\@glsname@ . . . 133
\@glsnavhypertarget . . . 268
\@glsnextpages . . . 192
\@glsnodesc . . . 82, 83, 86
\@glsnoname . . . 82, 84, 86
\@glsnonextpages . . . 192
\@glsnumberformat . . . 109, 112, 158, 178, 186, 187, 322, 323
\@glsopenfile . . . 171, 179
\@glsorder . . . 172, 173
\@glspl . . . 126
\@glspl@ . . . 94, 97, 126, 264–266
\@glsplural . . . 131
\@glsplural@ . . . 131
\@glsreset . . . 90, 94
\@glssee . . . 87, 188
\@glsshowtarget . . . 5, 6, 122
\@glssymbol . . . 135
\@glssymbol@ . . . 135
\@glssymbolplural . . . 136
\@glssymbolplural@ . . . 136
\@glstarget . . . 123, 208, 269

\@glstext . . . 129
\@glstext@ . . . 129
\@glsunset . . . 91, 94
\@glsuseri . . . 137
\@glsuseri@ . . . 137
\@glsuserii . . . 137, 138
\@glsuserii@ . . . 138
\@glsuseriii . . . 138
\@glsuseriii@ . . . 138
\@glsuseriv . . . 139
\@glsuseriv@ . . . 139
\@glsuserv . . . 140
\@glsuserv@ . . . 140
\@glsuservi . . . 141
\@glsuservi@ . . . 141
\@glswidestname . . . 317–319, 335
\@glswritefiles . . . 30
\@glsxtr@doaccsupp . . . 341
\@gobble . . . 5, 13–15, 73, 90, 115, 159, 160, 163, 164, 175, 321, 325, 326
\@idxitem . . . 311
\@ifclassloaded . . . 4, 11, 41
\@ifnextchar . . . 62, 110
\@ifpackageloaded . . . 4, 8, 25–27, 36, 52, 90, 156, 167, 341
\@ifstar . . . 61, 74, 75, 110, 160, 218
\@ifundefined . . . 35, 269, 276, 287, 298, 305, 318, 319, 335, 349
\@ignored@glossaries . . . 63
\@input@ . . . 193
\@istfilename . . . 172, 173
\@makecol . . . 286, 287
\@makeglossary . . . 173
\@minus . . . 273, 293, 311
\@mkboth . . . 41, 42
\@newglossary . . . 60, 61
\@newglossaryentry@defcounters . . 87, 93
\@newglossaryentryposthook . . . 75, 76, 87, 261, 344
\@newglossaryentryprehook . . . 75, 76, 81, 83, 261, 344
\@nil . . . 18, 87, 114–116, 150, 168, 169, 186, 188, 200, 201, 215, 216, 322, 323
\@nnil . . . 18, 189
\@no@makeglossaries . . . 174, 176
\@no@post@desc . . . 328
\@nopostdesc . . . 192
\@onelevel@sanitize . . . 22, 48, 73, 89, 115, 160, 161, 164, 188, 190, 200, 324, 325
\@onlypreamble . . . 62, 72, 81, 95, 98, 174, 177
\@onlypremakeg . . . 38, 39, 45, 46, 50, 62, 167
\@org@glossaryentrynumbers . . . 191, 192
\@org@gls@assign@descplural . . . 237, 246, 248–251, 372, 375, 376
\@org@gls@assign@firstpl . . . 237, 239–241, 243, 244, 246, 248–251, 372–376
\@org@gls@assign@plural . . . 237, 239–241, 243, 244, 246, 248–251, 372–376
\@org@gls@assign@symbolplural . . 237, 239–241, 243, 244, 249–251, 373, 374, 376
\@org@glsnumberformat . . . 157
\@org@newglossaryentryprehook . . . 81
\@outputpage . . . 286, 287
\@p@glossarysection . . . 41
\@pgls . . . 263
\@pgls@ . . . 263
\@pglspl . . . 264
\@pglspl@ . . . 264
\@plus . . . 273, 293, 311
\@print@glossary . . . 190
\@print@noidx@glossary . . . 190
\@printgloss@setsort . . . 174, 176, 191
\@printglossary . . . 190
\@roman . . . 48, 323
\@secondofthree . . . 110, 123, 125, 127, 142, 144, 146, 148, 359
\@secondoftwo . 23, 26, 27, 36, 73, 74, 122–125, 128, 142, 143, 145–147, 359–361, 379
\@set@glo@numformat . . . 187, 323
\@sglsaddkey . . . 75
\@sglsaddstoragekey . . . 74, 75
\@thirdofthree . . . 110, 125, 128, 143, 145, 147, 148, 359
\@this@attr . . . 163, 164
\@this@childlabel . . . 195
\@this@counter . . . 46
\@this@ctr . . . 163
\@this@key . . . 74
\@this@label . . . 194
\@thiscs . . . 34
\@tmp . . . 48, 324
\@use@ption . . . 32
\@warn@nomakeglossaries . . . 172, 194
\@wrglossary@pageformat . . . 182
\@wrglossarynumberhook . . . 182, 185
\@xdy@main@language . . . 29, 171, 193
\@xdyattributelist . . . 46, 163
\@xdyattributes . . . 46, 162, 321, 323

\@xdycounters 45, 46, 163
\@xdycrossrefhook 163
\@xdylanguage 193
\@xdylettergroups 53, 165, 326
\@xdylocationclassorder 50, 163, 325
\@xdylocref 46, 165, 321, 325
\@xdynumbergrouporder 52, 160, 161
\@xdyrequiredstyles 51, 161, 323
\@xdysortrules 51, 165, 326
\@xdystyle 161, 162, 323
\@xdyuseralphabets 47, 162, 323
\@xdyuserlocationdefs 49, 50, 163, 322, 324
\@xdyuserlocationnames 50, 322
\@xfor@nextelement 189
\\ 88, 115, 160, 166, 215, 216, 326, 327, 329–331, 339, 340
\{ 73, 159, 166, 321, 326, 327
\} 73, 159, 166, 321, 327
\^ 23
\' 23
\| 116, 118, 169
\~ 23

A

\a 23
\AA 23
\aa 23
accsupp package 341
\accsuppglossaryentryfield 341
\accsuppglossarysubentryfield 342
\acrfootnote 239, 245
\Acrfull 236
\acrfull 236
\ACRfullfmt 220, 223, 231, 233, 367, 369
\Acrfullfmt 220, 223, 231, 233, 367, 369
\acrfullfmt 219, 223, 231, 233, 367, 369
\acrfullformat 156, 219, 237, 252
\Acrfullpl 236
\acrfullpl 236
\ACRfullplfmt . . . 221, 223, 231, 233, 367, 369
\Acrfullplfmt . . . 221, 223, 231, 233, 367, 369
\acrfullplfmt . . . 221, 223, 231, 233, 367, 369
\acrlinkfootnote 238
\acrlinkfullformat 219–221
\Acrlong 236
\acrlong 235
\Acrlongpl 236
\acrlongpl 235
\acrnameformat 243, 374
\acronymentry 223, 225–230, 232–235, 363–365, 368–371
\acronymfont 105, 142–145, 150, 156, 222, 223, 225–235, 239, 240, 242, 244, 245, 247–249, 356, 357, 359–361, 363–365, 367–371, 373–375
\acronymname 16, 17, 37
\acronymsort 223, 225–230, 232–235, 363–365, 368, 369, 371
\acronymtype 16, 17, 222–224, 237–246, 248–251, 253, 372–375
\acrpluralsuffix 223, 225–228, 232–234, 237–241, 243–250, 252, 253, 363, 364, 368–370, 372–376
\Acrshort 235
\acrshort 235
\Acrshortpl 235
\acrshortpl 235
\addcontentsline 44
\addglossarytocaptions 36
\addtolength 319, 335
\advance 14, 15, 84, 113, 286
\AE 23
\ae 23
amsgen package 4, 109
amsmath package 90
\andname 189
\AnyTrackedLanguages 36, 379
\appto 19, 67, 75, 76, 261, 344
array package 283, 287, 305
article class 187
\AtBeginDocument 17, 52, 72, 90, 158, 175
\AtEndDocument 30, 72, 95, 174, 178, 193, 194, 269

B

\b 23
babel package 25, 34, 36, 51
\begin . . . 164, 200, 273, 277–282, 285–311, 325
\BeginAccSupp 347
\begingroup 5, 180, 185, 216
\bfseries 278–281, 283, 284, 288–290, 292, 300–305, 307–311
\bgroup 23, 81, 156, 191, 195
bib2gls 184
booktabs package 282–285
\boolean 252
\boolfalse 30
\booltrue 30

\bottomrule 283, 284
\box 286

C

\c 23
\c@equation 113
\c@glossaryentry 11
\c@glossarysubentry 12
\c@page 182, 183, 185
\catcode 72
\cGls 96
\cgls 96
\cGlsformat 94
\cglsformat 94
\cGlspl 97
\cglspl 97
\cGlsplformat 95
\cglsplformat 94
\char 212
classicthesis package 8
\cleardoublepage 43
\clearpage 43
\closeout 72, 165, 167, 171, 179
\compatglossarystyle 328–340
\compatibleglossentry 210
\compatiblesubglossentry 210
\copy 286, 287
\count@ 200, 201
\csdef 21, 75–77, 87, 88, 93, 94, 194, 195, 214, 224, 225, 327
\csedef 95, 182
\csgdef 40, 60, 63, 94, 95, 190, 203
\cslet 67, 81, 88, 199
\csname 12–15, 32, 34, 36, 37, 42, 43, 46, 48, 49, 52, 53, 56, 61, 62, 69, 70, 75–80, 83–89, 91, 92, 108, 112–114, 124–129, 142–149, 157, 158, 162–164, 168–171, 175, 178, 179, 181, 182, 186–188, 191–193, 195, 204, 209, 210, 213, 214, 218, 254–262, 269, 270, 317–319, 321–323, 335, 341, 342, 345, 346, 349, 359–361, 377, 378
\csshow 258
\csuse 37, 40, 60, 70, 76, 77, 108, 171, 172, 195, 198, 199, 201, 203–205, 214, 225, 262, 328–340
\csxdef 86, 95
\currentglossary 11, 12, 40, 192
\currentglssubentry 12, 207
\CurrentOption 32, 261, 341
\CurrentTrackedLanguage 36, 379, 380
\CurrentTrackedTag 36, 379, 380
\CustomAcronymFields 253
\CustomNewAcronymDef 253

D

\d 23
datatool package 196
\day 161, 165, 323, 326
\DeclareAcronymList 16, 17, 19, 222, 224, 238, 240, 242, 244, 246, 249, 251, 253
\DeclareListParser 174
\DeclareOption 8, 261, 341
\DeclareOptionX 8
\DeclareRobustCommand 37, 188, 189, 247, 347–349
\def 8, 9, 11–15, 18, 22, 23, 28, 29, 32–34, 37, 38, 41, 44, 47–52, 56, 60–62, 64–68, 71, 79, 81–86, 88, 90, 94–98, 108, 109, 112–122, 124–149, 157, 158, 161, 165, 167–169, 171–173, 175, 176, 179, 182, 183, 185–188, 191, 194, 198, 200–205, 212–217, 219–222, 237–244, 246, 248–251, 253, 261, 263–267, 270–272, 297, 298, 317–320, 322, 323, 326, 328, 335, 336, 341–344, 358–361, 372–376
\def@gls@xdycheckbackslash 121, 122
\DefaultNewAcronymDef 238
\defglsentryfmt 61, 63, 108, 224, 237, 238, 241, 242, 245, 247, 250, 252
\define@boolkey 7, 9, 11, 12, 16, 24, 27–30, 109, 205
\define@choicekey 5–8, 12, 25, 26, 28, 66, 204, 205
\define@key 8, 12, 19, 25, 29, 64–68, 75, 76, 109, 158, 203–205, 261, 342, 343
\DefineAcronymSynonyms 32, 236
\delimN 164, 174, 202, 216, 325
\delimR 164, 215, 216, 325
\DescriptionDUANewAcronymDef 242
\DescriptionFootnoteNewAcronymDef 240
\descriptionname 37, 278–281, 283, 284, 288–290, 292, 300–305, 307–311
\DescriptionNewAcronymDef 244
\DH 23
\dh 23
\dimen@ 226, 286, 317
\disable@keys 32
\do 27, 32, 34, 45, 46, 53, 73, 74, 115, 157, 161–163, 173,

174, 182, 189, 194, 195, 224, 238, 240, 242, 244, 246, 249, 251, 253, 269, 270, 323
\do@glo@storeentry 13–15, 87
\do@gls@link@checkfirsthyper 111, 112, 124–129, 142–148, 359, 360
\do@gls@xdycheckbackslash 115
\do@glsdisablehyperinlist 112
\do@glshaschildren 56
doc package 4, 5, 16
\doifglossarynoexistsordo 61
\dtl@ifsingle 212
\dtl@insertinto 196
\dtl@sortresult 196, 197
\dtlcompare 197
\dtlicompare 197
\DTLifinlist 63, 112
\DTLifint 212
\dtlletterindexcompare 196
\DTLsubstituteall 115
\dtlwordindexcompare 196
\DUANewAcronymDef 251

E

\eappto 63, 88, 182
\edef 15, 18, 34, 36, 44–51, 53, 56, 61, 63, 70, 72, 74, 78–80, 82, 83, 88, 89, 108, 112–121, 157–160, 165, 167–169, 171, 172, 174, 175, 179, 183, 186, 187, 189, 193–197, 201, 207, 212, 216, 218, 237, 239, 241, 243, 245, 248, 250, 268, 321, 322, 324, 326, 371–375
\egroup 23, 81, 157, 192, 195
\else 6, 10–12, 15–18, 20, 22, 24, 25, 30, 32, 33, 38, 39, 41–48, 50–53, 66, 69, 84, 85, 88–90, 94, 95, 111–113, 115–122, 124–129, 150, 160, 161, 164–170, 172, 179–181, 183, 185–189, 192, 201, 205, 207, 208, 213, 215, 216, 226, 240, 242, 244, 247, 249–252, 254, 269, 274, 277, 279, 280, 283, 284, 286, 288, 290, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316, 318, 319, 322–328, 333–336, 347
\emph 188, 217
\empty 216, 341
\end 164, 200, 273, 277–282, 285–311, 325
\end@doifinlist 44, 45
\end@getprefix 187
\end@gls@islistofacronyms 18
\EndAccSupp 347
\endcsname 12–15, 32, 34, 36, 37, 42, 43, 46, 48, 49, 52, 53, 56, 61, 62, 69, 70, 75–80, 83–89, 91, 92, 108, 112–114, 124–129, 142–149, 157, 158, 162–164, 168–171, 175, 178, 179, 181, 182, 186–188, 191–193, 195, 204, 209, 210, 213, 214, 218, 254–262, 269, 270, 317–319, 321–323, 335, 341, 342, 345, 346, 349, 359–361, 377, 378
\endfoot . 277–279, 281, 283, 284, 288–290, 292
\endgroup 5, 181, 185, 216
\endhead . 277–279, 281, 283, 284, 288–290, 292
\endtheglossary 5
\entryname 37, 278–281, 283, 284, 288–290, 292, 300–305, 307–311
\equal 25, 33, 43, 113, 173, 213, 269
equation (counter) 113, 114
etoolbox package 4
\expandafter . 13–15, 22, 32–34, 36, 46, 48, 49, 51–54, 56, 61–63, 69, 70, 73–80, 83–87, 89–92, 108, 112–121, 150, 157, 159, 160, 163, 164, 167–171, 179, 181–183, 185, 186, 189, 192, 195, 196, 200, 201, 209, 210, 214, 216, 218, 239, 245, 254–259, 261, 262, 269, 270, 317, 321–323, 325, 326, 341, 342, 345, 347, 371, 377, 378
\expandonce 70, 71, 115, 168, 169, 182, 196, 197, 209, 210, 223, 237, 239, 241, 243, 245, 246, 248, 250, 341, 342

F

\fi 5–8, 10–18, 20, 22, 24, 25, 27, 30, 32–34, 37–39, 41–53, 62, 67, 69, 83–86, 88–90, 94, 95, 111–122, 124–129, 151, 158, 160–162, 164–172, 174, 179–181, 183, 185–192, 194, 201, 204–208, 213–216, 226, 236, 238, 240–242, 244, 246, 247, 249–254, 260, 269, 274, 277, 279, 280, 283, 284, 286–288, 290, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316–319, 321–325, 327, 328, 333–336, 341, 347
file types
  .aux 193
  .glo 88
  .ist 159, 170
  .toc 43
  .xdy 38
  glo 259
\firstacronymfont 107, 225–227, 232, 233, 239, 243, 245, 248, 358, 362–364, 368, 369

\footnote 232, 233, 238, 369
\FootnoteNewAcronymDef 246
\forallglossaries . . . 54, 179, 190, 191, 317
\forallglsentries 92, 93, 95, 159
\ForEachTrackedDialect 36, 379, 380
\forglsentries 54, 56, 88, 198, 317
\forlistcsloop 194, 200
\forlistloop 177, 202

G

garamondx package 218
\gdef 14, 46, 61, 79, 84, 85, 180, 206, 269
\Genacrfullformat 106, 223, 225–227, 232, 358, 362, 363, 369
\genacrfullformat 106, 107, 223, 225, 226, 232, 357, 358, 362, 363, 368
\GenericAcronymFields . . 223, 225, 226, 228–232, 234, 235, 362–365, 367, 368, 371
\Genplacrfullformat 106, 223, 225–227, 233, 357, 363, 369
\genplacrfullformat 106, 107, 223, 225–227, 233, 357, 363, 369
\glo@desc 328
\glo@do@compare 196, 197
\glo@grabfirst 201
\glo@label 56, 88
\glo@list 88
\glo@name 209
\glo@parent 56
\glo@type 88
\glo@value 73
\global 14, 15, 69, 72, 81, 86, 91, 92, 181, 192, 193, 201, 202, 206, 286, 287
\glolinkprefix 113, 157, 208
\glosortentrieswarning 20, 194
glossaries package 31, 51, 52, 161, 253, 261, 273, 321, 341
glossaries-accsupp package 88, 341
glossaries-extra package 72, 163, 341
\GlossariesWarning 5, 7, 20, 24, 25, 40, 44, 55, 59, 66, 69, 96, 97, 108, 110, 156, 172, 175–178, 181, 187, 191, 192, 210, 211, 213, 321, 341
\GlossariesWarningNoLine 5, 6, 20, 173, 175, 179, 194, 269
\glossary 322, 323
glossary package 1, 217
glossary styles:
 altlist 274, 275, 329
 altlistgroup 275, 329
 altlisthypergroup 275, 329
 altlong4col 281, 282, 290, 331
 altlong4col-booktabs 284, 286
 altlong4colborder 282, 331
 altlong4colheader 282, 284, 331
 altlong4colheaderborder 282, 331
 altlongragged4col . . . 285, 290–292, 332
 altlongragged4col-booktabs 285
 altlongragged4colborder 292, 332
 altlongragged4colheader 291, 332
 altlongragged4colheaderborder 292, 333
 altsuper4col 304, 309, 340
 altsuper4colborder 304, 340
 altsuper4colheader 304, 340
 altsuper4colheaderborder . . . 304, 340
 altsuperragged4col 309, 310, 338
 altsuperragged4colborder . . . 310, 338
 altsuperragged4colheader . . . 310, 338
 altsuperragged4colheaderborder 310, 338
 alttree 297, 312, 317, 319, 335
 alttreegroup 320, 336
 alttreehypergroup 320, 336
 index 8, 293, 311–314, 333
 indexgroup 313, 333
 indexhypergroup 313, 333
 inline 328
 list 8, 10, 273–275, 328
 listdotted 275, 276, 329
 listgroup 274, 328
 listhypergroup 274, 329
 long 277, 278, 283, 287, 329, 331
 long-booktabs 283, 285
 long3col 278, 279, 283, 330
 long3col-booktabs 283, 285
 long3colborder 279, 330
 long3colheader 279, 283, 330
 long3colheaderborder 279, 330
 long4col 280, 281, 284, 330
 long4col-booktabs 284
 long4colborder 281, 331
 long4colheader 280, 284, 331
 long4colheaderborder 281, 331
 longborder 277, 330
 longheader 277, 283, 330
 longheaderborder 278, 330
 longragged 285, 287–289
 longragged-booktabs 285

longragged3col 285, 289, 290, 332
longragged3col-booktabs 285
longragged3colborder 290, 332
longragged3colheader 290, 332
longragged3colheaderborder . 290, 332
longraggedborder 288, 331
longraggedheader 288, 332
longraggedheaderborder 289, 332
mcolalttree 297, 337
mcolalttreegroup 297, 337
mcolalttreehypergroup . . . 297, 298, 337
mcolindex 293, 336
mcolindexgroup 293, 336
mcolindexhypergroup 293, 294, 336
mcoltree 294, 336
mcoltreegroup 336
mcoltreehypergroup 295, 336
mcoltreenoname 296, 337
mcoltreenonamegroup 296, 337
mcoltreenonamehypergroup . . . 296, 337
sublistdotted 329
super 299, 300, 307, 339
super3col 300–302, 339
super3colborder 301, 339
super3colheader 301, 339
super3colheaderborder 302, 339
super4col 302–304, 340
super4colborder 303, 340
super4colheader 303, 340
super4colheaderborder 303, 340
superborder 299, 339
superheader 300, 339
superheaderborder 300, 339
superragged 305, 307, 337
superragged3col 307–309, 338
superragged3colborder 308, 338
superragged3colheader 308, 338
superragged3colheaderborder 309, 338
superraggedborder 306, 337
superraggedheader 307, 337
superraggedheaderborder 307, 338
tree 294, 314, 315, 317, 333
treegroup 295, 315, 334
treehypergroup 315, 334
treenoname 295, 312, 315, 316, 334
treenonamegroup 316, 335
treenonamehypergroup 316, 335
glossary-hypernav package 159
glossary-list package 8, 10, 273
glossary-long package 9, 276, 290, 298, 299
glossary-longragged package 287
glossary-mcols package 292
glossary-super package . . 10, 276, 298, 305, 309
glossary-superragged package 305
glossary-tree package 10, 311
\glossaryentry 187, 188, 323
glossaryentry (counter) 11, 206, 207
\glossaryentryfield . . . .
. . . . 209, 328–335, 337–340, 362
\glossaryentrynumbers . . . .
. . 9, 164, 191, 192, 201, 202, 205, 206, 325
\glossaryheader . . . .
. . . . 164, 200, 271, 273–275, 277–281, 283, 284, 287–293, 295–297, 299, 300, 302, 306, 307, 309, 312–317, 320, 325
\glossarymark 41
\glossaryname 16, 36, 37
\glossarypostamble 164, 200, 325
\glossarypreamble 163, 200, 325
\glossarysection 163, 200, 325
glossarysubentry (counter) 12, 206–208
\glossarysubentryfield . . . .
. . . . 210, 328–335, 337–340, 362
\glossarytitle . . 163, 191, 192, 200, 204, 325
\glossarytoctitle 8, 16, 17, 31, 34, 37, 41, 163, 191, 200, 204, 325
\glossentry 88, 192, 202, 210, 271, 273–278, 280, 288, 289, 291, 299, 301, 302, 306, 307, 309, 312, 314, 315, 318
\Glossentrydesc 361
\glossentrydesc . . . .
. 271–278, 280, 288, 289, 291, 299, 301, 302, 306–309, 312–314, 316, 318, 319, 361
\glossentryname . . . .
. 271–278, 280, 288, 289, 291, 299, 301, 302, 306, 307, 309, 312–315, 318, 319, 361
\Glossentrysymbol 362
\glossentrysymbol 271, 272, 280, 291, 302, 309, 312–314, 316, 318, 319, 362
\Gls 96, 218, 236
\gls 96, 175, 207, 218, 236
\gls@Alphpage 181, 185
\gls@alphpage 181, 185
\gls@arabicpage 181, 185
\gls@assign@desc 81, 86
\gls@assign@descplural . . . .
. . . . 237, 246, 248–251, 372, 375, 376

\gls@assign@field 71, 75, 76, 81, 83, 85, 86, 261, 262
\gls@assign@firstpl 237, 239–241, 243, 244, 246, 248–251, 372–376
\gls@assign@plural 237, 239–241, 243, 244, 246, 248–251, 372–376
\gls@assign@symbolplural 237, 239–241, 243, 244, 249–251, 373, 374, 376
\gls@begindocdefs 72
\gls@checkisacronymlist 111
\gls@checkseeallowed 66, 72, 173, 175
\gls@checkseeallowed@preambleonly . . 72
\gls@codepage 52, 171, 194
\gls@defdocnewglossaryentry 73, 93
\gls@defglossaryentry 71–73, 81
\gls@disablepagerefexpansion . . 181, 185
\gls@do@addxdyattribute 46
\gls@doclearpage 43
\gls@dosubst 115
\gls@dotoctitle 191, 192, 204
\gls@end@sanitizesort 22, 23
\gls@endcheck 70, 71
\gls@glossary 180, 186–188
\gls@gobbleopt 62
\gls@grplabel 268
\gls@hypergroupprerun 269
\gls@ifnotmeasuring 90, 91
\gls@inlinepostchild 271, 272, 328
\gls@inlinesep 271, 328
\gls@inlinesubsep 271, 272, 328
\gls@islistofacronyms 18
\gls@istfilebase 38, 171
\gls@label 217
\gls@level 84, 85, 201
\gls@noidxglossary 175
\gls@nosetquote 82, 165, 167, 170
\gls@number 185
\gls@numberpage 181, 185
\gls@org@glossaryentryfield 192
\gls@org@glossarysubentryfield 192
\gls@org@insert 242, 245, 247
\gls@orgAlph 185
\gls@orgalph 185
\gls@orgarabic 185
\gls@orgnumber 185
\gls@orgRoman 185
\gls@orgromannumeral 185
\gls@orgthe 185
\gls@protected@pagefmts 115, 182
\gls@Romanpage 181, 185
\gls@romanpage 181, 185
\gls@save@numberlist 9
\gls@set@xr@key 66
\gls@suffixF 39, 164, 166, 325, 327
\gls@suffixFF 39, 164–167, 325, 327
\gls@text 107
\gls@the 185
\gls@thissty 27
\gls@tmp 179, 247
\gls@tmplen 122, 317–319, 335, 336
\gls@tr@set@acronym@toctitle 17
\gls@tr@set@main@toctitle 16
\gls@tr@set@numbers@toctitle 31
\gls@tr@set@symbols@toctitle 31
\gls@wrglossary 180
\gls@xdystring 115
\gls@xindy@glsnumbersfalse 29
\gls@xindy@glsnumberstrue 29
\gls@xr@key 6, 7, 66
\glsaccsupp 347
\glsacronymtrue 17
\glsacrpluralsuffix 35, 218, 227, 228, 232–234, 238
\glsacrshortcutsfalse 32
\glsacrshortcutstrue 32
\glsacspace 226, 228
\glsadd 159
\glsadd options
  counter 158
  format 158, 215
\glsaddall options
  types 158
\GlsAddXdyAttribute 45, 47, 321, 322
\GlsAddXdyCounters 45, 46, 62
\glsautomakefalse 30
\glsautoprefix 8, 204
\glscapscase 99, 100, 103–106, 124–128, 142–148, 230, 231, 243, 248, 349, 351, 353, 354, 356, 357, 359–361, 366
\glsclearpage 42
\glsclosebrace 49, 50, 164, 165, 325, 326
\glscompositor 38, 39, 49, 166, 324, 327
\glscounter 19, 33, 45, 62, 85, 113, 321
\glscurrententrylabel 190, 192
\glscurrentfieldvalue 58, 59
\glscustomtext 99, 102, 104, 105, 107, 124–128, 142–149,

230, 231, 238, 239, 242, 243, 245, 247, 248, 349, 353, 355, 356, 358–361, 366, 367
\GlsDeclareNoHyperList 19
\glsdefaulttype 16, 40, 52, 53, 60, 61, 83, 98, 108, 179, 190, 191
\glsdefmain 16, 63
\glsdescriptionaccessdisplay 351–353, 361, 362
\glsdescriptionpluralaccessdisplay 349–351
\glsdescwidth 277–279, 281, 282, 285–292, 299–302, 304–311
\glsdetoklabel 54–59, 67, 73, 78–82, 88, 91–95, 112, 149, 150, 157, 158, 175–177, 183, 186, 192, 195–197, 201, 203, 207, 209, 254–258, 317, 341, 342, 377, 378
\glsdisplay 99, 108
\glsdisplayfirst 99, 108
\glsdisplaynumberlist 176
\glsdohyperlink 122, 123
\glsdohypertarget 123
\glsdoifexists 56, 58, 78–80, 90, 91, 124–129, 142–148, 156, 158, 176, 177, 263–267, 358–362
\glsdoifexistsordo 111, 149
\glsdoifexistsorwarn 203, 209, 210
\glsdoifnoexists 71, 81
\glsdonohyperlink 113, 122, 123
\glsdosanitizesort 13
\glsentryaccess 347
\glsentrycounter 213, 216
\glsentrycounterfalse 11
\glsentrycounterlabel 208
\GlsEntryCounterLabelPrefix 207
\glsentrycountertrue 12
\glsentrycurrcount 93, 95
\Glsentrydesc 134, 209, 361
\glsentrydesc 101, 102, 134, 209, 351–353, 361
\glsentrydescaccess 348
\Glsentrydescplural 135
\glsentrydescplural 99, 100, 134, 135, 349–351
\glsentrydescpluralaccess 348
\Glsentryfirst . . . 97, 101, 104, 131, 352, 355
\glsentryfirst 96, 101, 102, 104, 130, 131, 351, 352, 355
\glsentryfirstaccess 348
\Glsentryfirstplural 98, 100, 103, 132, 350, 354
\glsentryfirstplural 97, 99, 100, 103, 132, 349, 351, 353, 354
\glsentryfirstpluralaccess 348
\glsentryfmt 61, 63
\Glsentryfull 223, 232, 233, 368, 370
\glsentryfull 223, 232, 233, 367, 370
\Glsentryfullpl 223, 232, 233, 368, 370
\glsentryfullpl 223, 232, 233, 368, 370
\glsentryitem 271, 273–278, 280, 288, 289, 291, 299, 301, 302, 306, 307, 309, 312, 314, 315, 318, 328–335, 337–340
\Glsentrylong 97, 146, 150, 156, 225, 226, 231, 232, 358, 360, 363, 366–368
\glsentrylong 96, 107, 145, 147, 150, 156, 225–235, 245, 358, 360–371
\glsentrylongaccess 348
\Glsentrylongpl 98, 148, 156, 225, 226, 230–232, 358, 363, 366–368
\glsentrylongpl 97, 107, 147, 149, 156, 225–227, 230–233, 245, 252, 358, 363, 364, 366–370
\glsentrylongpluralaccess 349
\Glsentryname 133, 209, 361
\glsentryname 133, 317, 361
\glsentrynumberlist 156, 176
\Glsentryplural 100, 103, 131, 350, 354
\glsentryplural 99, 100, 103, 131, 132, 349, 350, 353, 354
\glsentrypluralaccess 348
\Glsentryprefix 265
\glsentryprefix 263, 266
\Glsentryprefixfirst 265
\glsentryprefixfirst 264, 266
\Glsentryprefixfirstplural 266
\glsentryprefixfirstplural 264, 267
\Glsentryprefixplural 265
\glsentryprefixplural 264, 267
\glsentryprevcount 93–95
\Glsentryshort 105, 142, 150, 227, 233, 357–359, 363, 369, 370
\glsentryshort 105, 107, 142, 143, 150, 156, 223, 225–235, 356–359, 362–365, 367–371
\glsentryshortaccess 348
\Glsentryshortpl 105, 144, 227, 233, 356, 363, 369, 370
\glsentryshortpl 105, 107, 144, 145, 156, 225–227, 231–233, 252, 356, 358, 363, 367–370
\glsentryshortpluralaccess 348

\Glsentrysymbol 135, 210, 362
\glsentrysymbol 101, 102, 135, 136, 210, 239, 243, 248, 351–353, 362
\glsentrysymbolaccess 348
\Glsentrysymbolplural 136
\glsentrysymbolplural 99, 100, 136, 137, 239, 243, 248, 349–351
\glsentrysymbolpluralaccess 348
\Glsentrytext 101, 104, 130, 351, 355
\glsentrytext 101, 102, 104, 129, 130, 157, 189, 351, 352, 354, 355
\glsentrytextaccess 347
\glsentrytype 83
\Glsentryuseri 137
\glsentryuseri 137
\Glsentryuserii 138
\glsentryuserii 138
\Glsentryuseriii 139
\glsentryuseriii 138, 139
\Glsentryuseriv 139
\glsentryuseriv 139, 140
\Glsentryuserv 140
\glsentryuserv 140, 141
\Glsentryuservi 141
\glsentryuservi 141
\glsesclocationsfalse 175
\glsesclocationstrue 9
\glsfieldfetch 153
\glsfirstaccessdisplay 351, 352, 355
\glsfirstpluralaccessdisplay 349, 350, 353, 354
\glsfirstpluralacessdisplay 354
\glsgenacfmt 225, 226, 232, 362, 363, 368
\glsgenentryfmt 225, 226, 231, 232, 237, 239, 241–243, 245, 247, 250, 252, 362, 363, 367, 368
\glsgetgrouptitle 270, 274, 275, 293–298, 313–316, 320
\glsglossarymark 41
\glsgroupheading 165, 202, 271, 273–275, 277, 278, 280, 288, 289, 291, 293–300, 302, 306, 307, 309, 312–317, 319, 320, 326
\glsgroupskip 164, 165, 202, 272, 274, 277, 279, 280, 283, 284, 288–291, 299, 301, 303, 306, 308, 310, 313, 314, 316, 319, 325
\glshyperfirstfalse 232, 368
\glshyperfirsttrue 27
\glshyperlink 189
\glshypernavsep 270
\glshypernumber 40, 216, 217
\glsifhyperon 110
\glsIfListOfAcronyms 18
\glsifplural 99, 102, 105, 106, 124–128, 142–148, 230, 239, 243, 245, 247, 349, 353, 356, 357, 359–361, 366
\glsifusetranslator 26, 27, 36, 37, 379
\glsindexonlyfirstfalse 27
\glsinlinedescformat 271, 328
\glsinlinedopostchild 271, 328
\glsinlineemptydescformat 271, 328
\glsinlinenameformat 271, 328
\glsinlineparentchildseparator 271, 328
\glsinlinepostchild 271, 328
\glsinlineseparator 271, 328
\glsinlinesubdescformat 272, 328
\glsinlinesubnameformat 271, 328
\glsinlinesubseparator 272, 328
\glsinsert 99–106, 124–128, 142–148, 230, 231, 239, 242, 243, 245, 247, 248, 349–361, 366, 367
\glskeylisttok 222, 223, 237–244, 246, 248–251, 253, 371–376
\glslabel . . 82, 99–106, 111–113, 142–148, 225, 226, 230–232, 238, 239, 242, 243, 245, 247, 248, 349–358, 362, 363, 366–368
\glslabeltok 222, 223, 237–246, 248–253, 372–375
\glslink 223, 231, 233, 238, 367, 369
\glslink options
  counter 109, 123, 260
  format 109, 123, 215
  hyper 109, 111, 112, 123
  local 109
\glslinkcheckfirsthyperhook 111
\glslinkpostsetkeys 112
\glslinkvar 110
\glslistdottedwidth 275, 276, 329
\glslistgroupheaderfmt 274, 275
\glslistnavigationitem 274, 275
\glslocalreset 92
\glslocalunset 93, 124–129
\glslongaccessdisplay 358, 360–372
\glslongkey 376
\glslongpluralaccessdisplay 358, 363, 364, 366–370, 372
\glslongpluralkey 376

\glslongtok 222, 223, 225, 226, 231, 232, 237–246, 248–253, 362, 363, 367, 368, 371–376
\glsLTpenaltycheck 286, 287
\glsmcols 293–298
\glsnameaccessdisplay 361, 362
\glsnamefont 209, 210, 341, 342, 361
\glsnavhyperlink 270
\glsnavhyperlinkname 268, 269
\glsnavhypertarget 274, 275, 294–298, 314–316, 320
\glsnavigation 274, 275, 293–298, 313, 315, 316, 320
\glsnextpages 9, 67, 192
\glsnogroupskipfalse 11
\glsnoidxdisplayloc 177, 178
\glsnoidxdisplayloclisthandler 177
\glsnoidxloclist 176, 201, 202
\glsnoidxloclisthandler 202
\glsnoidxnumberlistloophandler 177
\glsnoidxstripaccents 23
\glsnomakeindexwarning 167
\glsnonextpages 66, 192
\glsnopostdotfalse 11
\glsnoxindywarning 39, 45–47, 50–52, 160, 161
\glsnumberformat 157
\glsnumberlistloop 177
\glsnumbersgroupname 31, 37, 213
\glsnumlistlastsep 157, 176
\glsnumlistparser 157, 174
\glsnumlistsep 157, 176
\glsopenbrace 49, 50, 164, 165, 325, 326
\glsorder 28, 171–173, 198
\glsorg@endtheglossary 5
\glsorg@PrintChanges 5
\glsorg@theglossary 5
\glspagelistwidth 278, 279, 281, 282, 285, 286, 289–292, 300–302, 304, 305, 307–311
\glspatchLToutput 283–285
\glspenaltygroupskip 283, 284
\glspercentchar 73, 164, 165
\Glspl 97, 236
\glspl 97, 236
\glspluralaccessdisplay 349, 350, 353, 354
\glspluralsuffix 35, 85, 225–227, 363, 364, 368–370
\glspostdescription . 37, 272–275, 277, 288, 299, 306, 312–314, 316, 318, 319, 328–331, 333–337, 339
\glspostinline 271
\glspostlinkhook 111, 124–129, 142–149, 359–361
\glsprestandardsort 13
\glsreset 92
\glsresetentrycounter 206
\glsresetentrylist 164, 200, 206, 325
\glsresetsubentrycounter . . . 208, 271, 328
\glssanitizesortfalse 24, 25
\glssanitizesorttrue 24, 25
\glssavenumberlistfalse 9
\glssavewritesfalse 30
\glsseeformat 163, 175, 177, 325
\glsseeitem 189
\glsseeitemformat 189
\glsseelastsep 189
\glsseelist 188
\glsseesep 189
\glssetexpandfield 21, 24, 25
\glssetnoexpandfield 21, 22, 24, 25
\GlsSetQuote 82, 165
\glssettoctitle 37, 191
\glsshortaccessdisplay 356–359, 362–365, 367–372
\glsshortkey 376
\glsshortpluralaccessdisplay 356, 358, 363, 367–370, 372
\glsshortpluralkey 376
\glsshorttok 222, 223, 237–246, 248–253, 372–376
\glsshowtarget 6
\glssortnumberfmt 14, 15
\glsspace 219
\glsstepentry 208
\glsstepsubentry 208
\glssubentrycounterfalse 12
\glssubentrycounterlabel 208
\glssubentryitem 272, 273, 275–278, 280, 288, 289, 291, 299, 301, 302, 306, 308, 309, 313, 314, 316, 318, 328–335, 337–340
\glssymbolaccessdisplay 351–353, 362
\glssymbolpluralaccessdisplay . 349–351
\glssymbolsgroupname 31, 37, 213
\glstarget 210, 211, 272–278, 280, 288, 289, 291, 299, 301, 302, 306–309, 312–316, 318, 319, 328–340
\glstextaccessdisplay . . 351, 352, 354, 355
\glstextformat 111, 113
\glstextup 35, 370

\glstildechar 46, 164, 165
\glstranslatefalse 26, 27
\glstranslatetrue 26, 27
\glstreechildpredesc 313, 314
\glstreegroupheaderfmt 293–298, 313, 315, 316, 320
\glstreeindent . . 314, 316, 318, 319, 334–336
\glstreeitem 293, 294, 312
\glstreenamebox 318, 319
\glstreenamefmt 311–315, 317–319
\glstreenavigationfmt 293–298, 313, 315, 316, 320
\glstreepredesc 312, 314, 316
\glstreesubitem 293, 312
\glstreesubsubitem 293, 312
\glstype . 111, 112, 124–129, 142–149, 359–361
\glsucmarkfalse 11
\glsucmarktrue 11
\glsunset 90, 92, 94, 95, 124–129
\glsupacrpluralsuffix 227, 228, 234, 240, 244, 247, 249
\GlsUseAcrEntryDispStyle . . . 224, 227–230, 232, 234, 235, 364, 365, 368, 370, 371
\GlsUseAcrStyleDefs 224, 227–230, 232, 234, 235, 364, 365, 368, 370, 371
\glswrallowprimitivemodstrue 183
\glswrite . . . 161–167, 172, 178, 179, 323–327
\glswritedefhook 73
\glswriteentry 181
\glswritefiles 30, 179
\glsxindyfalse 29
\glsxindytrue 29

H

\H 23
\hangindent 297, 298, 311, 314–316, 318–320, 333–336
\hbox 90, 275, 276, 329
\hfill 275, 276, 329
\hline . . . 277–279, 281, 288–290, 292, 299–311
\hsize 276, 287, 298, 305
\hspace 312
\hss 275, 276, 329
\ht 286
\hyperdef 33
\hyperlink 109, 122, 216
hyperref package 187, 190, 215, 260
\hypertarget 122

I

\IeC 23
\if 115, 186, 322
\if@endfor 269
\if@gls@debug 5, 20, 180
\if@gls@docloaded 4, 16, 180
\if@glsisacronymlist 111
\if@openright 43
\ifbool 17, 28, 30, 55, 99–101
\ifboolexpr 36, 60, 212
\ifcase 5, 6, 8, 26, 66, 204, 313, 333
\ifcsdef 26, 37, 43, 69, 70, 76–80, 108, 181, 194, 195, 198, 199, 214, 224
\ifcsempty 56, 57, 263
\ifcsequal 57
\ifcsstrequal 80
\ifcsstring 80
\ifcsundef 7, 14, 29, 33, 36, 40, 41, 43, 54, 61, 63, 65, 83, 85, 86, 93, 94, 109, 113, 114, 122, 171, 179, 190, 193, 195, 203, 204, 208, 212–215, 217, 224, 270, 327
\ifdef 58, 67, 72, 73, 90, 109, 149, 153, 176, 177, 218, 311, 312
\ifdefempty 19, 42, 43, 53, 57–59, 63, 99, 102, 105, 174, 200, 201, 222, 224, 230, 238, 242, 245, 247, 349, 353, 356, 366
\ifdefequal 56–59, 69–71, 74, 84, 88, 201
\ifdefstrequal 80
\ifdefstring 10, 36, 60, 171–173, 197, 198, 201–203
\ifdefvoid 22, 87, 201, 202
\ifdim 226, 286, 317
\iffalse 86, 92
\IfFileExists 10, 26, 193
\ifglossaryexists . . 40, 52, 56, 170–172, 191
\ifgls@sanitize@description 24
\ifgls@sanitize@name 24
\ifgls@sanitize@symbol 24
\ifgls@xindy@glsnumbers 52
\ifglsacrdescription 251
\ifglsacrdua 241, 247, 250–252
\ifglsacrfootnote 111, 251, 252
\ifglsacronym 16, 17
\ifglsacrshortcuts 32, 236
\ifglsacrsmallcaps 240, 242, 244, 246, 249, 250
\ifglsacrsmaller 240, 242, 244, 247
\ifglsautomake 30, 174
\ifglsdescsuppressed 271

\ifglsentrycounter 11, 12, 32, 206–208
\ifglsentryexists 55, 72, 73, 81, 84
\ifglsesclocations 183
\ifglshaschildren 271, 328
\ifglshasdesc 271
\ifglshaslong 96–98, 150, 225, 226, 230, 232, 245, 362, 363, 366, 368
\ifglshasparent 195, 201, 317
\ifglshasprefix 265
\ifglshasprefixfirst 265
\ifglshasprefixfirstplural 265
\ifglshasprefixplural 265
\ifglshassymbol 239, 243, 247, 312–314, 316, 318, 319
\ifglshyperfirst 111
\ifglsindexonlyfirst 181
\ifglsnogroupskip 274, 277, 279, 280, 283, 284, 288, 289, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316, 319
\ifglsnonumberlist 205
\ifglsnopostdot 10
\ifglsnumberline 44
\ifglssanitizesort 22, 24, 25
\ifglssavenumberlist 69, 173, 189
\ifglssavewrites 30, 170, 181
\ifglssubentrycounter 12, 33, 206–208
\ifglstoc 44
\ifglstranslate 35
\ifglsucmark 41, 42
\ifglsused 95, 99–105, 111, 159, 181, 238, 242, 245, 247, 263–267, 349–356
\ifglswrallowprimitivemods 185
\ifglsxindy 38, 39, 44–47, 49–52, 62, 88, 89, 116, 160, 161, 167, 171, 186, 188, 193, 321–323
\ifignoredglossary 83, 87, 181
\ifin@ 32
\ifinlistcs 199, 203
\ifinner 6
\ifKV@glslink@hyper 112, 113
\ifKV@glslink@local 124–129
\ifmeasuring@ 90
\ifmmode 6
\ifnum 13, 14, 94, 95, 201, 286, 314, 316, 318, 334, 335
\ifstrempty 328
\ifstrequal 212
\ifthenelse 25, 33, 43, 113, 173, 213, 252, 269
\IfTrackedLanguage 167
\IfTrackedLanguageFileExists 36, 379, 380
\iftrue 87, 91
\ifundef 11, 12, 61, 72, 83, 161, 165, 172
\ifvmode 158
\ifvoid 286
\ifx 11, 13, 15, 17, 18, 32, 33, 44, 46, 48, 49, 52, 53, 83–86, 89, 113, 114, 116–121, 150, 161, 164, 166–169, 179, 183, 185–189, 191, 192, 213–216, 238, 240, 242, 244, 246, 247, 249, 251, 253, 254, 323–325, 327, 328, 333–336, 341, 347
\immediate 72, 73, 95, 171, 179, 193, 194
\in@ 32
\indexname 31
\indexspace . 274, 293–298, 313–316, 319, 320
\input 35, 98
\inputencodingname 29
\InputIfFileExists 72
\istfilename . 38, 161, 165, 172, 173, 323, 326
\item 273–276, 293, 294, 312, 313, 328, 329, 333

J

\jobname 38, 72, 161, 165, 171, 172, 193, 323, 326

K

\key@ifundefined 75, 76
\KV@glslink@hyperfalse . 109, 111, 112, 123
\KV@glslink@hypertrue 109, 123

L

\L 23
\l 23
\label 8, 204, 207
\languagename 29
\leaders 275, 276, 329
\leavevmode 81, 112
\let 5, 10, 13–17, 23, 26, 27, 30–33, 36, 37, 46, 47, 58–60, 69, 71, 72, 81–84, 86, 87, 90–93, 95, 98, 110–113, 115, 122–129, 142–148, 150, 157, 165, 167, 170–177, 180, 182, 185, 188, 189, 191–193, 202, 204, 206, 210, 222, 235–237, 239–251, 261, 269, 270, 286, 293, 294, 312, 327, 344, 359–361, 372–376, 379
\letcs . . 56–59, 73, 75, 76, 79, 85, 149, 150, 171, 176, 177, 195–197, 201, 209, 212, 317
link text 98
\listcsadd 199
\listcsgadd 203
\listcsxadd 194, 195

\listeadd 199
\loadglsentries 98
\long 81, 216
\longnewglossaryentry 81
longtable package 276, 283, 287
\LT@end@pen 286
\LT@err 286
\LT@foot 286, 287
\LT@head 286, 287
\LT@lastfoot 286
\LT@output 286

M

\makeatletter 72, 193
\makebox 275, 276, 317–319, 329, 335, 336
makeglossaries 28, 38, 51, 52, 61, 167, 173, 193
\makeglossaries 6, 7, 30, 34, 66, 174, 176, 178, 194
\makeglossary 171, 173
makeindex **381**
makeindex 9, 13, 29, 30, 35, 38–40, 44, 60–62, 64, 89, 114, 117, 159, 163, 165, 167, 170, 179, 184–187, 211, 322, 323
  delim_n 40
  delim_r 40
  page_compositor 38
  special characters 116, 159
\makenoidxglossaries 6, 7, 66, 174, 178
\MakeTextUppercase 4
\MakeUppercase 350, 352, 359, 361
\marginpar 6
\markboth 41
\mbox 158, 274, 297, 298, 317, 329
memoir class 180
\memUChead 41
\MessageBreak 20, 60, 191, 341, 379, 380
mfirstuc package 1
\mfirstucMakeUppercase 4, 41, 42, 77, 100, 102–106, 130–141, 143, 145, 147, 149, 223, 230–233, 243, 248, 266, 267, 354–358, 366, 367, 369, 370
\midrule 283, 284
\month 161, 165, 323, 326
multicol package 292

N

\n 166, 326
\NeedsTeXFormat . . . 4, 261, 321, 327, 341, 379
\new@glossaryentry 72, 176
\new@ifnextchar 60, 76, 77, 96, 97, 123–127, 129–148, 219–221, 263–266
\newacronym 217, 222, 238, 240, 242, 244, 246, 249, 251, 253
\newacronymhook 223, 238, 240, 242, 244, 246, 249, 251, 253, 371
\newacronymstyle . . . 225–230, 232, 234, 235
\newcommand 6–23, 25–28, 30–35, 37–47, 49–63, 65–81, 87, 88, 90–93, 95–99, 102, 105, 107, 108, 110–113, 115, 116, 122–161, 167, 170–172, 174, 178–183, 185–191, 193–199, 201–203, 205–226, 235, 237–259, 262–266, 268–270, 272, 273, 286, 293, 311, 312, 317, 327, 345–347, 362, 376–378
\newcount 14, 69
\newcounter 11, 12
\newenvironment 208
\newglossary 16, 17, 31, 62, 173
\newglossaryentry 6, 32, 68, 69, 72, 93, 223, 237, 239, 241, 243, 245, 248, 250, 252, 372–375
\newglossaryentry options
  access 344, 345
  counter 65
  description 28, 64, 69, 71, 82, 133, 151, 218, 246, 343
  descriptionaccess 346, 348
  descriptionplural 134, 343
  descriptionpluralaccess 346, 348
  entrycounter 205
  first 65, 85, 123, 130, 152, 244, 249, 342
  firstaccess 346, 348
  firstplural 65, 132, 152, 343
  firstpluralaccess 346, 348
  format 162
  long 105, 155, 343
  longaccess 347, 348
  longplural 155, 343
  longpluralaccess 347, 349
  name 64, 68, 71, 82, 133, 150, 189, 342
  nonumberlist 66, 67
  parent 66, 71
  plural 64, 85, 131, 343
  pluralaccess 346, 348
  prefix 261
  prefixfirst 261
  prefixfirstplural 262
  prefixplural 262

see 6, 9, 66, 71, 173, 175
short 105, 155, 343
shortaccess 346, 348
shortplural 155, 343
shortpluralaccess 346, 348
sort 64, 153, 180, 211
symbol 64, 65, 135, 240, 242, 244, 249, 280, 302, 342–344
symbolaccess 346, 348
symbolplural 136, 343
symbolpluralaccess 346, 348
text 64, 65, 123, 129, 151, 240, 244, 342
textaccess 346, 347
type 16, 65, 98, 153
user1 . . . . 137, 153, 344
user2 . . . . 137, 154
user3 . . . . 138, 154
user4 . . . . 139, 154
user5 . . . . 140, 154
user6 . . . . 141, 155, 344

\newglossarystyle 270, 273–285, 287–310, 312–317, 319, 320
\newif 4, 5, 18, 25, 29, 183
\newlength 122, 276, 287, 298, 305, 315
\newrobustcmd 71, 72, 77, 96, 97, 111, 123–148, 150–156, 158, 159, 218–221, 262–266, 317
\newterm 32
\newtoks 116, 170, 222
\newwrite 72, 161, 165, 170, 172
\nfss@text 6
ngerman package 167
\noalign 286
\nobreak 274, 287, 329
\noexpand 18, 34, 45, 46, 72, 86, 87, 108, 113–115, 121, 122, 157, 167–172, 174, 182, 183, 186, 190, 193, 194, 196, 197, 209, 210, 218, 223, 237, 239, 241, 243, 245, 246, 248, 250, 252, 253, 321, 341, 342, 372–376
\nohyperpage 215
\noindent 210, 294–296, 298, 315, 316
\noist 326, 327
\nopostdesc 32, 37, 81, 192, 328
\normalbaselineskip 286
\nr 5, 6, 8, 26, 66, 204
\ns@ACRfull 220
\ns@Acrfull 219, 220
\ns@acrfull 219
\ns@ACRfullpl 221
\ns@Acrfullpl 221
\ns@acrfullpl 220
\ns@ACRlong 146
\ns@Acrlong 146
\ns@acrlong 145
\ns@ACRlongpl 148
\ns@Acrlongpl 148
\ns@acrlongpl 147
\ns@ACRshort 143
\ns@Acrshort 142
\ns@acrshort 141, 142
\ns@ACRshortpl 144
\ns@Acrshortpl 144
\ns@acrshortpl 143
\ns@newglossary 61
\null 115–122, 167–170, 193
\number 14, 72, 85, 95, 182, 185, 210, 342
\numberline 44
\numexpr 95

O

\O 23
\o 23
\OE 23
\oe 23
\openout 72, 161, 165, 171, 323, 326
\OR 252
\or 5, 7, 8, 27, 204, 313, 333
\org@glossaryentrynumbers 192, 206
\org@glossarytitle 191, 192
\org@glspostdescription 37
\org@ifKV@glslink@hyper 112, 113
\outputpenalty 286

P

\p@ 273, 293, 311, 312
\p@gls@hyp@opt 110
package options:
acronym 16, 17, 34, 190, 218
true 17
counter 19
debug
showtargets 6
description 244, 245
dua 242, 244, 245
entrycounter 11, 205, 206
true 11
esclocations 404

false 9
footnote 124–129, 240, 242, 244, 246
hyperfirst
false 124–129
indexonlyfirst 388
makeindex 163, 260
nogroupskip 277, 279, 280, 283, 284, 288, 289, 291, 299, 301, 303, 306, 308, 310
nolist 253
nolong 254, 276
nomain 16
nonumberlist 9
nosuper 254
notree 254
nowarn 5
numberline 7
sanitize 24, 64, 150, 151
sanitizesort 21
savewrites 30, 385
false 170
true 172, 178, 179
section 7, 42
sort
def 12, 13
none 12
standard 12
use 12, 13, 404
style 8, 253, 254
subentrycounter 12, 205, 206
toc 7
true 7
translate 26
false 26
translator 25
xindy 29, 163, 260
\PackageError 6, 7, 15, 30, 34, 45, 52, 55, 56, 60, 65, 68, 69, 75–80, 83, 84, 93, 109, 149, 170, 171, 173, 176, 178, 197–199, 204, 205, 213, 214, 224, 225, 241, 242, 247, 250, 327, 349
\PackageInfo 5, 6, 171, 180
\PackageWarning 5, 19
\PackageWarningNoLine . . . 5, 6, 20, 379, 380
\pagegoal 286
\pagelistname 37, 279, 281, 283, 284, 290, 292, 301–305, 308–311
\par 37, 210, 211, 273–275, 293, 295–298, 311, 312, 314–320, 329, 334–336
\parindent 293–298, 312, 314–316, 318–320, 334–336
\parskip 293–296, 312, 314, 315
\PassOptionsToPackage 261, 341
\penalty 286
\phantomsection 43
polyglossia package 25, 36
\printglossaries 173
\printglossary 17, 20, 31, 173, 190, 205
\printglossary options
entrycounter 205
nogroupskip 204
nonumberlist 205
nopostdot 205
numberedsection 204
style 204
subentrycounter 205
title 204
toctitle 204
type 16, 189, 203
\printindex 31
\printnoidxglossaries 175
\printnoidxglossary 175, 178, 191, 197, 198, 205
\printnoidxglossary options
sort 205
\printnumbers 31
\printsymbols 31
\ProcessOptions 261, 341
\ProcessOptionsX 32
\protect 44, 107, 225–227, 232, 233, 239, 243, 245, 358, 362, 363, 368, 369
\protected@edef 8, 46, 48, 51, 53, 83, 87, 89, 99–101, 107, 114, 157, 181, 183, 186, 204, 209, 210, 213, 214, 223, 247, 252, 262, 268, 322, 323, 341, 342, 347
\protected@write 60, 61, 162, 163, 172, 173, 175, 178, 181, 190, 269, 323
\protected@xdef 13–15, 18, 23, 70, 89, 185, 345
\providecommand 17, 34, 35, 42, 60, 95, 123, 163, 172, 175, 178, 193, 194, 209, 210, 273, 293, 311
\ProvidesFile 35
\ProvidesPackage 4, 261, 268, 270, 273, 276, 282, 287, 292, 298, 305, 311, 321, 327, 341, 379

R

\r 23

\raggedright 285–292, 306–311
\raisebox 122
\ref 207
\refstepcounter 207
\relax 5, 8, 10, 14–17, 26, 27, 32, 33, 47, 60, 65, 66, 70, 72, 84, 86, 90, 94, 95, 110, 111, 113, 115–121, 150, 164, 165, 167, 169, 170, 173, 175, 177, 182, 186, 188, 189, 191, 193, 200, 201, 204, 212–214, 254, 269, 273, 286, 293, 297, 298, 311, 313–320, 322, 325–327, 333–336, 344, 359, 360, 372–374, 376
\renewacronymstyle . 362–366, 368, 370, 371
\renewcommand 4–10, 12, 15–17, 19–21, 24, 26, 28, 30, 32, 36–39, 52, 63, 66, 67, 81, 93–95, 157, 158, 160, 161, 167, 168, 173–177, 180, 192, 194, 204, 222, 223, 225–235, 238, 240, 242, 244, 246, 247, 249, 251, 253, 271–281, 283, 284, 286–303, 306–310, 312–322, 327–335, 337–340, 344, 349, 353, 356, 358, 361–365, 367–375
\renewenvironment 208, 270, 273, 277–282, 285–312, 314, 315, 317
\RequireGlossariesLang 36, 379, 380
\RequirePackage 4, 9, 10, 26, 32, 35, 253, 260, 261, 276, 282, 283, 287, 293, 298, 305, 342, 379
\restorecounters@ 114
\romannumeral 182, 185, 317–319, 335

S

\s@gls@hyp@opt 110
\s@GlsSetXdyFirstLetterAfterDigits 160
\s@GlsSetXdyNumberGroupOrder . . 160, 161
\s@newglossary 61
\savecounters@ 113
\seename 188
\SetAcronymStyle 28
\setbool 25
\setbox 286, 287
\setcounter 206
\SetCustomDisplayStyle 253
\SetDefaultAcronymDisplayStyle 238
\SetDefaultAcronymStyle 251
\SetDescriptionAcronymDisplayStyle 244
\SetDescriptionAcronymStyle 251
\SetDescriptionDUAAcronymDisplayStyle 242
\SetDescriptionDUAAcronymStyle 251
\SetDescriptionFootnoteAcronymDisplayStyle 240
\SetDescriptionFootnoteAcronymStyle 251
\SetDUADisplayStyle 251
\SetDUAStyle 252
\setentrycounter 46, 163, 203, 321
\SetFootnoteAcronymDisplayStyle . . . 246
\SetFootnoteAcronymStyle 252
\SetGenericNewAcronym 224
\setglossarystyle 191, 213, 254, 274–286, 288–310, 313–316, 319, 320
\setglossentrycompatibility 204, 213, 214
\setkeys 25, 29, 33, 42, 83, 112, 158, 159, 191, 222, 238, 240, 242, 244, 246, 249, 251, 253
\setlength 276, 287, 293–296, 298, 305, 312, 314, 315, 319, 335, 336
\SetSmallAcronymDisplayStyle 249
\SetSmallAcronymStyle 252
\settoheight 122
\settowidth 226, 317–319, 335
\sfcode 10
\show 254–259, 377, 378
\small 6
\SmallNewAcronymDef 249
\space 6, 7, 30, 34, 45, 49, 50, 52, 53, 66, 68, 69, 93, 96, 97, 107, 108, 110, 156, 162–165, 170–173, 175, 178, 189, 191, 194, 207, 213, 219, 225–235, 243, 247, 248, 270, 272–275, 277, 288, 299, 306, 312–314, 316, 318, 319, 321, 322, 324–326, 328–331, 333–337, 339, 358, 362–365, 367–372
\spacefactor 10
\SS 23
\ss 23
\string 6, 7, 15, 20, 30, 34, 44–46, 48–53, 60, 61, 66, 68, 69, 73, 76, 77, 88, 89, 93, 95–97, 108, 110, 114, 115, 117–119, 121, 156, 159, 160, 162–167, 169–173, 175, 176, 178, 186–188, 191, 193, 194, 197, 198, 205, 210, 211, 213, 269, 321–327
\strut 211, 274–278, 280, 288, 289, 291, 299, 301, 302, 306, 308, 309, 316, 328–332, 334, 337–340
\subglossentry 88, 192, 201, 210, 211, 271, 273, 275–278, 280, 288, 289, 291, 299, 301, 302, 306, 308, 309, 313, 314, 316, 318
\subitem 293, 312, 313, 333

\subsubitem 293, 312, 313, 333
supertabular package 10, 254, 298, 305
\symbolname 37, 280, 281, 284, 292, 303–305, 310, 311

T

\t 23
\tablehead 299–311
\tabletail 299–311
\tabularnewline 277–281, 283, 284, 288–292, 299–311, 331, 332, 337, 338
\texorpdfstring 153
\textbar 270
\textbf 210, 217, 311, 333–336
textcase package 4
\textit 217
\textmd 217
\textrm 216
\textsc 217, 227, 228, 234, 240, 244, 247, 249, 370
\textsf 216
\textsl 217
\textsmaller 227, 228, 234, 240, 244, 247, 249, 370
\texttt 6, 217
\textulc 218
\textup 217, 218
\TH 23
\th 23
\the 34, 36, 46, 51, 53, 61, 116–121, 161, 165, 167–169, 179, 181, 185, 190, 201, 209, 210, 216, 223, 225, 226, 231, 232, 237, 239, 241, 243, 245, 246, 248, 250, 252, 253, 321, 323, 326, 342, 362, 363, 367, 368, 371–376
\the@numberlist 157
\theglossary 5
\theglossaryentry 11, 207
\theglossarysubentry 12, 207
\theglsentrycounter 113, 114, 183, 185, 186, 322, 323
\theH 187
\theHglossaryentry 11
\theHglossarysubentry 12
\theHglsentrycounter 114, 183, 186
\thesection 33
\this@dialect 36, 379, 380
\toks@ 33, 34, 36, 46, 51, 53, 61, 116–121, 167–169, 189, 190, 209, 210, 216, 321, 341, 342
\toprule 283, 284
tracklang package 35, 379
\trans@languages 36
\translate 36, 37
\translatelet 16, 17, 31
translator package 16, 17, 26, 31, 35–37, 190
\ttfamily 6
\TX@trial 90
\typeout 20

U

\u 23
\uccode 200
\undef 67, 73, 190
\unskip 81, 275, 276, 329
\unvbox 286, 287
\usedictionary 36
\usepackage 197, 198

V

\v 23
\val 5, 6, 8, 26, 66, 204
\vbox 286, 287
\vsize 286, 287
\vskip 273, 286, 293, 311
\vss 286, 287

W

\warn@nomakeglossaries 173–175
\warn@noprintglossary 173–175, 193
\write 73, 95, 161–167, 171, 172, 175, 179, 193, 194, 323–327
\writeist 170, 171, 327

X

\x 216
\xatlevel@ 113
\xcapitalisewords 153
\xdef 78, 83–86, 192, 269
\xglsaccsupp 347
\xifinlistcs 194, 195, 199
xindy **381**
xindy 9, 13, 29, 30, 38, 39, 44, 47, 49, 51–53, 89, 120, 121, 160, 161, 163, 179, 184, 186, 193, 211, 260, 322
\xmakefirstuc 99–101, 107, 149, 151, 262
\xspace 218
xspace package 4, 217

Y

\year 161, 165, 323, 326

Z

\z@ 286