Documented Code For glossaries v3.04

Nicola L.C. Talbot

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

2012-11-18

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

`glossariesbegin.pdf` If you are a complete beginner, start with "The glossaries package: a guide for beginners".

`glossary2glossaries.pdf` If you are moving over from the obsolete glossary package, read "Upgrading from the glossary package to the glossaries package".

`glossaries-user.pdf` For the main user guide, read "glossaries.sty v3.04: 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.pdf This document is for advanced users wishing to know more about the inner workings of the glossaries package.

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

Contents

1 Main Package Code

1.1 Package Definition

This package requires LaTeX 2ε.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries}[2012/11/18 v3.04 (NLCT)]
```

Required packages:

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

\if@gls@docloaded

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

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

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

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

```
  \newcommand{\glsorg@glossary}{%
    \@bsphack
      \begingroup
        \@sanitize \glsorg@wrglossary
  }
```

\glsorg@wrglossary

```
  \newcommand{\glsorg@wrglossary}[1]{%
        \protected@write\@glossaryfile{}{%
          \string \glossaryentry{#1}{\thepage}}%
      \endgroup
    \@esphack
  }
```

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

```
  \let\glsorg@changes\changes
  \renewcommand{\changes}[3]{%
    \begingroup
      \let\glossary\glsorg@glossary
      \glsorg@changes{#1}{#2}{#3}%
    \endgroup
  }
```

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

\glsorg@theglossary

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

sorg@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

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}{*}
```

\@@glossaryseclabel

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

\glsautoprefix Prefix to add before label if automatically generated:

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

numberedsection

```
\define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
false,nolabel,autolabel}[nolabel]{%
  \ifcase\nr\relax
    \renewcommand*{\@@glossarysecstar}{*}%
    \renewcommand*{\@@glossaryseclabel}{}%
  \or
    \renewcommand*{\@@glossarysecstar}{}%
    \renewcommand*{\@@glossaryseclabel}{}%
  \or
    \renewcommand*{\@@glossarysecstar}{}%
    \renewcommand*{\@@glossaryseclabel}{%
      \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 subsection 1.18.)

ssary@default@style

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

style The default glossary style can be changed using the style package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the style key to set a style that is defined in another package. This package comes with some predefined styles that are defined in subsection 1.18.

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

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

lossaryentrynumbers

```
\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).

```
\DeclareOptionX{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
```

o@seeautonumberlist

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

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

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

\@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.

```
\DeclareOptionX{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.

```
\DeclareOptionX{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.

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

\@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.

```
\DeclareOptionX{notree}{\renewcommand*{\@gls@loadtree}{}}
```

nostyles Provide an option to suppress all the predefined styles (in the event that the user has custom styles that are not dependent on the predefined styles).

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

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

```
\newcommand*{\glspostdescription}{\ifglsnopostdot\else.\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 `\MakeUppercase` in definition of `\glossarymark`

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

entrycounter Defines a counter that can be used in the standard glossary styles to number each (main) entry. If true, this will define a counter called glossaryentry.

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

entrycounterwithin 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@counterwithin` The default value is no parent counter:

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

`subentrycounter` Define a counter that can be used in the standard glossary styles to number each level 1 entry. If true, this will define a counter called glossarysubentry.

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

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

```
\define@choicekey{glossaries.sty}{sort}{standard,def,use}{%
  \csname @gls@setupsort@#1\endcsname
}
```

`@setupsort@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`).

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

    \@gls@sanitizesort
    \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{}%
}
```

Set standard sort as the default:

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

`\glssortnumberfmt` 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%
}
```

\@gls@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{%
    \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{}%
}
```

\@gls@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
  }%
}
```

\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
}
```

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the type key in a key-value list). This was mainly done so that \loadglsentries can temporarily change \glsdefaulttype while it loads a file containing new glossary entries (see subsection 1.9).

\glsdefaulttype

```
\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}
```

The nomain option suppress the creation of the main glossary.

```
\DeclareOptionX{nomain}{%
   \let\glsdefaulttype\relax
   \renewcommand*{\glsdefmain}{}%
}
```

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

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

\@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}{}
```

\@addtoacronynlists

```
\newcommand*{\@addtoacronymlists}[1]{%
  \ifx\@glsacronymlists\@empty
    \protected@xdef\@glsacronymlists{#1}%
  \else
    \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
  \fi
}
```

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

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

glsIfListOfAcronyms \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
}
```

if@glsisacronymlist Convenient boolean.

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

@checkisacronymlist 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}{%
  \@addtoacronymlists{#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 subsection 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}%
}
```

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

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

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

The default values are defined as:

`\@gls@sanitizedesc`

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

`\@gls@sanitizename`

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

`@gls@sanitizesymbol`

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

`\@gls@sanitizesort`

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

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

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

```
\define@boolkey[gls]{sanitize}{description}[true]{%
\ifgls@sanitize@description
  \renewcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}%
\else
  \renewcommand*{\@gls@sanitizedesc}{}%
\fi
}
```

Similarly for the name key:

```
\define@boolkey[gls]{sanitize}{name}[true]{%
\ifgls@sanitize@name
  \renewcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}%
\else
  \renewcommand*{\@gls@sanitizename}{}%
\fi}
```

and for the symbol key:

```
\define@boolkey[gls]{sanitize}{symbol}[true]{%
\ifgls@sanitize@symbol
```

```
  \renewcommand*{\@gls@sanitizesymbol}{%
\@onelevel@sanitize\@glo@symbol}%
\else
  \renewcommand*{\@gls@sanitizesymbol}{}%
\fi}
```

and for the sort key:

```
\define@boolkey[gls]{sanitize}{sort}[true]{%
\ifgls@sanitize@sort
  \renewcommand*{\@gls@sanitizesort}{%
\@onelevel@sanitize\@glo@sort}%
\else
  \renewcommand*{\@gls@sanitizesort}{}%
\fi}
```

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

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

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

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

Set the default value:

```
\glstranslatefalse
  \@ifpackageloaded{translator}%
    {\glstranslatetrue}%
    {%
      \@ifpackageloaded{polyglossia}%
        {\glstranslatetrue}%
        {%
          \@ifpackageloaded{babel}{\glstranslatetrue}{}%
        }%
}
```

`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
```

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

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

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}{}%
}
```

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

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

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}{}%
}
```

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}{}%
}
```

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
```

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

```
\DeclareOptionX{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
```

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

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

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}}
```

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

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

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

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

Set default:

```
\glssavewritesfalse
```

\GlossariesWarning Prints a warning message.

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

sariesWarningNoLine Prints a warning message without the line number.

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

Define package option to suppress warnings

```
\DeclareOptionX{nowarn}{%
  \renewcommand*{\GlossariesWarning}[1]{}%
  \renewcommand*{\GlossariesWarningNoLine}[1]{}%
}
```

compatible-2.07

```
\define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{}
\csname glscompatible-2.07false\endcsname
```

Process package options:

```
\ProcessOptionsX
```

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

```
\ifglstranslate
  \@ifpackageloaded{polyglossia}%
  {%
```

polyglossia fakes babel so need to check for polyglossia first.

```
  }%
  {%
     \@ifpackageloaded{babel}%
     {%
         \IfFileExists{translator.sty}%
         {%
            \RequirePackage{translator}%
         }%
         {}%
     }%
     {}
  }
\fi
```

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

The same problem will also occur if a lower sectional unit is used, but this is less likely to happen. If it does, or if you change `\glscounter` to `section` later, you will have to specify a different counter for the entries that give rise to a `name{`⟨*section-level*⟩`.`⟨*n*⟩`.0}` non-existent warning (e.g. `\gls[counter=chapter]{label}`).

```
\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}{}}}%
  }%
}%
{}
```

`\@gls@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}
```

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

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

`\@disable@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 Default values

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

Main glossary title:

\glossaryname

```
\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:

glssymbolsgroupname

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

glsnumbersgroupname

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

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

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

\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.

dglossarytocaptions

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.)

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

```
\ifglstranslate
```

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

```
  \@ifpackageloaded{translator}{%
    \usedictionary{glossaries-dictionary}%
    \addglossarytocaptions{portuges}%
    \addglossarytocaptions{portuguese}%
    \addglossarytocaptions{brazil}%
    \addglossarytocaptions{brazilian}%
    \addglossarytocaptions{danish}%
    \addglossarytocaptions{dutch}%
    \addglossarytocaptions{afrikaans}%
    \addglossarytocaptions{english}%
    \addglossarytocaptions{UKenglish}%
    \addglossarytocaptions{USenglish}%
    \addglossarytocaptions{american}%
    \addglossarytocaptions{australian}%
    \addglossarytocaptions{british}%
    \addglossarytocaptions{canadian}%
    \addglossarytocaptions{newzealand}%
    \addglossarytocaptions{french}%
    \addglossarytocaptions{frenchb}%
    \addglossarytocaptions{francais}%
    \addglossarytocaptions{acadian}%
    \addglossarytocaptions{canadien}%
    \addglossarytocaptions{german}%
    \addglossarytocaptions{germanb}%
    \addglossarytocaptions{austrian}%
    \addglossarytocaptions{naustrian}%
    \addglossarytocaptions{ngerman}%
    \addglossarytocaptions{irish}%
    \addglossarytocaptions{italian}%
    \addglossarytocaptions{magyar}%
    \addglossarytocaptions{hungarian}%
    \addglossarytocaptions{polish}%
    \addglossarytocaptions{spanish}%
    \renewcommand*{\glssettoctitle}[1]{%
    \ifthenelse{\equal{#1}{main}}{%
      \translatelet{\glossarytoctitle}{Glossary}}{%
      \ifthenelse{\equal{#1}{acronym}}{%
        \translatelet{\glossarytoctitle}{Acronyms}}{%
        \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)}}%
  }{%

    \@ifpackageloaded{polyglossia}%
    {\RequirePackage{glossaries-polyglossia}}%
    {%
      \@ifpackageloaded{babel}{%
        \RequirePackage{glossaries-babel}}{}%
    }}
\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.

```
\newcommand*{\nopostdesc}{}
```

`\@nopostdesc` Suppress next description terminator.

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

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

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

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

```
\ifglsxindy
  \newcommand{\setStyleFile}[1]{%
    \renewcommand{\istfilename}{#1.xdy}}
```

```
\else
  \newcommand{\setStyleFile}[1]{%
    \renewcommand{\istfilename}{#1.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{\jobname.xdy}
\else
  \def\istfilename{\jobname.ist}
\fi
```

The `makeglossaries` Perl script picks up this name from the auxiliary file. If the name ends with `.xdy` it calls `xindy` otherwise it calls `makeindex`. Since its not required by 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}{.}
```

`\glsSetCompositor` 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.

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

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

`sSetAlphaCompositor` 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`.

`\glossarypreamble`

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

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

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

`\glossarypostamble`

```
\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}%
  }%
  \glossarymark{\glossarytoctitle}%
}
```

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

```
\ifcsundef{glossarymark}%
{%
  \ifglsucmark
    \newcommand{\glossarymark}[1]{%
       \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
    }
  \else
    \newcommand{\glossarymark}[1]{\@mkboth{#1}{#1}}
  \fi
}%
{%
  \GlossariesWarning{overriding \string\glossarymark}%
  \@ifclassloaded{memoir}%
  {
    \ifglsucmark
      \renewcommand{\glossarymark}[1]{%
         \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
      }
    \else
      \renewcommand{\glossarymark}[1]{%
        \markboth{\memUChead{#1}}{\memUChead{#1}}%
      }
    \fi
  }
  {
    \ifglsucmark
      \renewcommand{\glossarymark}[1]{%
         \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
      }
    \else
      \renewcommand{\glossarymark}[1]{\@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.

\setglossarysection

```
\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]{%
\ifx\@@glossarysecstar\@empty
  \csname\@@glossarysec\endcsname{#2}%
\else
  \csname\@@glossarysec\endcsname*{#2}%
  \@gls@toc{#1}{\@@glossarysec}%
\fi
\@@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.

\@p@glossarysection

```
\newcommand*{\@p@glossarysection}[2]{%
\glsclearpage
\phantomsection
\ifx\@@glossarysecstar\@empty
  \csname\@@glossarysec\endcsname{#2}%
\else
  \@gls@toc{#1}{\@@glossarysec}%
  \csname\@@glossarysec\endcsname*{#2}%
\fi
\@@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}{\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}{\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.

\glsnoxindywarning 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}%
}
```

\@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
```

\@xdyattributelist 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
}
```

\GlsAddXdyCounters Need to know all the counters that will be used in location numbers for Xindy. Argument may be a single counter name or a comma-separated list of counter names.

```
\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
```

d@glsaddxdycounters 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}%
}
```

\GlsAddXdyAttribute 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"\string~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
```

redefinedattributes 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}
}
\else
  \let\@gls@addpredefinedattributes\relax
\fi
```

\@xdyuseralphabets List of additional alphabets

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

\GlsAddXdyAlphabet `\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
```

ls@xdy@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.

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

```
\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
```

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

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

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

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

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

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

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

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

pendix-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"%
}
```

bic-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"%
}%
```

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

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

dyuserlocationnames List of additional user location names

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

End of xindy-only block:

```
\fi
```

`\GlsAddXdyLocation` `\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"#1\string"}%
  }
```

Only has an effect before \writeist:

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

ylocationclassorder Define location class order

```
\ifglsxindy
  \edef\@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.

yLocationClassOrder

```
\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
```

`\@xdyrequiredstyles` 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
```

`\findrootlanguage` The root language name is required by `xindy`. This information is for `makeglossaries` to pass to `xindy`. Since `\languagename` only stores the regional dialect rather than the root language name, some trickery is required to determine the root language.

```
\ifglsxindy
  \@ifpackageloaded{babel}{%
```

Need to parse babel.sty to determine the root language. This code was provided by Enrico Gregorio.

```
\def\findrootlanguage{\begingroup
  \escapechar=-1\relax
```

normalize \languagename to category 12 chars

```
    \edef\languagename{%
      \expandafter\string\csname\languagename\endcsname}%
```

disable babel.sty useless commands

```
    \def\NeedsTeXFormat##1[##2]{}%
    \def\ProvidesPackage##1[##2]{}%
    \let\LdfInit\relax
    \def\languageattribute##1##2{}%
```

change the meaning of \DeclareOption

```
    \def\DeclareOption##1##2{%
```

at \DeclareOption* we end

```
      \ifx##1*\expandafter\endinput\else
```

else we build a string with the first argument

```
      \edef\testlanguage{\expandafter\string\csname##1\endcsname}%
```

if \testlanguage and \languagename are the same we execute the second argument

```
      \ifx\testlanguage\languagename##2\fi
    \fi}
```

almost all options of babel are \input{⟨*name*⟩.ldf}

```
  \def\input##1{\stripldf##1}%
```

we put the root language name in \rootlanguagename

```
  \def\stripldf##1.ldf{\gdef\rootlanguagename{##1}}%
```

now input babel.sty, using the primitive \input

```
  \@@input babel.sty
  \endgroup}%
  }{%
```

hasn't been loaded, so check if has been loaded

```
    \@ifpackageloaded{ngerman}{%
       \def\findrootlanguage{%
         \def\rootlanguagename{german}}%
    }{%
```

Neither babel nor ngerman have been loaded, so assume the root language is English

```
       \def\findrootlanguage{%
         \def\rootlanguagename{english}}%
    }%
  }%
\fi
```

\rootlanguagename Set default root language to English.

```
\def\rootlanguagename{english}
```

\@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{}
```

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

```
\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{}
```

\GlsSetXdyCodePage Define command to set the code page.

```
\ifglsxindy
  \newcommand*{\GlsSetXdyCodePage}[1]{%
    \renewcommand*{\gls@codepage}{#1}%
  }
\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
      :before \string"\@glsfirstletter\string")}
  \else
    \def\@xdylettergroups{}
  \fi
\fi
```

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

```
  \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

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

\forallglossaries[⟨*glossary list*⟩]{⟨*cmd*⟩}{⟨*code*⟩}

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

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

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

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

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

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

\ifglossaryexists

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}%
}
```

\ifglsentryexists

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

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

where ⟨*label*⟩ is the entry's label.

```
\newcommand{\ifglsentryexists}[3]{%
  \ifcsundef{glo@#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]{\ifthenelse{\boolean{glo@#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 '#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 '#1' has already
    been defined}{}}{#2}%
}
```

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

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

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

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

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}{,}
```

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}[5][glg]{%
\ifglossaryexists{#2}{%
  \PackageError{glossaries}{Glossary type '#2' already exists}{%
  You can't define a new glossary called '#2' because it already
  exists}%
}{%
```

Check if default has been set

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

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 details of this new glossary type:

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

How to display this entry in the document text (uses \glsdisplay and \glsdisplayfirst by default). These can be redefined by the user later if required (see \defglsdisplay and \defglsdisplayfirst). These may already have been defined if this has been specified as a list of acronyms.

```
  \ifcsundef{gls@#2@display}%
  {%
    \expandafter\gdef\csname gls@#2@display\endcsname{\glsdisplay}%
  }%
  {}%
  \ifcsundef{gls@#2@displayfirst}%
  {%
    \expandafter\gdef\csname gls@#2@displayfirst\endcsname{%
      \glsdisplayfirst
    }%
  }%
  {}%
```

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]}}%
}
```

`\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
```

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

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

`descriptionplural`

```
\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}%
}
```

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

```
\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}{%
  \def\@glo@see{#1}%
  \@glo@seeautonumberlist
}
```

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}%
  \else
    \def\@glo@prefix{\glsnextpages}%
  \fi
}
```

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

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}}
```

\@glsdefaultplural Define command to set default plural.

```
\newcommand*{\@glsdefaultplural}{\@glo@text\glspluralsuffix}
```

s@missingnumberlist 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
```

\newglossaryentry 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}%
  {%
```

Store label

```
    \def\@glo@label{#1}%
```

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

```
    \let\@glo@name\@glsnoname

    \def\@glo@desc{%
      \PackageError{glossaries}
      {%
        description key required in \string\newglossaryentry\space
        for entry ‘\@glo@label’%
      }%
      {%
        You haven’t specified the entry description%
      }%
    }%
    \def\@glo@descplural{\@glo@desc}%
    \def\@glo@type{\glsdefaulttype}%
    \def\@glo@symbol{\relax}%
    \def\@glo@symbolplural{\@glo@symbol}%
    \def\@glo@text{\@glo@name}%
    \let\@glo@plural\@glsdefaultplural
```

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

```
    \let\@glo@first\relax
    \let\@glo@firstplural\relax
```

Set the default sort:

```
    \let\@glo@sort\@glsdefaultsort
```

Set the default counter:

```
    \def\@glo@counter{\@gls@getcounter{\@glo@type}}%
    \def\@glo@see{}%
    \def\@glo@parent{}%
    \def\@glo@prefix{}%
    \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 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}%
    }%
    {%
      \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
      \expandafter\xdef\csname glolist@\@glo@type\endcsname{\@glolist@{#1},}%
    }%
```

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@#1@parent\endcsname{}%
  \else
```

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

```
    \ifthenelse{\equal{#1}{\@glo@parent}}%
    {%
      \PackageError{glossaries}{Entry ‘#1’ can’t be its own parent}{}%
      \def\@glo@parent{}%
      \expandafter\gdef\csname glo@#1@parent\endcsname{}%
    }%
    {%
```

Check the parent exists:

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

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

```
        \expandafter\xdef\csname glo@#1@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\@glsdefaultplural
            \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 ‘#1’ - parent doesn’t exist%
          }%
          {%
            Parent entries must be defined before their children%
          }%
          \def\@glo@parent{}%
          \expandafter\gdef\csname glo@#1@parent\endcsname{}%
        }%
      }%
    \fi
```

Set the level for this entry

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

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

```
    \ifx\relax\@glo@firstplural
       \ifx\relax\@glo@first
          \def\@glo@firstplural{\@glo@plural}%
          \def\@glo@first{\@glo@text}%
       \else
          \def\@glo@firstplural{\@glo@first\glspluralsuffix}%
       \fi
    \else
       \ifx\relax\@glo@first
          \def\@glo@first{\@glo@text}%
       \fi
    \fi
```

Define commands associated with this entry:

```
    \expandafter
      \protected@xdef\csname glo@#1@text\endcsname{\@glo@text}%
    \expandafter
      \protected@xdef\csname glo@#1@plural\endcsname{\@glo@plural}%
    \expandafter
      \protected@xdef\csname glo@#1@first\endcsname{\@glo@first}%
    \expandafter
      \protected@xdef\csname glo@#1@firstpl\endcsname{\@glo@firstplural}%
    \expandafter
      \protected@xdef\csname glo@#1@type\endcsname{\@glo@type}%
    \expandafter
      \protected@xdef\csname glo@#1@counter\endcsname{\@glo@counter}%
    \expandafter
      \protected@xdef\csname glo@#1@useri\endcsname{\@glo@useri}%
    \expandafter
      \protected@xdef\csname glo@#1@userii\endcsname{\@glo@userii}%
    \expandafter
      \protected@xdef\csname glo@#1@useriii\endcsname{\@glo@useriii}%
    \expandafter
      \protected@xdef\csname glo@#1@useriv\endcsname{\@glo@useriv}%
    \expandafter
      \protected@xdef\csname glo@#1@userv\endcsname{\@glo@userv}%
    \expandafter
      \protected@xdef\csname glo@#1@uservi\endcsname{\@glo@uservi}%
    \expandafter
      \protected@xdef\csname glo@#1@short\endcsname{\@glo@short}%
    \expandafter
      \protected@xdef\csname glo@#1@shortpl\endcsname{\@glo@shortpl}%
    \expandafter
      \protected@xdef\csname glo@#1@long\endcsname{\@glo@long}%
    \expandafter
      \protected@xdef\csname glo@#1@longpl\endcsname{\@glo@longpl}%
    \@gls@sanitizename
    \expandafter\protected@xdef\csname glo@#1@name\endcsname{\@glo@name}%
```

Set default numberlist if not defined:

```
    \ifcsundef{glo@#1@numberlist}%
    {%
      \csxdef{glo@#1@numberlist}{\noexpand\@gls@missingnumberlist{\@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
    \@gls@sanitizedesc
    \expandafter\protected@xdef\csname glo@#1@desc\endcsname{\@glo@desc}%
    \expandafter\protected@xdef\csname glo@#1@descplural\endcsname{\@glo@descplural}%
```

Set the sort key for this entry:

```
    \@gls@defsort{\@glo@type}{#1}%

    \def\@glo@@symbol{\@glo@text}%
    \ifx\@glo@symbol\@glo@@symbol
      \let\@glo@symbol\@glo@text
    \fi
    \@gls@sanitizesymbol
    \expandafter\protected@xdef\csname glo@#1@symbol\endcsname{\@glo@symbol}%
    \expandafter\protected@xdef\csname glo@#1@symbolplural\endcsname{\@glo@symbolplural}%
```

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

```
    \expandafter\gdef\csname glo@#1@flagfalse\endcsname{%
      \expandafter\global\expandafter
        \let\csname ifglo@#1@flag\endcsname\iffalse
    }%
    \expandafter\gdef\csname glo@#1@flagtrue\endcsname{%
      \expandafter\global\expandafter
      \let\csname ifglo@#1@flag\endcsname\iftrue
    }%
    \csname glo@#1@flagfalse\endcsname
```

Sort out any cross-referencing if required.

```
    \ifx\@glo@see\@empty
    \else
      \protected@edef\@do@glssee{%
        \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
          \noexpand\@nil
        \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{#1}}%
      \@do@glssee
    \fi
  }%
```

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

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

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

```
  \@newglossaryentryposthook
}
```

lossaryentryprehook Allow extra information to be added to glossary entries:

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

ossaryentryposthook Allow extra information to be added to glossary entries:

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

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

```
\newcommand*{\glsmoveentry}[2]{%
  \edef\glo@type{\csname glo@#1@type\endcsname}%
  \def\glo@list{,}%
  \forglsentries[\glo@type]{\glo@label}%
   {%
     \ifthenelse{\equal{\glo@label}{#1}}{}{\eappto\glo@list{\glo@label,}}%
   }%
  \cslet{glolist@\glo@type}{\glo@list}%
  \csdef{glo@#1@type}{#2}%
}
```

@glossaryentryfield 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\\glossaryentryfield}
\else
  \newcommand*{\@glossaryentryfield}{\string\glossaryentryfield}
\fi
```

ossarysubentryfield 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\\glossarysubentryfield}
\else
  \newcommand*{\@glossarysubentryfield}{%
    \string\glossarysubentryfield}
\fi
```

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

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

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.

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

Add the font command. (The backslash needs to be escaped for `xindy`.)

```
\ifglsxindy
  \protected@edef\@glo@name{\string\\glsnamefont{\@@glo@name}}%
\else
  \protected@edef\@glo@name{\string\glsnamefont{\@@glo@name}}%
\fi
```

Get the description string and escape any special characters

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

Same again for the symbol

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

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{#1}{\@glo@name
     }{\@glo@desc}{\@glo@symbol}\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}{#1}{\@glo@name
      }{\@glo@desc}{\@glo@symbol}\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{#1}{\@glo@name}{\@glo@desc
      }{\@glo@symbol}%
    }%
  \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}{#1}{\@glo@name}{\@glo@desc
      }{\@glo@symbol}%
    }%
  \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:

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

`\glsreset`

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

As above, but with only a local effect:

`\glslocalreset`

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

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

`\glsunset`

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

As above, but with only a local effect:

`\glslocalunset`

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

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

`\glsresetall`

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

As above, but with only a local effect:

`\glslocalresetall`

```
\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:

`\glslocalunsetall`

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

1.9 Loading files containing glossary entries

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

`\loadglsentries[⟨type⟩]{⟨filename⟩}`

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

`\loadglsentries`

```
\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}
```

1.10 Using glossary entries in the text

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

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

\glstextformat

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

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

\glsdisplayfirst

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

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

\glsdisplay

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

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

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

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

\defglsdisplay

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

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

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

\defglsdisplayfirst

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

1.10.1 Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defglsdisplay` and `\defglsdisplayfirst`). It goes against the 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]{}
```

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]{}
```

Syntax:

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

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

There is also a starred version:

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

which is equivalent to `\glslink[hyper=false,`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

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

`\glslink`

```
\newrobustcmd*{\glslink}{%
\@ifstar\@sgls@link\@gls@@link}
```

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

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

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

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

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

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

`\@gls@link`

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

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

```
    \leavevmode
    \def\glslabel{#2}%
    \def\@glsnumberformat{glsnumberformat}%
    \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
    \KV@glslink@hypertrue
    \setkeys{glslink}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
    \@gls@saveentrycounter
```

Define sort key if necessary:

```
    \@gls@setsort{#2}%
```

```
    \@do@wrglossary{#2}%
    \ifKV@glslink@hyper
      \@glslink{\glolinkprefix#2}{\glstextformat{#3}}%
    \else
      \glstextformat{#3}\relax
    \fi
}
```

\glolinkprefix

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

\glsentrycounter Set default value of entry counter

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

ls@saveentrycounter 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 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
}
```

`\@set@glo@numformat` Set the formatting information in the format required by `makeindex`. The first argument is the format specified by the user (via the format key), the second argument is the name of the counter used to indicate the location, the third argument is a control sequence which stores the required format and the fourth argument (new to v3.0) is the hyper-prefix.

```
\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):

gls@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`:

`\@gls@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}
```

\@gls@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}
```

@gls@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}
```

\@gls@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}
```

`\@gls@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
}
```

s@xdycheckbackslash 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
```

\@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}%
{%
  \gdef\@glslink#1#2{#2}%
}%
{%
  \gdef\@glslink#1#2{\hyperlink{#1}{#2}}%
}
```

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

```
\newlength\gls@tmplen
\ifcsundef{hypertarget}%
{%
  \gdef\@glstarget#1#2{#2}%
}%
{%
  \gdef\@glstarget#1#2{%
    \settoheight{\gls@tmplen}{#2}%
    \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
  }%
}
```

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

\glsdisablehyper

```
\newcommand{\glsdisablehyper}{%
\renewcommand*\@glslink[2]{##2}%
\renewcommand*\@glstarget[2]{##2}}
```

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

\glsenablehyper

```
\newcommand{\glsenablehyper}{%
\renewcommand*\@glslink[2]{\hyperlink{##1}{##2}}%
\renewcommand*\@glstarget[2]{%
  \settoheight{\gls@tmplen}{##2}%
  \raisebox{\gls@tmplen}{\hypertarget{##1}{}}##2}}
```

Syntax:

\gls[⟨*options*⟩]{⟨*label*⟩}[⟨*insert text*⟩]

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

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

First determine if we are using the starred form:

`\gls`

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

Define the starred form:

`\@sgls`

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

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @display\endcsname
          {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
    }%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @displayfirst\endcsname
          {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
    }%
```

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.

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\@glo@text}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse
      {\(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
       \OR \NOT\boolean{glshyperfirst}
      }%
      {%
        \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
      }%
      {%
        \@gls@link[#1]{#2}{\@glo@text}%
      }%
    }%
```

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
}
```

\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}{\@ifstar\@sGls\@Gls}
```

Define the starred form:

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

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

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

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

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

```
  \ifglsused{#2}%
  {%
    \protected@edef\@glo@text{%
      \csname gls@\@glo@type @display\endcsname
        {\glsentrytext{#2}}{\glsentrydesc{#2}}%
        {\glsentrysymbol{#2}}{#3}}%
  }%
  {%
    \protected@edef\@glo@text{%
      \csname gls@\@glo@type @displayfirst\endcsname
        {\glsentryfirst{#2}}{\glsentrydesc{#2}}%
        {\glsentrysymbol{#2}}{#3}}%
  }%
```

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.

```
  \ifglsused{#2}%
  {%
    \@gls@link[#1]{#2}{%
    \expandafter\makefirstuc\expandafter{\@glo@text}}%
  }%
  {%
    \gls@checkisacronymlist\@glo@type
    \ifthenelse
    {%
      \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
      \OR \NOT\boolean{glshyperfirst}%
    }%
    {%
      \@gls@link[#1,hyper=false]{#2}{%
        \expandafter\makefirstuc\expandafter{\@glo@text}}%
    }%
    {%
      \@gls@link[#1]{#2}{%
        \expandafter\makefirstuc\expandafter{\@glo@text}}%
    }%
  }%
```

Indicate that this entry has now been used

```
  \ifKV@glslink@local
```

```
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
}
```

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

\GLS

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

Define the starred form:

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

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @display\endcsname
        {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}%
      }%
    }%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @displayfirst\endcsname
        {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}%
      }%
    }%
```

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.

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

```
      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse
      {%
        \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
        \OR \NOT\boolean{glshyperfirst}}{%
        \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
      }%
      {%
        \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
      }%
    }%
```

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
}
```

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

`\glspl`

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

Define the starred form:

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

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

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

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

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

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

```
      \def\@glo@text{%
        \csname gls@\@glo@type @display\endcsname
          {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
          {\glsentrysymbolplural{#2}}{#3}}%
    }%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @displayfirst\endcsname
          {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
          {\glsentrysymbolplural{#2}}{#3}}%
    }%
```

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.

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\@glo@text}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse
      {%
        \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
         \OR \NOT\boolean{glshyperfirst}%
      }%
      {%
        \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
      }%
      {%
        \@gls@link[#1]{#2}{\@glo@text}%
      }%
    }%
```

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
}
```

`\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}{\@ifstar\@sGlspl\@Glspl}
```

Define the starred form:

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

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \protected@edef\@glo@text{%
        \csname gls@\@glo@type @display\endcsname
          {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
          {\glsentrysymbolplural{#2}}{#3}}%
    }%
    {%
      \protected@edef\@glo@text{%
        \csname gls@\@glo@type @displayfirst\endcsname
          {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
          {\glsentrysymbolplural{#2}}{#3}}%
    }%
```

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.

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{%
        \expandafter\makefirstuc\expandafter{\@glo@text}}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse
      {%
        \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
        \OR \NOT\boolean{glshyperfirst}%
      }%
      {%
        \@gls@link[#1,hyper=false]{#2}{%
          \expandafter\makefirstuc\expandafter{\@glo@text}}%
      }%
      {%
        \@gls@link[#1]{#2}{%
          \expandafter\makefirstuc\expandafter{\@glo@text}}%
      }%
    }%
```

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
}
```

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

`\GLSpl`

```
\newrobustcmd*{\GLSpl}{\@ifstar\@sGLSpl\@GLSpl}
```

Define the starred form:

```
\newcommand*{\@sGLSpl}[1][]{\@GLSpl[hyper=false,#1]}
```

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @display\endcsname
        {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
        {\glsentrysymbolplural{#2}}{#3}%
      }%
    }%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @displayfirst\endcsname
        {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
        {\glsentrysymbolplural{#2}}{#3}%
      }%
    }%
```

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.

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse
      {%
        \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
        \OR \NOT\boolean{glshyperfirst}%
      }%
      {%
        \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
      }%
      {%
        \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
      }%
    }%
```

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
}
```

`\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}{\@ifstar\@sglsdisp\@glsdisp}
```

Define the starred form:

\@sgls

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

Defined the un-starred form.

\@glsdisp

```
\newcommand*{\@glsdisp}[3][]{%
  \glsdoifexists{#2}{%

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @display\endcsname
        {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
    }%
    {%
      \def\@glo@text{%
        \csname gls@\@glo@type @displayfirst\endcsname
        {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
    }%
```

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.

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\@glo@text}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse{\(\boolean{@glsisacronymlist}\AND
        \boolean{glsacrfootnote}\) \OR \NOT\boolean{glshyperfirst}}%
      {%
        \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
      }%
      {%
        \@gls@link[#1]{#2}{\@glo@text}%
      }%
    }%
```

Indicate that this entry has now been used

```
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
```

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

\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}{\@ifstar\@sglstext\@glstext}
```

Define the starred form:

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

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

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

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

Call \@gls@link

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

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

\GLStext

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

Define the starred form:

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

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

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

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

Call `\@gls@link`

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

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

`\Glstext`

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

Define the starred form:

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

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{%
    \expandafter\makefirstuc\expandafter{\@glo@text}#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}{\@ifstar\@sglsfirst\@glsfirst}
```

Define the starred form:

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

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

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

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

Call `\@gls@link`

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

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

`\Glsfirst`

```
\newrobustcmd*{\Glsfirst}{\@ifstar\@sGlsfirst\@Glsfirst}
```

Define the starred form:

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

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

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

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

Call `\@gls@link`

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

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

`\GLSfirst`

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

Define the starred form:

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

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsplural\@glsplural}
```

Define the starred form:

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

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

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

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

Call `\@gls@link`

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

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

`\Glsplural`

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

Define the starred form:

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

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

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

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

Call `\@gls@link`

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

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

`\GLSplural`

```
\newrobustcmd*{\GLSplural}{\@ifstar\@sGLSplural\@GLSplural}
```

Define the starred form:

```
\newcommand*{\@sGLSplural}[1][]{\@GLSplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsfirstplural\@glsfirstplural}
```

Define the starred form:

```
\newcommand*{\@sglsfirstplural}[1][]{\@glsfirstplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

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

`\Glsfirstplural`

```
\newrobustcmd*{\Glsfirstplural}{\@ifstar\@sGlsfirstplural\@Glsfirstplural}
```

Define the starred form:

```
\newcommand*{\@sGlsfirstplural}[1][]{\@Glsfirstplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

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

`\GLSfirstplural`

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

Define the starred form:

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

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

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

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

Call \@gls@link

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsname\@glsname}
```

Define the starred form:

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

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

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

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

Call \@gls@link

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

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

\Glsname

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

Define the starred form:

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

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

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

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

Call `\@gls@link`

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

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
\newrobustcmd*{\GLSname}{\@ifstar\@sGLSname\@GLSname}
```

Define the starred form:

```
\newcommand*{\@sGLSname}[1][]{\@GLSname[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsdesc\@glsdesc}
```

Define the starred form:

```
\newcommand*{\@sglsdesc}[1][]{\@glsdesc[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glsdesc` behaves like `\glsdesc` except that the first letter is converted to uppercase.

`\Glsdesc`

```
\newrobustcmd*{\Glsdesc}{\@ifstar\@sGlsdesc\@Glsdesc}
```

Define the starred form:

```
\newcommand*{\@sGlsdesc}[1][]{\@Glsdesc[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSdesc` behaves like `\glsdesc` except that the link text is converted to uppercase.

`\GLSdesc`

```
\newrobustcmd*{\GLSdesc}{\@ifstar\@sGLSdesc\@GLSdesc}
```

Define the starred form:

```
\newcommand*{\@sGLSdesc}[1][]{\@GLSdesc[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsdescplural\@glsdescplural}
```

Define the starred form:

```
\newcommand*{\@sglsdescplural}[1][]{\@glsdescplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glsdescplural` behaves like `\glsdescplural` except that the first letter is converted to uppercase.

`\Glsdescplural`

```
\newrobustcmd*{\Glsdescplural}{\@ifstar\@sGlsdescplural\@Glsdescplural}
```

Define the starred form:

```
\newcommand*{\@sGlsdescplural}[1][]{\@Glsdescplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
\newrobustcmd*{\GLSdescplural}{\@ifstar\@sGLSdescplural\@GLSdescplural}
```

Define the starred form:

```
\newcommand*{\@sGLSdescplural}[1][]{\@GLSdescplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglssymbol\@glssymbol}
```

Define the starred form:

```
\newcommand*{\@sglssymbol}[1][]{\@glssymbol[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
\newrobustcmd*{\Glssymbol}{\@ifstar\@sGlssymbol\@Glssymbol}
```

Define the starred form:

```
\newcommand*{\@sGlssymbol}[1][]{\@Glssymbol[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
\newrobustcmd*{\GLSsymbol}{\@ifstar\@sGLSsymbol\@GLSsymbol}
```

Define the starred form:

```
\newcommand*{\@sGLSsymbol}[1][]{\@GLSsymbol[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglssymbolplural\@glssymbolplural}
```

Define the starred form:

```
\newcommand*{\@sglssymbolplural}[1][]{\@glssymbolplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glssymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`\Glssymbolplural`

```
\newrobustcmd*{\Glssymbolplural}{\@ifstar\@sGlssymbolplural\@Glssymbolplural}
```

Define the starred form:

```
\newcommand*{\@sGlssymbolplural}[1][]{\@Glssymbolplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSsymbolplural` behaves like `\glssymbolplural` except that the link text is converted to uppercase.

`\GLSsymbolplural`

```
\newrobustcmd*{\GLSsymbolplural}{\@ifstar\@sGLSsymbolplural\@GLSsymbolplural}
```

Define the starred form:

```
\newcommand*{\@sGLSsymbolplural}[1][]{\@GLSsymbolplural[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsuseri\@glsuseri}
```

Define the starred form:

```
\newcommand*{\@sglsuseri}[1][]{\@glsuseri[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
\newrobustcmd*{\Glsuseri}{\@ifstar\@sGlsuseri\@Glsuseri}
```

Define the starred form:

```
\newcommand*{\@sGlsuseri}[1][]{\@Glsuseri[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
\newrobustcmd*{\GLSuseri}{\@ifstar\@sGLSuseri\@GLSuseri}
```

Define the starred form:

```
\newcommand*{\@sGLSuseri}[1][]{\@GLSuseri[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsuserii\@glsuserii}
```

Define the starred form:

```
\newcommand*{\@sglsuserii}[1][]{\@glsuserii[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glsuserii` behaves like `\glsuserii` except that the first letter is converted to uppercase.

`\Glsuserii`

```
\newrobustcmd*{\Glsuserii}{\@ifstar\@sGlsuserii\@Glsuserii}
```

Define the starred form:

```
\newcommand*{\@sGlsuserii}[1][]{\@Glsuserii[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSuserii` behaves like `\glsuserii` except that the link text is converted to uppercase.

`\GLSuserii`

```
\newrobustcmd*{\GLSuserii}{\@ifstar\@sGLSuserii\@GLSuserii}
```

Define the starred form:

```
\newcommand*{\@sGLSuserii}[1][]{\@GLSuserii[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsuseriii\@glsuseriii}
```

Define the starred form:

```
\newcommand*{\@sglsuseriii}[1][]{\@glsuseriii[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

`\Glsuseriii`

```
\newrobustcmd*{\Glsuseriii}{\@ifstar\@sGlsuseriii\@Glsuseriii}
```

Define the starred form:

```
\newcommand*{\@sGlsuseriii}[1][]{\@Glsuseriii[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
\newrobustcmd*{\GLSuseriii}{\@ifstar\@sGLSuseriii\@GLSuseriii}
```

Define the starred form:

```
\newcommand*{\@sGLSuseriii}[1][]{\@GLSuseriii[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsuseriv\@glsuseriv}
```

Define the starred form:

```
\newcommand*{\@sglsuseriv}[1][]{\@glsuseriv[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glsuseriv` behaves like `\glsuseriv` except that the first letter is converted to uppercase.

`\Glsuseriv`

```
\newrobustcmd*{\Glsuseriv}{\@ifstar\@sGlsuseriv\@Glsuseriv}
```

Define the starred form:

```
\newcommand*{\@sGlsuseriv}[1][]{\@Glsuseriv[hyper=false,#1]}
```

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

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

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

Call \@gls@link

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

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
\newrobustcmd*{\GLSuseriv}{\@ifstar\@sGLSuseriv\@GLSuseriv}
```

Define the starred form:

```
\newcommand*{\@sGLSuseriv}[1][]{\@GLSuseriv[hyper=false,#1]}
```

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

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

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

Call \@gls@link

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsuserv\@glsuserv}
```

Define the starred form:

```
\newcommand*{\@sglsuserv}[1][]{\@glsuserv[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\Glsuserv` behaves like `\glsuserv` except that the first letter is converted to uppercase.

`\Glsuserv`

```
\newrobustcmd*{\Glsuserv}{\@ifstar\@sGlsuserv\@Glsuserv}
```

Define the starred form:

```
\newcommand*{\@sGlsuserv}[1][]{\@Glsuserv[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSuserv` behaves like `\glsuserv` except that the link text is converted to uppercase.

`\GLSuserv`

```
\newrobustcmd*{\GLSuserv}{\@ifstar\@sGLSuserv\@GLSuserv}
```

Define the starred form:

```
\newcommand*{\@sGLSuserv}[1][]{\@GLSuserv[hyper=false,#1]}
```

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

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

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

Call \@gls@link

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#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}{\@ifstar\@sglsuservi\@glsuservi}
```

Define the starred form:

```
\newcommand*{\@sglsuservi}[1][]{\@glsuservi[hyper=false,#1]}
```

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

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

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

Call \@gls@link

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

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
\newrobustcmd*{\Glsuservi}{\@ifstar\@sGlsuservi\@Glsuservi}
```

Define the starred form:

```
\newcommand*{\@sGlsuservi}[1][]{\@Glsuservi[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

`\GLSuservi` behaves like `\glsuservi` except that the link text is converted to uppercase.

`\GLSuservi`

```
\newrobustcmd*{\GLSuservi}{\@ifstar\@sGLSuservi\@GLSuservi}
```

Define the starred form:

```
\newcommand*{\@sGLSuservi}[1][]{\@GLSuservi[hyper=false,#1]}
```

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

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

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

Call `\@gls@link`

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

Now deal with acronym related keys. First the short form:

`\acrshort`

```
\newrobustcmd*{\acrshort}{\@ifstar\s@acrshort\ns@acrshort}
```

Define the starred form:

```
\newcommand*{\s@acrshort}[2][]{%
  \new@ifnextchar[{\@acrshort{hyper=false,#1}{#2}}%
                  {\@acrshort{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\Acrshort`

```
\newrobustcmd*{\Acrshort}{\@ifstar\s@Acrshort\ns@Acrshort}
```

Define the starred form:

```
\newcommand*{\s@Acrshort}[2][]{%
  \new@ifnextchar[{\@Acrshort{hyper=false,#1}{#2}}%
                  {\@Acrshort{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\ACRshort`

```
\newrobustcmd*{\ACRshort}{\@ifstar\s@ACRshort\ns@ACRshort}
```

Define the starred form:

```
\newcommand*{\s@ACRshort}[2][]{%
  \new@ifnextchar[{\@ACRshort{hyper=false,#1}{#2}}%
                  {\@ACRshort{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

Short plural:

`\acrshortpl`

```
\newrobustcmd*{\acrshortpl}{\@ifstar\s@acrshortpl\ns@acrshortpl}
```

Define the starred form:

```
\newcommand*{\s@acrshortpl}[2][]{%
  \new@ifnextchar[{\@acrshortpl{hyper=false,#1}{#2}}%
                  {\@acrshortpl{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\Acrshortpl`

```
\newrobustcmd*{\Acrshortpl}{\@ifstar\s@Acrshortpl\ns@Acrshortpl}
```

Define the starred form:

```
\newcommand*{\s@Acrshortpl}[2][]{%
  \new@ifnextchar[{\@Acrshortpl{hyper=false,#1}{#2}}%
                  {\@Acrshortpl{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\ACRshortpl`

```
\newrobustcmd*{\ACRshortpl}{\@ifstar\s@ACRshortpl\ns@ACRshortpl}
```

Define the starred form:

```
\newcommand*{\s@ACRshortpl}[2][]{%
  \new@ifnextchar[{\@ACRshortpl{hyper=false,#1}{#2}}%
                  {\@ACRshortpl{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\acrlong`

```
\newrobustcmd*{\acrlong}{\@ifstar\s@acrlong\ns@acrlong}
```

Define the starred form:

```
\newcommand*{\s@acrlong}[2][]{%
  \new@ifnextchar[{\@acrlong{hyper=false,#1}{#2}}%
                  {\@acrlong{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\Acrlong`

```
\newrobustcmd*{\Acrlong}{\@ifstar\s@Acrlong\ns@Acrlong}
```

Define the starred form:

```
\newcommand*{\s@Acrlong}[2][]{%
  \new@ifnextchar[{\@Acrlong{hyper=false,#1}{#2}}%
                  {\@Acrlong{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\ACRlong`

```
\newrobustcmd*{\ACRlong}{\@ifstar\s@ACRlong\ns@ACRlong}
```

Define the starred form:

```
\newcommand*{\s@ACRlong}[2][]{%
  \new@ifnextchar[{\@ACRlong{hyper=false,#1}{#2}}%
                  {\@ACRlong{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

Short plural:

`\acrlongpl`

```
\newrobustcmd*{\acrlongpl}{\@ifstar\s@acrlongpl\ns@acrlongpl}
```

Define the starred form:

```
\newcommand*{\s@acrlongpl}[2][]{%
  \new@ifnextchar[{\@acrlongpl{hyper=false,#1}{#2}}%
                  {\@acrlongpl{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\Acrlongpl`

```
\newrobustcmd*{\Acrlongpl}{\@ifstar\s@Acrlongpl\ns@Acrlongpl}
```

Define the starred form:

```
\newcommand*{\s@Acrlongpl}[2][]{%
  \new@ifnextchar[{\@Acrlongpl{hyper=false#1}{#2}}%
                  {\@Acrlongpl{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

`\ACRlongpl`

```
\newrobustcmd*{\ACRlongpl}{\@ifstar\s@ACRlongpl\ns@ACRlongpl}
```

Define the starred form:

```
\newcommand*{\s@ACRlongpl}[2][]{%
  \new@ifnextchar[{\@ACRlongpl{hyper=false,#1}{#2}}%
                  {\@ACRlongpl{hyper=false,#1}{#2}[]}%
}
```

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

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

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

Call `\@gls@link`

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

1.10.2 Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

Get the entry name (as specified by the name key when the entry was defined). The argument is the label associated with the entry. Note that unless you used `name=false` in the sanitize package option you may get unexpected results if the name key contains any commands.

`\glsentryname`

```
\newcommand*{\glsentryname}[1]{\csname glo@#1@name\endcsname}
```

\Glsentryname

```
\newcommand*{\Glsentryname}[1]{%
\protected@edef\@glo@text{\csname glo@#1@name\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the sanitize package option you may get unexpected results if the description key contained any commands.

\glsentrydesc

```
\newcommand*{\glsentrydesc}[1]{\csname glo@#1@desc\endcsname}
```

\Glsentrydesc

```
\newcommand*{\Glsentrydesc}[1]{%
\protected@edef\@glo@text{\csname glo@#1@desc\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

\glsentrydescplural

```
\newcommand*{\glsentrydescplural}[1]{%
\csname glo@#1@descplural\endcsname}
```

\Glsentrydescplural

```
\newcommand*{\Glsentrydescplural}[1]{%
\protected@edef\@glo@text{\csname glo@#1@descplural\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

\glsentrytext

```
\newcommand*{\glsentrytext}[1]{\csname glo@#1@text\endcsname}
```

\Glsentrytext

```
\newcommand*{\Glsentrytext}[1]{%
\protected@edef\@glo@text{\csname glo@#1@text\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form:

\glsentryplural

```
\newcommand*{\glsentryplural}[1]{\csname glo@#1@plural\endcsname}
```

\Glsentryplural

```
\newcommand*{\Glsentryplural}[1]{%
\protected@edef\@glo@text{\csname glo@#1@plural\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the symbol associated with this entry. The argument is the label associated with the entry. Note that unless you used `symbol=false` in the sanitize package option you may get unexpected results if the symbol key contained any commands.

\glsentrysymbol

```
\newcommand*{\glsentrysymbol}[1]{\csname glo@#1@symbol\endcsname}
```

\Glsentrysymbol

```
\newcommand*{\Glsentrysymbol}[1]{%
\protected@edef\@glo@text{\csname glo@#1@symbol\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

lsentrysymbolplural

```
\newcommand*{\glsentrysymbolplural}[1]{%
\csname glo@#1@symbolplural\endcsname}
```

lsentrysymbolplural

```
\newcommand*{\Glsentrysymbolplural}[1]{%
\protected@edef\@glo@text{\csname glo@#1@symbolplural\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

\glsentryfirst

```
\newcommand*{\glsentryfirst}[1]{\csname glo@#1@first\endcsname}
```

\Glsentryfirst

```
\newcommand*{\Glsentryfirst}[1]{%
\protected@edef\@glo@text{\csname glo@#1@first\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form (as specified by the firstplural key when the entry was defined).

glsentryfirstplural

```
\newcommand*{\glsentryfirstplural}[1]{%
\csname glo@#1@firstpl\endcsname}
```

Glsentryfirstplural

```
\newcommand*{\Glsentryfirstplural}[1]{%
\protected@edef\@glo@text{\csname glo@#1@firstpl\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
\newcommand*{\glsentrytype}[1]{\csname glo@#1@type\endcsname}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
\newcommand*{\glsentrysort}[1]{\csname glo@#1@sort\endcsname}
```

\glsentryuseri Get the first user key (as specified by the user1 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuseri}[1]{\csname glo@#1@useri\endcsname}
```

\Glsentryuseri

```
\newcommand*{\Glsentryuseri}[1]{%
\protected@edef\@glo@text{\csname glo@#1@useri\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuserii Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuserii}[1]{\csname glo@#1@userii\endcsname}
```

\Glsentryuserii

```
\newcommand*{\Glsentryuserii}[1]{%
\protected@edef\@glo@text{\csname glo@#1@userii\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriii Get the third user key (as specified by the user3 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuseriii}[1]{\csname glo@#1@useriii\endcsname}
```

\Glsentryuseriii

```
\newcommand*{\Glsentryuseriii}[1]{%
\protected@edef\@glo@text{\csname glo@#1@useriii\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriv Get the fourth user key (as specified by the user4 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuseriv}[1]{\csname glo@#1@useriv\endcsname}
```

\Glsentryuseriv

```
\newcommand*{\Glsentryuseriv}[1]{%
\protected@edef\@glo@text{\csname glo@#1@useriv\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\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]{\csname glo@#1@userv\endcsname}
```

\Glsentryuserv

```
\newcommand*{\Glsentryuserv}[1]{%
\protected@edef\@glo@text{\csname glo@#1@userv\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuservi Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryuservi}[1]{\csname glo@#1@uservi\endcsname}
```

\Glsentryuservi

```
\newcommand*{\Glsentryuservi}[1]{%
\protected@edef\@glo@text{\csname glo@#1@uservi\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryshort Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryshort}[1]{\csname glo@#1@short\endcsname}
```

\Glsentryshort

```
\newcommand*{\Glsentryshort}[1]{%
\protected@edef\@glo@text{\csname glo@#1@short\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryshortpl Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentryshortpl}[1]{\csname glo@#1@shortpl\endcsname}
```

\Glsentryshortpl

```
\newcommand*{\Glsentryshortpl}[1]{%
\protected@edef\@glo@text{\csname glo@#1@shortpl\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentrylong Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentrylong}[1]{\csname glo@#1@long\endcsname}
```

\Glsentrylong

```
\newcommand*{\Glsentrylong}[1]{%
\protected@edef\@glo@text{\csname glo@#1@long\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentrylongpl Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
\newcommand*{\glsentrylongpl}[1]{\csname glo@#1@longpl\endcsname}
```

\Glsentrylongpl

```
\newcommand*{\Glsentrylongpl}[1]{%
\protected@edef\@glo@text{\csname glo@#1@longpl\endcsname}%
\expandafter\makefirstuc\expandafter{\@glo@text}}
```

Short cut macros to access full form:

\glsentryfull

```
\newcommand*{\glsentryfull}[1]{%
  \glsentrylong{#1}\space(\glsentryshort{#1})%
}
```

\Glsentryfull

```
\newcommand*{\Glsentryfull}[1]{%
  \Glsentrylong{#1}\space(\glsentryshort{#1})%
}
```

\glsentryfullpl

```
\newcommand*{\glsentryfullpl}[1]{%
  \glsentrylongpl{#1}\space(\glsentryshortpl{#1})%
}
```

\Glsentryfullpl

```
\newcommand*{\Glsentryfullpl}[1]{%
  \Glsentrylongpl{#1}\space(\glsentryshortpl{#1})%
}
```

\glsentrynumberlist Displays the number list as is.

```
\newcommand*{\glsentrynumberlist}[1]{%
  \glsdoifexists{#1}%
  {%
    \csname glo@#1@numberlist\endcsname
  }%
}
```

lsdisplaynumberlist 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
         \def\@glo@label{#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}{, }
```

`\glsnumlistlastsep`

```
\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#2}{#1}}
```

1.11 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][]{%
  \glsdoifexists{#2}%
  {%
    \def\@glsnumberformat{glsnumberformat}%
    \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
    \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
    \@gls@saveentrycounter
    \@do@wrglossary{#2}%
  }%
}
```

`\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}}%
}
```

1.12 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The makeindex actual character (usually @) is redefined to be a ?, to allow internal commands to be written to the glossary file output file.

The special characters are stored in \@gls@actualchar, \@gls@encapchar, \@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\}}
```

\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"}
```

\@glsfirstletter Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
\ifglsxindy
  \newcommand*{\@glsfirstletter}{A}
\fi
```

stLetterAfterDigits Sets the first letter to come after the digits 0,...,9.

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

\@glsminrange Define the minimum number of successive location references to merge into a range.

```
\newcommand*{\@glsminrange}{2}
```

etXdyMinRangeLength 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{%
```

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 were ⟨*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")}%
```

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\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)}%
```

Specify what to put between letter groups

```
  \write\glswrite{(markup-letter-group-list
      :sep \string"\string\glsgroupskip\string~n\string")}%
```

Specify what to put between entries

```
    \write\glswrite{(markup-indexentry
        :open \string"\string\relax \string\glsresetentrylist
           \string~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\string~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}%
```

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"}
  \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
    \closeout\glswrite
    \let\writeist\relax
  }
\fi
```

The command `\noist` will suppress the creation of the `.ist` file. Obviously you need to use this command before `\writeist` to have any effect.

`\noist`

```
\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}%
}
```

`rn@nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
\newcommand*{\warn@nomakeglossaries}{%
  \GlossariesWarningNoLine{\string\makeglossaries\space
  hasn't been used,^^Jthe glossaries will not be updated}%
}
```

`\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}{%
```

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
```

Suppress warning about no \makeglossaries

```
  \let\warn@nomakeglossaries\relax
```

Declare list parser for \glsdisplaynumberlist

```
  \ifglssavenumberlist
    \edef\@gls@dodeflistparser{\noexpand\DeclareListParser
      {\noexpand\glsnumlistparser}{\delimN}}%
    \@gls@dodeflistparser
  \fi
}
```

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
}
```

1.13 Writing information to associated files

\glswrite The write used for style file also used for all other output files if savewrites=true.

```
\newwrite\glswrite
```

\istfile Deprecated.

```
\def\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

```
\ifglssavewrites
  \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
    }%
  }
\else
  \let\glswritefiles\relax
```

```
\fi
```

The `\glossary` command is redefined so that it takes an optional argument ⟨*type*⟩ to specify the glossary type (use `\glsdefaulttype` glossary by default). This shouldn't be used at user level as `\glslink` sets the correct format. The associated number should be stored in `\theglsentrycounter` before using `\glossary`.

`\glossary`

```
\renewcommand*{\glossary}[1][\glsdefaulttype]{%
  \@glossary[#1]%
}
```

Define internal `\@glossary` to ignore its argument. This gets redefined in `\@makeglossary`. This is defined to just `\index` as memoir changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.)

`\@glossary`

```
\def\@glossary[#1]{\index}
```

This is a convenience command to set `\@glossary`. It is used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`\@gls@renewglossary`

```
\newcommand{\@gls@renewglossary}{%
  \gdef\@glossary[##1]{\@bsphack\begingroup\@wrglossary{##1}}%
  \let\@gls@renewglossary\@empty
}
```

The `\@wrglossary` command is redefined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in `\glslink`).

`\@wrglossary`

```
\renewcommand*{\@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}%
    }%
    {%
       \GlossariesWarning{No file defined for glossary ‘#1’}%
    }%
  \fi
  \endgroup\@esphack
}
```

\@do@wrglossary

```
\newcommand*{\@do@wrglossary}[1]{%
  \ifglsindexonlyfirst
    \ifglsused{#1}{}{\@@do@wrglossary{#1}}%
  \else
    \@@do@wrglossary{#1}%
  \fi
}
```

@protected@pagefmts List of page formats to be protected against expansion.

```
\newcommand{\gls@protected@pagefmts}{%
  \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage
}
```

blepagerefexpansion

```
\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@romanpage

```
\newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

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.

```
\newcommand*{\@@do@wrglossary}[1]{%
  \begingroup
% First a bit of hackery to prevent premature
% expansion of \cs{c@page}. Store original definitions:
%\changes{3.04}{2012-11-18}{modified to compensate for possible
%incorrect page number}
%    \begin{macrocode}
    \let\orgnumber\number
    \let\orgromannumeral\romannumeral
    \let\orgalph\@alph
    \let\orgAlph\@Alph
```

Redefine:

```
    \def\the##1{%
      \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
    \def\number##1{%
      \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
    \def\romannumeral##1{%
      \ifx##1\c@page \gls@romanpage\else\orgnumber##1\fi}%
    \def\@alph##1{%
      \ifx##1\c@page \gls@alphpage\else\orgnumber##1\fi}%
    \def\@Alph##1{%
      \ifx##1\c@page \gls@Alphpage\else\orgnumber##1\fi}%
```

Prevent expansion:

```
    \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
    \protected@xdef\@glslocref{\theglsentrycounter}%
  \endgroup
```

Escape any special characters

```
  \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
  \expandafter\ifx\theHglsentrycounter\theglsentrycounter
    \def\@glo@counterprefix{}%
  \else
    \protected@edef\@glsHlocref{\theHglsentrycounter}%
    \@gls@checkmkidxchars\@glsHlocref
    \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
      {\@glslocref}{\@glsHlocref}%
    }%
    \@do@gls@getcounterprefix
  \fi
```

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.

```
    \glossary[\csname glo@#1@type\endcsname]{%
```

```
  (indexentry :tkey (\csname glo@#1@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.

```
  \glossary[\csname glo@#1@type\endcsname]{%
  \string\glossaryentry{\csname glo@#1@index\endcsname
   \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
 \fi
}
```

`ls@getcounterprefix` Get the prefix that needs to be prepended to counter in order to get the hyper counter. (For example, with the standard article class and hyperref, `\theequation` needs to be prefixed with ⟨*section num*⟩|.| to get the equivalent `\theHequation`.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```
\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
 \fi
}
```

1.14 Glossary Entry Cross-References

`\@do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry's label, the second must be in the form `[`⟨*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
  \glossary[\csname glo@#1@type\endcsname]{%
    (indexentry
      :tkey (\csname glo@#1@index\endcsname)
      :xref (\string"\@gls@xref\string")
      :attr \string"see\string"
    )
  }%
\else
  \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
    \def#1{#2#3}%
  \else
    \def#1{{#2#3}}%
  \fi
}
```

`\glssee` `\glssee{`⟨*label*⟩`}{`⟨*cross-ref list*⟩`}`

```
\newcommand*{\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.

```
\newcommand*{\glsseeformat}[3][\seename]{\emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{`⟨*list*⟩`}` formats list of entry labels.

```
\newcommand*{\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

```
    \glsseeitem{\@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 entires in a cross-referencing list.

```
\newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{`⟨*label*⟩`}` formats individual entry in a cross-referencing list.

```
\newcommand*{\glsseeitem}[1]{\glshyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glsentrytext` instead of `\glsentryname`. (To avoid problems with the name key being sanitized.)

```
\newcommand*{\glsseeitemformat}[1]{\glsentrytext{#1}}
```

1.15 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[`⟨*key-val list*⟩`]`. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`gls@save@numberlist` Provide command to store number list.

```
\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
}
```

arn@noprintglossary Warn the user if they have forgotten \printglossaries or \printglossary. (Will be suppressed if there is at least one occurance of \printglossary. There is no check to ensure that there is a \printglossary for each defined glossary.)

```
\def\warn@noprintglossary{%
  \GlossariesWarningNoLine{No \string\printglossary\space
    or \string\printglossaries\space
    found.^^JThis document will not have a glossary}%
}
```

\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.

```
  \GlossariesWarning{Overriding \string\printglossary}%
  \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]{%
```

If xindy is being used, need to find the root language for makeglossaries to pass to xindy.

```
  \ifglsxindy\findrootlanguage\fi
```

Set up defaults.

```
  \def\@glo@type{\glsdefaulttype}%
  \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
  \def\glossarytoctitle{\glossarytitle}%
  \let\org@glossarytitle\glossarytitle
  \def\@glossarystyle{}%
  \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
```

Determine settings specified in the optional argument.

```
    \setkeys{printgloss}{#1}%
```

If title has been set, but toctitle hasn't, make toctitle the same as given title (rather than the title used when the glossary was defined)

```
  \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:

```
    \let\gls@org@glossaryentryfield\glossaryentryfield
    \let\gls@org@glossarysubentryfield\glossarysubentryfield
    \renewcommand{\glossaryentryfield}[1]{%
      \gdef\glscurrententrylabel{##1}%
      \gls@org@glossaryentryfield{##1}%
    }%
    \renewcommand{\glossarysubentryfield}[2]{%
      \gdef\glscurrententrylabel{##2}%
      \gls@org@glossarysubentryfield{##1}{##2}%
    }%
```

Some macros may end up being expanded into internals in the glossary, so need to make @ a letter.

```
    \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{%
            \noexpand\immediate\noexpand\write\@auxout{%
              \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
          }%
        }%
      }%
      {%
        \edef\@do@auxoutstuff{%
          \noexpand\AtEndDocument{%
            \noexpand\immediate\noexpand\write\@auxout{%
              \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
                @language\endcsname}}%
          }%
        }%
      }%
      \@do@auxoutstuff
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
           \noexpand\immediate\noexpand\write\@auxout{%
            \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
        }%
      }%
      \@do@auxoutstuff
    \fi
  \egroup
```

Reset `\glossaryentrynumbers`

```
  \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
  \global\let\warn@noprintglossary\relax
}
```

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]}%
}
```

The keys that can be used in the optional argument to `\printglossary` 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{\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}[nolabel]{%
\ifcase\nr\relax
  \renewcommand*{\@@glossarysecstar}{*}%
  \renewcommand*{\@@glossaryseclabel}{}%
\or
  \renewcommand*{\@@glossarysecstar}{}%
  \renewcommand*{\@@glossaryseclabel}{}%
\or
  \renewcommand*{\@@glossarysecstar}{}%
  \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
\fi}
```

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}
```

\@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}}
```

\glsresetentrylist 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}{}
```

glossaryentry If the entrycounter package option has been used, define a counter to number each level 0 entry.

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

glossarysubentry If the subentrycounter package option has been used, define a counter to number each level 1 entry.

```
\ifglssubentrycounter
  \ifglsentrycounter
    \newcounter{glossarysubentry}[glossaryentry]
  \else
```

```
    \newcounter{glossarysubentry}
  \fi
  \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
\fi
```

esetsubentrycounter Resets the glossarysubentry counter.

```
\ifglssubentrycounter
  \newcommand*{\glsresetsubentrycounter}{%
    \setcounter{glossarysubentry}{0}%
  }
\else
  \newcommand*{\glsresetsubentrycounter}{}
\fi
```

esetsubentrycounter Resets the glossarentry counter.

```
\ifglsentrycounter
  \newcommand*{\glsresetentrycounter}{%
    \setcounter{glossaryentry}{0}%
  }
\else
  \newcommand*{\glsresetentrycounter}{}
\fi
```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```
\ifglsentrycounter
  \newcommand*{\glsstepentry}[1]{%
    \refstepcounter{glossaryentry}%
    \label{glsentry-#1}%
  }
\else
  \newcommand*{\glsstepentry}[1]{}
\fi
```

\glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
\ifglssubentrycounter
  \newcommand*{\glsstepsubentry}[1]{%
    \def\currentglssubentry{#1}%
    \refstepcounter{glossarysubentry}%
    \label{glsentry-#1}%
  }
\else
  \newcommand*{\glsstepsubentry}[1]{}
\fi
```

\glsrefentry Reference the entry or sub-entry counter if in use, otherwise just do \gls.

```
\ifglsentrycounter
```

```
  \newcommand*{\glsrefentry}[1]{\ref{glsentry-#1}}
\else
  \ifglssubentrycounter
    \newcommand*{\glsrefentry}[1]{\ref{glsentry-#1}}
  \else
    \newcommand*{\glsrefentry}[1]{\gls{#1}}
  \fi
\fi
```

lsentrycounterlabel Defines how to display the glossaryentry counter.

```
\ifglsentrycounter
  \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
\else
  \newcommand*{\glsentrycounterlabel}{}
\fi
```

ubentrycounterlabel Defines how to display the glossarysubentry counter.

```
\ifglssubentrycounter
  \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
\else
  \newcommand*{\glssubentrycounterlabel}{}
\fi
```

\glsentryitem Step and display glossaryentry counter, if appropriate.

```
\ifglsentrycounter
  \newcommand*{\glsentryitem}[1]{%
    \glsstepentry{#1}\glsentrycounterlabel
  }
\else
  \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
\fi
```

\glssubentryitem Step and display glossarysubentry counter, if appropriate.

```
\ifglssubentrycounter
  \newcommand*{\glssubentryitem}[1]{%
    \glsstepsubentry{#1}\glssubentrycounterlabel
  }
\else
  \newcommand*{\glssubentryitem}[1]{}
\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}{}{}%
}%
{%
```

```
  \GlossariesWarning{overriding ‘theglossary’ environment}%
  \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}}
```

`\glossaryentryfield` `\glossaryentryfield{`⟨*label*⟩`}{`⟨*name*⟩`}{`⟨*description*⟩`}{`⟨*symbol*⟩`}{`⟨*page-list*⟩`}`

This command governs how each entry row should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore ⟨*symbol*⟩.

```
\newcommand*{\glossaryentryfield}[5]{%
\noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

`\glossaryentryfield` `\glossarysubentryfield{`⟨*level*⟩`}{`⟨*label*⟩`}{`⟨*name*⟩`}{`⟨*description*⟩`}{`⟨*symbol*⟩`}{`⟨*page-list*⟩`}`

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore ⟨*symbol*⟩. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```
\newcommand*{\glossarysubentryfield}[6]{%
\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.

\glsgetgrouptitle

```
\newcommand*{\glsgetgrouptitle}[1]{%
  \ifcsundef{#1groupname}{#1}{\csname #1groupname\endcsname}%
}
```

`\glsgetgrouplabel{`⟨*title*⟩`}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

\glsgetgrouplabel

```
\newcommand*{\glsgetgrouplabel}[1]{%
\ifthenelse{\equals{#1}{\glssymbolsgroupname}}{glssymbols}{%
\ifthenelse{\equals{#1}{\glsnumbersgroupname}}{glsnumbers}{#1}}}
```

The command `\setentrycounter` sets the entry's associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`\setentrycounter`

```
\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 `\glossarystyle{`⟨*style*⟩`}`.

`\glossarystyle`

```
\newcommand*{\glossarystyle}[1]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
  }%
  {%
    \csname @glsstyle@#1\endcsname
  }%
}
```

`\newglossarystyle` New glossary styles can be defined using:

`\newglossarystyle{`⟨*name*⟩`}{`⟨*definition*⟩`}`

The ⟨*definition*⟩ argument should redefine theglossary, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see subsection 1.18 for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```
\newcommand{\newglossarystyle}[2]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
  }%
  {%
    \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
  }%
}
```

`\renewglossarystyle` 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}}}
```

1.16 Acronyms

If the acronym package option is used, a new glossary called `acronym` is created

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

and `\acronymtype` is set to the name of this new glossary.

```
  \renewcommand*{\acronymtype}{acronym}
\fi
```

`\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, all it does is call `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be acronym if the package option acronym has been used, otherwise it will be the default glossary. Since \newacronym merely calls \newglossaryentry, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine \newacronym as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like \SetDefaultAcronymStyle.

\newacronym

```
  \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}{\glspluralsuffix}
```

The following are defined for compatibility with version 2.07 and earlier.

\glsshortkey

```
\newcommand*{\glsshortkey}{short}
```

\glsshortpluralkey

```
\newcommand*{\glsshortpluralkey}{shortplural}
```

\glslongkey

```
\newcommand*{\glslongkey}{long}
```

\glslongpluralkey

```
\newcommand*{\glslongpluralkey}{longplural}
```

\acrfull Full form of the acronym.

```
\newrobustcmd*{\acrfull}{%
  \@ifstar\s@acrfull\ns@acrfull
}

\newcommand*\s@acrfull[2][]{%
  \new@ifnextchar[{\@acrfull{hyper=false,#1}{#2}}%
                  {\@acrfull{hyper=false,#1}{#2}[]}%
}
\newcommand*\ns@acrfull[2][]{%
  \new@ifnextchar[{\@acrfull{#1}{#2}}%
                  {\@acrfull{#1}{#2}[]}%
}
```

Low-level macro:

```
\def\@acrfull#1#2[#3]{%
  \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
}
```

\acrlinkfullformat 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\space(#2)}
```

Default format for full acronym

\Acrfull

```
\newrobustcmd*{\Acrfull}{%
  \@ifstar\s@Acrfull\ns@Acrfull
}

\newcommand*\s@Acrfull[2][]{%
  \new@ifnextchar[{\@Acrfull{hyper=false,#1}{#2}}%
                  {\@Acrfull{hyper=false,#1}{#2}[]}%
}
\newcommand*\ns@Acrfull[2][]{%
  \new@ifnextchar[{\@Acrfull{#1}{#2}}%
                  {\@Acrfull{#1}{#2}[]}%
}
```

Low-level macro:

```
\def\@Acrfull#1#2[#3]{%
  \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
}
```

\ACRfull

```
\newrobustcmd*{\ACRfull}{%
  \@ifstar\s@ACRfull\ns@ACRfull
}

\newcommand*\s@ACRfull[2][]{%
  \new@ifnextchar[{\@ACRfull{hyper=false,#1}{#2}}%
                  {\@ACRfull{hyper=false,#1}{#2}[]}%
}
\newcommand*\ns@ACRfull[2][]{%
  \new@ifnextchar[{\@ACRfull{#1}{#2}}%
                  {\@ACRfull{#1}{#2}[]}%
}
```

Low-level macro:

```
\def\@ACRfull#1#2[#3]{%
  \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
}
```

Plural:

\acrfullpl

```
\newrobustcmd*{\acrfullpl}{%
  \@ifstar\s@acrfullpl\ns@acrfullpl
}

\newcommand*\s@acrfullpl[2][]{%
  \new@ifnextchar[{\@acrfullpl{hyper=false,#1}{#2}}%
                  {\@acrfullpl{hyper=false,#1}{#2}[]}%
}
\newcommand*\ns@acrfullpl[2][]{%
  \new@ifnextchar[{\@acrfullpl{#1}{#2}}%
                  {\@acrfullpl{#1}{#2}[]}%
}
```

Low-level macro:

```
\def\@acrfullpl#1#2[#3]{%
  \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
}
```

\Acrfullpl

```
\newrobustcmd*{\Acrfullpl}{%
  \@ifstar\s@Acrfullpl\ns@Acrfullpl
}

\newcommand*\s@Acrfullpl[2][]{%
  \new@ifnextchar[{\@Acrfullpl{hyper=false,#1}{#2}}%
                  {\@Acrfullpl{hyper=false,#1}{#2}[]}%
}
\newcommand*\ns@Acrfullpl[2][]{%
  \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%
                  {\@Acrfullpl{#1}{#2}[]}%
}
```

Low-level macro:

```
\def\@Acrfullpl#1#2[#3]{%
  \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
}
```

\ACRfullpl

```
\newrobustcmd*{\ACRfullpl}{%
  \@ifstar\s@ACRfullpl\ns@ACRfullpl
}
```

```
\newcommand*\s@ACRfullpl[2][]{%
  \new@ifnextchar[{\@ACRfullpl{hyper=false,#1}{#2}}%
                  {\@ACRfullpl{hyper=false,#1}{#2}[]}%
}
\newcommand*\ns@ACRfullpl[2][]{%
  \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
                  {\@ACRfullpl{#1}{#2}[]}%
}
```

Low-level macro:

```
\def\@ACRfullpl#1#2[#3]{%
  \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
}
```

1.17 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

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

`\firstacronymfont` 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}{}
```

`AcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
\newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
  \defglsdisplay[#1]{##1##4}%
  \defglsdisplayfirst[#1]{##1##4}%
}
```

`efaultNewAcronymDef` 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={\the\glsshorttok\noexpand\acrpluralsuffix},%
      firstplural={\acrfullformat{\noexpand\@glo@longpl}%
                                 {\noexpand\@glo@shortpl}},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\the\glslongtok},%
      descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
```

Remaining options specified by the user:

```
      \the\glskeylisttok
    }%
  }%
  \@do@newglossaryentry
}
```

`DefaultAcronymStyle` 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}{\glspluralsuffix}%
}
```

\acrfootnote Used by the footnote acronym styles.

```
\newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

\acrlinkfootnote

```
\newcommand*{\acrlinkfootnote}[3]{%
  \footnote{\glslink[#1]{#2}{#3}}%
}
```

\acrnolinkfootnote

```
\newcommand*{\acrnolinkfootnote}[3]{%
  \footnote{#3}%
}
```

AcronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
\newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
  \defglsdisplayfirst[#1]{%
    \firstacronymfont{##1}##4%
    \expandafter\protect\expandafter\acrfootnote\expandafter
       {\@gls@link@opts}{\@gls@link@label}{##3}%
  }%
  \defglsdisplay[#1]{\acronymfont{##1}##4}%
}
```

otnoteNewAcronymDef

```
\newcommand*{\DescriptionFootnoteNewAcronymDef}{%
    \edef\@do@newglossaryentry{%
      \noexpand\newglossaryentry{\the\glslabeltok}%
      {%
        type=\acronymtype,%
        name={\noexpand\acronymfont{\the\glsshorttok}},%
        sort={\the\glsshorttok},%
        text={\the\glsshorttok},%
        plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
        short={\the\glsshorttok},%
        shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
        long={\the\glslongtok},%
        longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
        symbol={\the\glslongtok},%
        symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
        \the\glskeylisttok
      }%
    }%
    \@do@newglossaryentry
}
```

ootnoteAcronymStyle 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}{%
      \textup{\glspluralsuffix}}%
  \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
}%
```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```
\newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
  \defglsdisplay[#1]{##1##4}%
  \defglsdisplayfirst[#1]{##1##4}%
}
```

ionDUANewAcronymDef

```
\newcommand*{\DescriptionDUANewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\the\glslongtok},%
      sort={\the\glslongtok},
      text={\the\glslongtok},%
      plural={\the\glslongtok\noexpand\acrpluralsuffix},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      symbol={\the\glsshorttok},%
      symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      \the\glskeylisttok
    }%
  }%
  \@do@newglossaryentry
}
```

tionDUAAcronymStyle Description, don't use acronym and no footnote. Note that the short form is stored in the symbol key, so if the short form needs to be displayed in the glossary, use a style the displays the symbol.

```
\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}%
  }%
}%
```

`AcronymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```
\newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
  \defglsdisplayfirst[#1]{%
    ##1##4 (\firstacronymfont{##3})}%
  \defglsdisplay[#1]{\acronymfont{##1}##4}%
}
```

`iptionNewAcronymDef`

```
\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={\the\glslongtok\noexpand\acrpluralsuffix},%
      text={\the\glsshorttok},%
      plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      symbol={\noexpand\@glo@text},%
      symbolplural={\noexpand\@glo@plural},%
      \the\glskeylisttok}%
  }%
  \@do@newglossaryentry
}
```

riptionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acrnameformat to allow the user to override the way the name is displayed in the list of acronyms.

```
\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}{%
      \textup{\glspluralsuffix}}%
  \else
    \ifglsacrsmaller
      \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
    \fi
  \fi
}%
```

AcronymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```
\newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
  \defglsdisplayfirst[#1]{%
    \firstacronymfont{##1}##4%
    \expandafter\protect\expandafter\acrfootnote\expandafter
      {\@gls@link@opts}{\@gls@link@label}{##2}%
  }%
  \defglsdisplay[#1]{\acronymfont{##1}##4}%
}
```

otnoteNewAcronymDef

```
\newcommand*{\FootnoteNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      sort={\the\glsshorttok},%
      text={\the\glsshorttok},%
      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
    }%
  }%
  \@do@newglossaryentry
}
```

ootnoteAcronymStyle

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}{%
        \textup{\glspluralsuffix}}%
  \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
}%
```

AcronymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
\newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
  \defglsdisplayfirst[#1]{##1##4 (\firstacronymfont{##3})}
  \defglsdisplay[#1]{\acronymfont{##1}##4}%
}
```

\SmallNewAcronymDef

```
\newcommand*{\SmallNewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\noexpand\acronymfont{\the\glsshorttok}},%
      sort={\the\glsshorttok},%
      text={\noexpand\@glo@symbol},%
```

Default to the short plural.

```
      plural={\noexpand\@glo@shortpl},%
      first={\the\glslongtok},%
```

Default to the long plural.

```
      firstplural={\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\@glo@firstplural},%
      symbol={\the\glsshorttok},%
```

Default to the short plural.

```
      symbolplural={\noexpand\@glo@shortpl},%
      \the\glskeylisttok
```

```
    }%
  }%
  \@do@newglossaryentry
}
```

etSmallAcronymStyle 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}{%
       \textup{\glspluralsuffix}}%
  \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
}%
```

\SetDUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```
\newcommand*{\SetDUADisplayStyle}[1]{%
  \defglsdisplay[#1]{##1##4}%
  \defglsdisplayfirst[#1]{##1##4}%
}
```

\DUANewAcronymDef

```
\newcommand*{\DUANewAcronymDef}{%
  \edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\acronymtype,%
      name={\the\glsshorttok},%
      text={\the\glslongtok},%
      plural={\the\glslongtok\noexpand\acrpluralsuffix},%
      short={\the\glsshorttok},%
      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      long={\the\glslongtok},%
      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
      description={\the\glslongtok},%
      symbol={\the\glsshorttok},%
      symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
      \the\glskeylisttok
    }%
  }%
  \@do@newglossaryentry
}
```

\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.

tCustomDisplayStyle

Sets the acronym display style.

```
\newcommand*{\SetCustomDisplayStyle}[1]{%
  \defglsdisplay[#1]{##1##4}%
  \defglsdisplayfirst[#1]{##1##4}%
}
```

CustomAcronymFields

```
\newcommand*{\CustomAcronymFields}{%
  name={\the\glsshorttok},%
  description={\the\glslongtok},%
  first={\noexpand\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
  firstplural={\noexpand\acrfullformat
    {\the\glslongtok\noexpand\acrpluralsuffix}{\the\glsshorttok}}%
  text={\the\glsshorttok},%
  plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
}
```

CustomNewAcronymDef

```
\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}%
  }%
```

```
}
```

fineAcronymSynonyms

```
\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
```

1.18 Predefined Glossary Styles

The glossaries bundle comes with some predefined glossary styles. These need to be loaded now for the style option to use them.

First, the glossary hyper-navigation commands need to be loaded.

```
\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
  \glossarystyle{\@glossary@default@style}
\fi
```

1.19 Debugging Commands

\showgloparent

```
\showgloparent{⟨label⟩}
```

```
\newcommand*{\showgloparent}[1]{%
  \expandafter\show\csname glo@#1@parent\endcsname
}
```

\showglolevel

```
\showglolevel{⟨label⟩}
```

```
\newcommand*{\showglolevel}[1]{%
  \expandafter\show\csname glo@#1@level\endcsname
}
```

\showglotext

```
\showglotext{⟨label⟩}
```

```
\newcommand*{\showglotext}[1]{%
  \expandafter\show\csname glo@#1@text\endcsname
}
```

\showgloplural

```
\showgloplural{⟨label⟩}
```

```
\newcommand*{\showgloplural}[1]{%
  \expandafter\show\csname glo@#1@plural\endcsname
}
```

\showglofirst

```
\showglofirst{⟨label⟩}
```

```
\newcommand*{\showglofirst}[1]{%
  \expandafter\show\csname glo@#1@first\endcsname
}
```

\showglofirstpl

```
\showglofirstpl{⟨label⟩}
```

```
\newcommand*{\showglofirstpl}[1]{%
  \expandafter\show\csname glo@#1@firstpl\endcsname
}
```

\showglotype

```
\showglotype{⟨label⟩}
```

```
\newcommand*{\showglotype}[1]{%
  \expandafter\show\csname glo@#1@type\endcsname
}
```

\showglocounter

```
\showglocounter{⟨label⟩}
```

```
\newcommand*{\showglocounter}[1]{%
  \expandafter\show\csname glo@#1@counter\endcsname
}
```

\showglouseri

```
\showglouseri{⟨label⟩}
```

```
\newcommand*{\showglouseri}[1]{%
  \expandafter\show\csname glo@#1@useri\endcsname
}
```

\showglouserii

```
\showglouserii{⟨label⟩}
```

```
\newcommand*{\showglouserii}[1]{%
  \expandafter\show\csname glo@#1@userii\endcsname
}
```

\showglouseriii

```
\showglouseriii{⟨label⟩}
```

```
\newcommand*{\showglouseriii}[1]{%
  \expandafter\show\csname glo@#1@useriii\endcsname
}
```

\showglouseriv

```
\showglouseriv{⟨label⟩}
```

```
\newcommand*{\showglouseriv}[1]{%
  \expandafter\show\csname glo@#1@useriv\endcsname
}
```

\showglouserv

```
\showglouserv{⟨label⟩}
```

```
\newcommand*{\showglouserv}[1]{%
  \expandafter\show\csname glo@#1@userv\endcsname
}
```

\showglouservi

```
\showglouservi{⟨label⟩}
```

```
\newcommand*{\showglouservi}[1]{%
  \expandafter\show\csname glo@#1@uservi\endcsname
}
```

\showgloname

```
\showgloname{⟨label⟩}
```

```
\newcommand*{\showgloname}[1]{%
  \expandafter\show\csname glo@#1@name\endcsname
}
```

\showglodesc

```
\showglodesc{⟨label⟩}
```

```
\newcommand*{\showglodesc}[1]{%
  \expandafter\show\csname glo@#1@desc\endcsname
}
```

\showglodescplural

```
\showglodescplural{⟨label⟩}
```

```
\newcommand*{\showglodescplural}[1]{%
  \expandafter\show\csname glo@#1@descplural\endcsname
}
```

\showglosort

```
\showglosort{⟨label⟩}
```

```
\newcommand*{\showglosort}[1]{%
  \expandafter\show\csname glo@#1@sort\endcsname
}
```

\showglosymbol

```
\showglosymbol{⟨label⟩}
```

```
\newcommand*{\showglosymbol}[1]{%
  \expandafter\show\csname glo@#1@symbol\endcsname
}
```

\showglosymbolplural

```
\showglosymbolplural{⟨label⟩}
```

```
\newcommand*{\showglosymbolplural}[1]{%
  \expandafter\show\csname glo@#1@symbolplural\endcsname
}
```

\showgloindex

```
\showgloindex{⟨label⟩}
```

```
\newcommand*{\showgloindex}[1]{%
  \expandafter\show\csname glo@#1@index\endcsname
}
```

\showgloflag

```
\showgloflag{⟨label⟩}
```

```
\newcommand*{\showgloflag}[1]{%
  \expandafter\show\csname ifglo@#1@flag\endcsname
}
```

\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
}
```

\showglossarytitle

```
\showglossarytitle{⟨glossary-label⟩}
```

Show the title for the given glossary.

```
\newcommand*{\showglossarytitle}[1]{%
  \expandafter\show\csname @glotype@#1@title\endcsname
}
```

\showglossarycounter

```
\showglossarycounter{⟨glossary-label⟩}
```

Show the counter for the given glossary.

```
\newcommand*{\showglossarycounter}[1]{%
  \expandafter\show\csname @glotype@#1@counter\endcsname
}
```

\showglossaryentries

```
\showglossaryentries{⟨glossary-label⟩}
```

Show the list of entry labels for the given glossary.

```
\newcommand*{\showglossaryentries}[1]{%
  \expandafter\show\csname glolist@#1\endcsname
}
```

1.20 Compatibility with version 2.07 and below

In order to fix some bugs in v3.0, it was necessary to change the way information is written to the `glo` file, which also meant a change in the format of the Xindy style file. The compatibility option is meant for documents that use a customised Xindy style file with `\noist`. With the compatibility option, hopefully `xindy` will still be able to process the old document, but the bugs will remain. The issues in versions 2.07 and below:

- With xindy, the counter used by the entry was hard-coded into the Xindy style file. This meant that you couldn't use the counter to swap counters.
- With both xindy and makeindex, if used with hyperref and `\theH`⟨*counter*⟩ was different to `\thecounter`, the link in the location number would be undefined.

```
\csname ifglscompatible-2.07\endcsname
  \RequirePackage{glossaries-compatible-207}
\fi
```

2 Mfirstuc Documented Code

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{mfirstuc}[2012/05/21 v1.06 (NLCT)]
```

Requires etoolbox:

```
\RequirePackage{etoolbox}
```

`\makefirstuc` Syntax:

`\makefirstuc{`⟨*text*⟩`}`

Makes the first letter uppercase, but will skip initial control sequences if they are followed by a group and make the first thing in the group uppercase, unless the group is empty. Thus `\makefirstuc{abc}` will produce: Abc, `\makefirstuc{\ae bc}` will produce: Æbc, but `\makefirstuc{\emph{abc}}` will produce *Abc*. This is required by `\Gls` and `\Glspl`.

```
\newif\if@glscs
\newtoks\@glsmfirst
\newtoks\@glsmrest
\def\makefirstuc#1{%
  \def\gls@argi{#1}%
  \ifx\gls@argi\@empty
```

If the argument is empty, do nothing.

```
  \else
    \def\@gls@tmp{\ #1}%
    \@onelevel@sanitize\@gls@tmp
    \expandafter\@gls@checkcs\@gls@tmp\relax\relax
    \if@glscs
      \@gls@getbody #1{}\@nil
      \ifx\@gls@rest\@empty
        \glsmakefirstuc{#1}%
      \else
        \expandafter\@gls@split\@gls@rest\@nil
        \ifx\@gls@first\@empty
           \glsmakefirstuc{#1}%
        \else
           \expandafter\@glsmfirst\expandafter{\@gls@first}%
      \expandafter\@glsmrest\expandafter{\@gls@rest}%
      \edef\@gls@domfirstuc{\noexpand\@gls@body
        {\noexpand\glsmakefirstuc\the\@glsmfirst}%
        \the\@glsmrest}%
      \@gls@domfirstuc
    \fi
   \fi
  \else
   \glsmakefirstuc{#1}%
  \fi
 \fi
}
```

Put first argument in `\@gls@first` and second argument in `\@gls@rest`:

```
\def\@gls@split#1#2\@nil{%
 \def\@gls@first{#1}\def\@gls@rest{#2}%
}
```

```
\def\@gls@checkcs#1 #2#3\relax{%
 \def\@gls@argi{#1}\def\@gls@argii{#2}%
 \ifx\@gls@argi\@gls@argii
  \@glscstrue
 \else
  \@glscsfalse
 \fi
}
```

Make first thing upper case:

```
\def\@gls@makefirstuc#1{\MakeUppercase #1}
```

`\glsmakefirstuc` Provide a user command to make it easier to customise.

```
\newcommand*{\glsmakefirstuc}[1]{\@gls@makefirstuc{#1}}
```

Get the first grouped argument and stores in `\@gls@body`.

```
\def\@gls@getbody#1#{\def\@gls@body{#1}\@gls@gobbletonil}
```

Scoup up everything to `\@nil` and store in `\@gls@rest`:

```
\def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}
```

`\xmakefirstuc` Expand argument once before applying `\makefirstuc` (added v1.01).

```
\newcommand*{\xmakefirstuc}[1]{%
\expandafter\makefirstuc\expandafter{#1}}
```

`\capitalisewords` Capitalise each word in the argument. Words are considered to be separated by plain spaces (i.e. non-breakable spaces won't be considered a word break).

```
\newcommand*{\capitalisewords}[1]{%
 \def\gls@add@space{}%
 \mfu@capitalisewords#1 \@nil\mfu@endcap
  %\gls@add@space\makefirstuc{##1}\def\gls@add@space{ }%
}
```

```
\def\mfu@capitalisewords#1 #2\mfu@endcap{%
  \def\mfu@cap@first{#1}%
  \def\mfu@cap@second{#2}%
  \gls@add@space
  \makefirstuc{#1}%
  \def\gls@add@space{ }%
  \ifx\mfu@cap@second\@nnil
    \let\next@mfu@cap\mfu@noop
  \else
    \let\next@mfu@cap\mfu@capitalisewords
  \fi
  \next@mfu@cap#2\mfu@endcap
}
\def\mfu@noop#1\mfu@endcap{}
```

\xcapitalisewords Short-cut command:

```
\newcommand*{\xcapitalisewords}[1]{%
  \expandafter\capitalisewords\expandafter{#1}%
}
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
\ProvidesPackage{glossary-hypernav}[2007/07/04 v1.01 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see subsection 1.15.) \printglossary (and \printglossaries) set \@glo@type to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

\glsnavhyperlink[⟨*type*⟩]{⟨*label*⟩}{⟨*text*⟩}

This command makes ⟨*text*⟩ a hyperlink to the glossary group whose label is given by ⟨*label*⟩ for the glossary given by ⟨*type*⟩.

\glsnavhyperlink

```
\newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
  \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
  \@glslink{glsn:#1@#2}{#3}}
```

\glsnavhypertarget[⟨*type*⟩]{⟨*label*⟩}{⟨*text*⟩}

This command makes ⟨*text*⟩ a hypertarget for the glossary group whose label is given by ⟨*label*⟩ in the glossary given by ⟨*type*⟩. If ⟨*type*⟩ is omitted, \@glo@type is used which is set by \printglossary to the current glossary label.

\glsnavhypertarget

```
\newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
```

Add this group to the aux file for re-run check.

```
  \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
  \@glstarget{glsn:#1@#2}{#3}%
```

Check list of know 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
}
```

gls@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. Note that this command will produce a link to all 28 groups, but some groups may not be defined if there are groups that do not contain any terms, in which case you will get an undefined hyperlink warning. Now for the whole navigation bit:

\glsnavigation

```
\newcommand*{\glsnavigation}{%
\def\@gls@between{}%
\@ifundefined{@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
  \glsnavhyperlink{\@gls@tmp}{\glsgetgrouptitle{\@gls@tmp}}%
  \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}[2012/09/21 v3.03 (NLCT)]
```

inline Define the inline style.

```
\newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossaryentryfield`)

```
\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{\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}%
}%
```

Nothing special between groups:

```
  \renewcommand*{\glsgroupskip}{}%
}
```

lsinlinedopostchild

```
\newcommand*{\glsinlinedopostchild}{%
    \gls@inlinepostchild
    \def\gls@inlinepostchild{}%
}
```

\glsinlineseparator Separator to use between entries.

```
\newcommand*{\glsinlineseparator}{;\space}
```

sinlinesubseparator Separator to use between sub-entries.

```
\newcommand*{\glsinlinesubseparator}{,\space}
```

arentchildseparator Separator to use between parent and children.

```
\newcommand*{\glsinlineparentchildseparator}{:\space}
```

\glsinlinepostchild Hook to use between child and next entry

```
\newcommand*{\glsinlinepostchild}{}
```

\glspostinline Terminator for inline glossary.

```
\newcommand*{\glspostinline}{\glspostdescription\space}
```

glsinlinenameformat Formats the name of the entry (first argument label, second argument name):

```
\newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

glsinlinedescformat Formats the entry's description, symbol and location list:

```
\newcommand*{\glsinlinedescformat}[3]{\space#1}
```

lineemptydescformat Formats the entry's symbol and location list when the description is empty:

```
\newcommand*{\glsinlineemptydescformat}[2]{}
```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```
\newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

inlinesubdescformat 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}[2012/11/11 v3.04 (NLCT)]
```

list The list glossary style uses the description environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```
\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*{\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.}%
%    \end{macrocode}
% Add vertical space between groups:
%\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
%    \begin{macrocode}
  \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:

```
  \glossarystyle{list}%
```

Each group has a heading:

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

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

```
\newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
  \glossarystyle{list}%
```

Add navigation links at the start of the environment:

```
  \renewcommand*{\glossaryheader}{%
    \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

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

`altlist` The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
\newglossarystyle{altlist}{%
```

Base it on the list style:

```
  \glossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
  \renewcommand*{\glossaryentryfield}[5]{%
    \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
```

Version 3.04 changed `\newline` to the following paragraph break stuff (thanks to Daniel Gebhardt for supplying the fix) to prevent a page break occurring at this point.

```
       \mbox{}\par\nobreak\@afterheading
       ##3\glspostdescription\space ##5}%
```

Sub-entries start a new paragraph:

```
  \renewcommand{\glossarysubentryfield}[6]{%
    \par
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
}
```

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

```
\newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
  \glossarystyle{altlist}%
```

Each group has a heading:

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

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

```
\newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
  \glossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
\renewcommand*{\glossaryheader}{%
  \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

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

listdotted The listdotted glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
\newglossarystyle{listdotted}{%
```

Base it on the list style:

```
  \glossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
  \renewcommand*{\glossaryentryfield}[5]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
      \glsentryitem{##1}\glstarget{##1}{##2}%
      \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
```

Sub entries have the same format as main entries:

```
  \renewcommand*{\glossarysubentryfield}[6]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
    \glssubentryitem{##2}%
    \glstarget{##2}{##3}%
    \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
}
```

\glslistdottedwidth

```
\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:

```
  \glossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
  \renewcommand*{\glossaryentryfield}[5]{%
    \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
}
```

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}[2012/09/21 v3.03 (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}
}{}
```

`\glspagelistwidth` 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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
```

Sub entries displayed on the following row without the name:

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \\\fi}%
}
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
  \glossarystyle{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:

```
  \glossarystyle{long}%
```

Set the table's header:

```
  \renewcommand*{\glossaryheader}{%
    \bfseries \entryname & \bfseries \descriptionname\\\endhead}%
}
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
\newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
  \glossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
  \renewcommand*{\glossaryheader}{%
    \hline\bfseries \entryname & \bfseries \descriptionname\\\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*{\glossaryentryfield}[5]{%
  \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
\renewcommand*{\glossarysubentryfield}[6]{%
   &
   \glssubentryitem{##2}%
   \glstarget{##2}{\strut}##4 & ##6\\}%
```

Blank row between groups:

```
\renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\\\fi}%
}
```

`long3colborder` The long3colborder style is like the long3col style but with a border:

```
\newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
\glossarystyle{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:

```
\glossarystyle{long3col}%
```

Set the table's header:

```
\renewcommand*{\glossaryheader}{%
  \bfseries\entryname&\bfseries\descriptionname&
  \bfseries\pagelistname\\\endhead}%
}
```

`ong3colheaderborder` The long3colheaderborder style is like the above but with a border

```
\newglossarystyle{long3colheaderborder}{%
```

Base it on the glostylelong3colborder style:

```
\glossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
  \renewcommand*{\glossaryheader}{%
    \hline
    \bfseries\entryname&\bfseries\descriptionname&
    \bfseries\pagelistname\\\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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & &\\\fi}%
}
```

long4colheader The long4colheader style is like long4col but with a header row.

```
\newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
  \glossarystyle{long4col}%
```

Table has a header:

```
  \renewcommand*{\glossaryheader}{%
    \bfseries\entryname&\bfseries\descriptionname&
    \bfseries \symbolname&
    \bfseries\pagelistname\\\endhead}%
}
```

`long4colborder` The long4colborder style is like long4col but with a border.

```
\newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
  \glossarystyle{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}%
}
```

`ong4colheaderborder` The long4colheaderborder style is like the above but with a border.

```
\newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```
  \glossarystyle{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\\\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:

```
   \glossarystyle{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}}%
}
```

`altlong4colheader` The altlong4colheader style is like altlong4col but with a header row.

```
\newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
  \glossarystyle{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}}%
}
```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
\newglossarystyle{altlong4colborder}{%
```

Base it on the glostylelong4colborder style:

```
  \glossarystyle{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}}%
}
```

ong4colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
\newglossarystyle{altlong4colheaderborder}{%
```

Base it on the glostylelong4colheaderborder style:

```
 \glossarystyle{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 (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}[2012/09/21 v3.03 (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}
}{}
```

\glspagelistwidth 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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
    \tabularnewline}%
```

Sub entries displayed on the following row without the name:

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
    \tabularnewline}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
}
```

longraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{longraggedborder}{%
```

Base it on the glostylelongragged style:

```
  \glossarystyle{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}%
}
```

longraggedheader The longraggedheader style is like the longragged style but with a header:

```
\newglossarystyle{longraggedheader}{%
```

Base it on the glostylelongragged style:

```
  \glossarystyle{longragged}%
```

Set the table's header:

```
  \renewcommand*{\glossaryheader}{%
    \bfseries \entryname & \bfseries \descriptionname
    \tabularnewline\endhead}%
}
```

graggedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
\newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```
  \glossarystyle{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*{\glossaryentryfield}[5]{%
  \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
\renewcommand*{\glossarysubentryfield}[6]{%
   &
   \glssubentryitem{##2}%
   \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
\renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%
}
```

ongragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
\newglossarystyle{longragged3colborder}{%
```

Base it on the glostylelongragged3col style:

```
\glossarystyle{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}%
}
```

ongragged3colheader The longragged3colheader style is like longragged3col but with a header row:

```
\newglossarystyle{longragged3colheader}{%
```

Base it on the glostylelongragged3col style:

```
\glossarystyle{longragged3col}%
```

Set the table's header:

```
\renewcommand*{\glossaryheader}{%
  \bfseries\entryname&\bfseries\descriptionname&
  \bfseries\pagelistname\tabularnewline\endhead}%
}
```

ged3colheaderborder The longragged3colheaderborder style is like the above but with a border

```
\newglossarystyle{longragged3colheaderborder}{%
```

Base it on the glostylelongragged3colborder style:

```
\glossarystyle{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}%
}
```

altlongragged4col

The altlongragged4col style is like the altlong4col style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
\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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & &\tabularnewline\fi}%
}
```

ongragged4colheader

The altlongragged4colheader style is like altlongragged4col but with a header row.

```
\newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
  \glossarystyle{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}%
}
```

ongragged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
\newglossarystyle{altlongragged4colborder}{%
```

Base it on the glostylealtlongragged4col style:

```
 \glossarystyle{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}%
}
```

ged4colheaderborder The altlongragged4colheaderborder style is like the above but with a header as well as a border.

```
\newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the glostylealtlongragged4col style:

```
\glossarystyle{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}%
}
```

3.6 Glossary Styles using multicol (glossary-mcols.sty)

The style file defines glossary styles that use the multicol package. These use the tree-like glossary styles in a multicol environment.

```
\ProvidesPackage{glossary-mcols}[2012/05/21 v1.0 (NLCT)]
```

Required packages:

```
\RequirePackage{multicol}
\RequirePackage{glossary-tree}
```

`\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}{%
  \glossarystyle{index}%
  \renewenvironment{theglossary}%
    {%
     \begin{multicols}{2}
     \setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%
     \let\item\@idxitem}%
    {\end{multicols}}%
}
```

`mcolindexgroup` As mcolindex but has headings:

```
\newglossarystyle{mcolindexgroup}{%
 \glossarystyle{mcolindex}%
 \renewcommand*{\glsgroupheading}[1]{%
    \item\textbf{\glsgetgrouptitle{##1}}\indexspace}%
}
```

`mcolindexhypergroup` The mcolindexhypergroup style is like the mcolindexgroup style but has hyper navigation.

```
\newglossarystyle{mcolindexhypergroup}{%
```

Base it on the glostylemcolindex style:

```
  \glossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
  \renewcommand*{\glossaryheader}{%
    \item\textbf{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
  \renewcommand*{\glsgroupheading}[1]{%
    \item\textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
    \indexspace}%
}
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
\newglossarystyle{mcoltree}{%
  \glossarystyle{tree}%
  \renewenvironment{theglossary}%
  {%
     \begin{multicols}{2}
     \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:

```
  \glossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

mcoltreehypergroup 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:

```
  \glossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
\newglossarystyle{mcoltreenoname}{%
  \glossarystyle{treenoname}%
  \renewenvironment{theglossary}%
  {%
     \begin{multicols}{2}
     \setlength{\parindent}{0pt}%
```

```
    \setlength{\parskip}{0pt plus 0.3pt}%
  }%
  {\end{multicols}}%
}
```

mcoltreenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```
\newglossarystyle{mcoltreenonamegroup}{%
```

Base it on the glostylemcoltreenoname style:

```
  \glossarystyle{mcoltreenoname}%
```

Give each group a heading:

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

reenonamehypergroup 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:

```
  \glossarystyle{mcoltreenoname}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

mcolalttree Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```
\newglossarystyle{mcolalttree}{%
  \glossarystyle{alttree}%
  \renewenvironment{theglossary}%
  {%
    \begin{multicols}{2}%
    \def\@gls@prevlevel{-1}%
    \mbox{}\par
  }%
  {\par\end{multicols}}%
}
```

mcolalttreegroup Like the mcolalttree style but the glossary groups have headings.

```
\newglossarystyle{mcolalttreegroup}{%
```

Base it on the glostylemcolalttree style:

```
\glossarystyle{mcolalttree}%
```

Give each group a heading.

```
\renewcommand{\glsgroupheading}[1]{\par
  \def\@gls@prevlevel{-1}%
  \hangindent0pt\relax
  \parindent0pt\relax
  \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

olalttreehypergroup 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:

```
\glossarystyle{mcolalttree}%
```

Put the navigation links in the header

```
\renewcommand*{\glossaryheader}{%
  \par
  \def\@gls@prevlevel{-1}%
  \hangindent0pt\relax
  \parindent0pt\relax
  \textbf{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
\renewcommand*{\glsgroupheading}[1]{%
  \par
  \def\@gls@prevlevel{-1}%
  \hangindent0pt\relax
  \parindent0pt\relax
  \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
  \indexspace}}
```

3.7 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
\ProvidesPackage{glossary-super}[2012/09/21 v3.03 (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}
}{}
```

\glspagelistwidth 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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
```

Sub entries put in a row (no name, description and page list in second column):

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \\\fi}%
}
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{superborder}{%
```

Base it on the glostylesuper style:

```
  \glossarystyle{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:

```
  \glossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
\renewenvironment{theglossary}%
  {\tablehead{\bfseries \entryname & \bfseries \descriptionname\\}%
   \tabletail{}%
   \begin{supertabular}{lp{\glsdescwidth}}}%
  {\end{supertabular}}%
}
```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
\newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
  \glossarystyle{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\\\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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##6\\}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\\\fi}%
}
```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
\newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
  \glossarystyle{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:

```
  \glossarystyle{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\\}\tabletail{}%
     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
    {\end{supertabular}}%
}
```

per3colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
\newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
  \glossarystyle{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\\\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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & &\\\fi}%
}
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
\newglossarystyle{super4colheader}{%
```

Base it on the glostylesuper4col style:

```
  \glossarystyle{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\\}%
     \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:

```
  \glossarystyle{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}}%
}
```

per4colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```
\newglossarystyle{super4colheaderborder}{%
```

Base it on the glostylesuper4col style:

```
  \glossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
        \bfseries\symbolname &
        \bfseries\pagelistname\\\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:

```
  \glossarystyle{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}}%
}
```

altsuper4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```
\newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylesuper4colheader style:

```
  \glossarystyle{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\\}\tabletail{}%
     \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
    {\end{supertabular}}%
}
```

altsuper4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```
\newglossarystyle{altsuper4colborder}{%
```

Base it on the glostylesuper4colborder style:

```
  \glossarystyle{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}}%
}
```

per4colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```
\newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the glostylesuper4colheaderborder style:

```
  \glossarystyle{super4colheaderborder}%
```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
  \renewenvironment{theglossary}%
    {\tablehead{\hline
       \bfseries\entryname &
       \bfseries\descriptionname &
       \bfseries\symbolname &
       \bfseries\pagelistname\\\hline}%
     \tabletail{\hline}%
     \begin{supertabular}%
       {|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
    {\end{supertabular}}%
}
```

3.8 Glossary Styles using supertabular environment (glossary-superragged package)

The glossary styles defined in the package use the supertabular environment. These styles are like those provided by the package, except that the multiline columns have ragged right justification.

```
\ProvidesPackage{glossary-superragged}[2012/09/21 v3.03 (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}
}{}
```

\glspagelistwidth 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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
      \tabularnewline}%
```

Sub entries put in a row (no name, description and page list in second column):

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
     \tabularnewline}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
}
```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{superraggedborder}{%
```

Base it on the glostylesuperragged style:

```
  \glossarystyle{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}}%
}
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
\newglossarystyle{superraggedheader}{%
```

Base it on the glostylesuperragged style:

```
  \glossarystyle{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}}%
}
```

rraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
\newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostylesuper style:

```
  \glossarystyle{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*{\glossaryentryfield}[5]{%
   \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
 \renewcommand*{\glossarysubentryfield}[6]{%
    &
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%
}
```

perragged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
\newglossarystyle{superragged3colborder}{%
```

Base it on the glostylesuperragged3col style:

```
 \glossarystyle{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}}%
}
```

perragged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
\newglossarystyle{superragged3colheader}{%
```

Base it on the glostylesuperragged3col style:

```
  \glossarystyle{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}}%
}
```

ght3colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
\newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
  \glossarystyle{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}}%
}
```

altsuperragged4col 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*{\glossaryentryfield}[5]{%
    \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
  \renewcommand*{\glossarysubentryfield}[6]{%
     &
     \glssubentryitem{##2}%
     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
```

Blank row between groups:

```
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & &\tabularnewline\fi}%
}
```

perragged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
\newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
  \glossarystyle{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}}%
}
```

perragged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
\newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
  \glossarystyle{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}}%
}
```

ged4colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
\newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
  \glossarystyle{altsuperragged4col}%
```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
  \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}}%
}
```

3.9 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}[2012/09/21 v3.03 (NLCT)]
```

index The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```
\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\@idxitem}%
  {}%
```

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*{\glossaryentryfield}[5]{%
\item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
\ifx\relax##4\relax
\else
  \space(##4)%
\fi
\space ##3\glspostdescription \space ##5}%
```

Sub entries: level 1 entries use `\subitem`, levels greater than 1 use `\subsubitem`. The level (##1) shouldn't be 0, as that's catered by `\glossaryentryfield`, but for completeness, if the level is 0, `\item` is used. The name is put in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
\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}%
```

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:

```
  \glossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

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

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```
\newglossarystyle{indexhypergroup}{%
```

Base it on the glostyleindex style:

```
  \glossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```
  \renewcommand*{\glossaryheader}{%
    \item\textbf{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

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

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{\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}%
```

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:

```
  \glossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\textbf{\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:

```
  \glossarystyle{tree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \textbf{\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{\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}%
```

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{\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}%
```

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:

```
  \glossarystyle{treenoname}%
```

Give each group a heading:

```
  \renewcommand{\glsgroupheading}[1]{\par
    \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

reenonamehypergroup 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:

```
  \glossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
  \renewcommand*{\glossaryheader}{%
    \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par\noindent
    \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}%
}
```

\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}{}
```

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{\glossaryentryfield}[5]{%
```

If the level hasn't changed, keep the same settings, otherwise change `\glstreeindent` accordingly.

```
    \ifnum\@gls@prevlevel=0\relax
    \else
```

Find out how big the indentation should be by measuring the widest entry.

```
        \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
```

Set the hangindent and paragraph indent.

```
       \hangindent\glstreeindent
       \parindent\glstreeindent
     \fi
```

Put the name to the left of the paragraph block.

```
     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
        \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
     \ifx\relax##4\relax
     \else
       (##4)\space
     \fi
```

Do the description followed by the description terminator and location list.

```
     ##3\glspostdescription \space ##5\par
```

Set the previous level to 0.

```
     \def\@gls@prevlevel{0}%
   }%
```

Redefine the way sub-entries are displayed.

```
   \renewcommand{\glossarysubentryfield}[6]{%
```

Increment and display the sub-entry counter if this is a level 1 entry and the sub-entry counter is in use.

```
     \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}{\textbf{\@glswidestname\space}}}{%
         \settowidth{\gls@tmplen}{\textbf{%
            \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}{\textbf{%
             \@glswidestname\space}}}{%
          \settowidth{\glstreeindent}{\textbf{%
             \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]{\makebox[\gls@tmplen][l]{%
      \textbf{\glstarget{##2}{##3}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
    \ifx##5\relax\relax
    \else
      (##5)\space
    \fi
```

Do the description followed by the description terminator and location list.

```
    ##4\glspostdescription\space ##6\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:

```
  \glossarystyle{alttree}%
```

Give each group a heading.

```
  \renewcommand{\glsgroupheading}[1]{\par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
}
```

alttreehypergroup 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:

```
  \glossarystyle{alttree}%
```

Put the navigation links in the header

```
  \renewcommand*{\glossaryheader}{%
    \par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \textbf{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
  \renewcommand*{\glsgroupheading}[1]{%
    \par
    \def\@gls@prevlevel{-1}%
    \hangindent0pt\relax
    \parindent0pt\relax
    \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
    \indexspace}}
```

4 glossaries-compatible-207

Provides compatibility with version 2.07 and below. This uses original glossaries xindy and makeindex formatting, so can be used with old documents that had customized style files, but hyperlinks may not work properly.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-compatible-207}[2011/04/02 v1.0 (NLCT)]
```

\GlsAddXdyAttribute 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
```

`\GlsAddXdyCounters`

```
\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}
```

`\GlsAddXdyLocation` 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
}
```

`\@set@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}
```

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 but will only be updated when `glossaries-accsupp.sty` is modified.

```
\ProvidesPackage{glossaries-accsupp}[2011/04/02 v3.0 (NLCT)
  Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
\ProcessOptions
```

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}%
}
```

firstpluralaccess 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}%
}
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

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

descriptionaccess The replacement text corresponding to the description key:

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

riptionpluralaccess 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}%
}
```

shortpluralaccess 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}%
}
```

longpluralaccess 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}}`.

`\@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]{%
  \csname glo@#1@access\endcsname
}
```

\glsentrytextaccess Get the value of the textaccess key for the entry with the given label:

```
\newcommand*{\glsentrytextaccess}[1]{%
  \csname glo@#1@textaccess\endcsname
}
```

glsentryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
\newcommand*{\glsentryfirstaccess}[1]{%
  \csname glo@#1@firstaccess\endcsname
}
```

lsentrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrypluralaccess}[1]{%
  \csname glo@#1@pluralaccess\endcsname
}
```

ryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentryfirstpluralaccess}[1]{%
  \csname glo@#1@firstpluralaccess\endcsname
}
```

lsentrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
\newcommand*{\glsentrysymbolaccess}[1]{%
  \csname glo@#1@symbolaccess\endcsname
}
```

ysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrysymbolpluralaccess}[1]{%
  \csname glo@#1@symbolpluralaccess\endcsname
}
```

\glsentrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
\newcommand*{\glsentrydescaccess}[1]{%
  \csname glo@#1@descaccess\endcsname
}
```

trydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrydescpluralaccess}[1]{%
  \csname glo@#1@descaccess\endcsname
}
```

glsentryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
\newcommand*{\glsentryshortaccess}[1]{%
  \csname glo@#1@shortaccess\endcsname
}
```

ryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentryshortpluralaccess}[1]{%
  \csname glo@#1@shortpluralaccess\endcsname
}
```

\glsentrylongaccess Get the value of the longaccess key for the entry with the given label:

```
\newcommand*{\glsentrylongaccess}[1]{%
  \csname glo@#1@longaccess\endcsname
}
```

trylongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
\newcommand*{\glsentrylongpluralaccess}[1]{%
  \csname glo@#1@longpluralaccess\endcsname
}
```

\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}%
}
```

`lsnameaccessdisplay` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
\DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentryaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

`lstextaccessdisplay` As above but for the textaccess replacement text.

```
\DeclareRobustCommand*{\glstextaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrytextaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

`pluralaccessdisplay` As above but for the pluralaccess replacement text.

```
\DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrypluralaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

`sfirstaccessdisplay` As above but for the firstaccess replacement text.

```
\DeclareRobustCommand*{\glsfirstaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentryfirstaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

pluralaccessdisplay As above but for the firstpluralaccess replacement text.

```
\DeclareRobustCommand*{\glsfirstpluralaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentryfirstpluralaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

symbolaccessdisplay As above but for the symbolaccess replacement text.

```
\DeclareRobustCommand*{\glssymbolaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrysymbolaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```
\DeclareRobustCommand*{\glssymbolpluralaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrysymbolpluralaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

iptionaccessdisplay As above but for the descriptionaccess replacement text.

```
\DeclareRobustCommand*{\glsdescriptionaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrydescaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

pluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
\DeclareRobustCommand*{\glsdescriptionpluralaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrydescpluralaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
```

```
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

sshortaccessdisplay As above but for the shortaccess replacement text.

```
\DeclareRobustCommand*{\glsshortaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentryshortaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

pluralaccessdisplay As above but for the shortpluralaccess replacement text.

```
\DeclareRobustCommand*{\glsshortpluralaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentryshortpluralaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

lslongaccessdisplay As above but for the longaccess replacement text.

```
\DeclareRobustCommand*{\glslongaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrylongaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

pluralaccessdisplay As above but for the longpluralaccess replacement text.

```
\DeclareRobustCommand*{\glslongpluralaccessdisplay}[2]{%
  \protected@edef\@glo@access{\glsentrylongpluralaccess{#2}}%
  \ifx\@glo@access\@gls@noaccess
    #1%
  \else
    \xglsaccsupp{\@glo@access}{#1}%
  \fi
}
```

\glsaccessdisplay 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}%
  }%
}
```

\@gls@ Redefine \@gls@ to change the way the link text is defined

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

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

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

Determine what the link text should be (this is stored in \@glo@text). This is no longer expanded.

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{\csname gls@\@glo@type @display\endcsname
        {\glstextaccessdisplay{\glsentrytext{#2}}{#2}}%
        {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
        {\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
        {#3}}%
    }%
    {%
      \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
        {\glsfirstaccessdisplay{\glsentryfirst{#2}}{#2}}%
        {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
        {\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
        {#3}}%
    }%
```

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

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\@glo@text}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse{\(\boolean{@glsisacronymlist}\AND
        \boolean{glsacrfootnote}\) \OR\NOT\boolean{glshyperfirst}}%
      {%
```

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

Indicate that this entry has now been used

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

`\@Gls@`

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

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

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

Determine what the link text should be (this is stored in `\@glo@text`). The first character of the entry text is converted to uppercase before passing to `\gls@`⟨*type*⟩`@display` or `\gls@`⟨*type*⟩`@displayfirst`

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{\csname gls@\@glo@type @display\endcsname
        {\glstextaccessdisplay{\Glsentrytext{#2}}{#2}}%
        {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
        {\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
        {#3}}%
    }%
    {%
      \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
        {\glsfirstaccessdisplay{\Glsentryfirst{#2}}{#2}}%
        {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
        {\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
        {#3}}%
    }%
```

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

```
  \ifglsused{#2}%
  {%
    \@gls@link[#1]{#2}{\@glo@text}%
  }%
  {%
    \gls@checkisacronymlist\@glo@type
    \ifthenelse{\(\boolean{@glsisacronymlist}\AND
      \boolean{glsacrfootnote}\) \OR\NOT\boolean{glshyperfirst}}%
    {%
      \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
    }%
    {%
    \@gls@link[#1]{#2}{\@glo@text}%
    }%
  }%
```

Indicate that this entry has now been used

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

\@GLS@

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{\csname gls@\@glo@type @display\endcsname
        {\glstextaccessdisplay{\glsentrytext{#2}}{#2}}%
        {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
        {\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
        {#3}}%
    }%
    {%
      \edef\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
        {\glsfirstaccessdisplay{\glsentryfirst{#2}}{#2}}%
        {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
        {\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
        {#3}}%
    }%
```

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

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse{\(\boolean{@glsisacronymlist}\AND
        \boolean{glsacrfootnote}\) \OR\NOT\boolean{glshyperfirst}}{%
        \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
      }%
      {%
        \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
      }%
    }%
```

Indicate that this entry has now been used

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

\@gls@pl@

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{\csname gls@\@glo@type @display\endcsname
        {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%
        {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
        {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
        {#3}}%
    }%
    {%
      \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
        {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
        {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
        {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
        {#3}}%
    }%
```

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

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\@glo@text}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse{\(\boolean{@glsisacronymlist}\AND
        \boolean{glsacrfootnote}\) \OR\NOT\boolean{glshyperfirst}}%
      {%
        \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
      }%
      {%
        \@gls@link[#1]{#2}{\@glo@text}%
      }%
    }%
```

Indicate that this entry has now been used

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

\@Glspl@

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{\csname gls@\@glo@type @display\endcsname
        {\glspluralaccessdisplay{\Glsentryplural{#2}}{#2}}%
        {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
        {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
        {#3}}%
    }%
    {%
      \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
        {\glsfirstpluralaccessdisplay{\Glsentryfirstplural{#2}}{#2}}%
        {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
        {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
        {#3}}%
    }%
```

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

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\@glo@text}%
    }%
    {%
      \ifthenelse{\equal{\@glo@type}{\acronymtype}\and
        \boolean{glsacrfootnote}}%
      {%
        \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
      }%
      {%
        \@gls@link[#1]{#2}{\@glo@text}%
      }%
    }%
```

Indicate that this entry has now been used

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

\@GLSpl@

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

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

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

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

```
    \ifglsused{#2}%
    {%
      \def\@glo@text{\csname gls@\@glo@type @display\endcsname
        {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%
        {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
        {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
        {#3}}%
    }%
    {%
      \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
      {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
      {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
      {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
      {#3}}%
    }%
```

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

```
    \ifglsused{#2}%
    {%
      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
    }%
    {%
      \gls@checkisacronymlist\@glo@type
      \ifthenelse{\(\boolean{@glsisacronymlist}\AND
        \boolean{glsacrfootnote}\)\OR\NOT\boolean{glshyperfirst}}%
      {%
        \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
      }%
      {%
        \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
      }%
    }%
```

Indicate that this entry has now been used

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

\@acrshort

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

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

```
    \def\@glo@text{%
      \glsshortaccessdisplay{\glsentryshort{#2}}{#2}%
    }%
```

Call `\@gls@link`

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

\@Acrshort

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

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

```
    \def\@glo@text{%
      \glsshortaccessdisplay{\Glsentryshort{#2}}{#2}%
    }%
```

Call `\@gls@link`

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

\@ACRshort

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

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

```
    \def\@glo@text{%
      \glsshortaccessdisplay{\MakeUppercase{\glsentryshort{#2}}}{#2}%
    }%
```

Call \@gls@link

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

\@acrlong

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

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

```
    \def\@glo@text{%
      \glslongaccessdisplay{\glsentrylong{#2}}{#2}%
    }%
```

Call \@gls@link

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

\@Acrlong

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

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

```
    \def\@glo@text{%
      \glslongaccessdisplay{\Glsentrylong{#2}}{#2}%
    }%
```

Call \@gls@link

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

\@ACRlong

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

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

```
    \def\@glo@text{%
      \glslongaccessdisplay{\MakeUppercase{\glsentrylong{#2}}}{#2}%
    }%
```

Call \@gls@link

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

5.3 Displaying the Glossary

Entries within the glossary or list of acronyms are now formatted via \accsuppglossaryentryfield and \accsuppglossarysubentryfield.

@glossaryentryfield

```
\ifglsxindy
  \renewcommand*{\@glossaryentryfield}{%
     \string\\accsuppglossaryentryfield}
\else
  \renewcommand*{\@glossaryentryfield}{%
     \string\accsuppglossaryentryfield}
\fi
```

ossarysubentryfield

```
\ifglsxindy
  \renewcommand*{\@glossarysubentryfield}{%
    \string\\accsuppglossarysubentryfield}
\else
  \renewcommand*{\@glossarysubentryfield}{%
    \string\accsuppglossarysubentryfield}
\fi
```

pglossaryentryfield

```
\newcommand*{\accsuppglossaryentryfield}[5]{%
  \glossaryentryfield{#1}%
  {\glsnameaccessdisplay{#2}{#1}}%
  {\glsdescriptionaccessdisplay{#3}{#1}}%
  {\glssymbolaccessdisplay{#4}{#1}}{#5}%
}
```

ossarysubentryfield

```
\newcommand*{\accsuppglossarysubentryfield}[6]{%
  \glossaryentryfield{#1}{#2}%
  {\glsnameaccessdisplay{#3}{#2}}%
  {\glsdescriptionaccessdisplay{#4}{#2}}%
  {\glssymbolaccessdisplay{#5}{#2}}{#6}%
}
```

5.4 Acronyms

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}%
}
```

efaultNewAcronymDef 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
    }%
  }%
  \@do@newglossaryentry
}
```

otnoteNewAcronymDef

```
\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
    }%
  }%
  \@do@newglossaryentry
}
```

iptionNewAcronymDef

```
\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}%
  }%
  \@do@newglossaryentry
}
```

otnoteNewAcronymDef

```
\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
    }%
  }%
  \@do@newglossaryentry
}
```

\SmallNewAcronymDef

```
\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
    }%
  }%
  \@do@newglossaryentry
}
```

The following are kept for compatibility with versions before 3.0:

\glsshortaccesskey

```
\newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

hortpluralaccesskey

```
\newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

\glslongaccesskey

```
\newcommand*{\glslongaccesskey}{\glslongkey access}%
```

longpluralaccesskey

```
\newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

\showglonameaccess

```
\newcommand*{\showglonameaccess}[1]{%
  \expandafter\show\csname glo@#1@textaccess\endcsname
}
```

\showglotextaccess

```
\newcommand*{\showglotextaccess}[1]{%
  \expandafter\show\csname glo@#1@textaccess\endcsname
}
```

showglopluralaccess

```
\newcommand*{\showglopluralaccess}[1]{%
  \expandafter\show\csname glo@#1@pluralaccess\endcsname
}
```

\showglofirstaccess

```
\newcommand*{\showglofirstaccess}[1]{%
  \expandafter\show\csname glo@#1@firstaccess\endcsname
}
```

lofirstpluralaccess

```
\newcommand*{\showglofirstpluralaccess}[1]{%
  \expandafter\show\csname glo@#1@firstpluralaccess\endcsname
}
```

showglosymbolaccess

```
\newcommand*{\showglosymbolaccess}[1]{%
  \expandafter\show\csname glo@#1@symbolaccess\endcsname
}
```

osymbolpluralaccess

```
\newcommand*{\showglosymbolpluralaccess}[1]{%
  \expandafter\show\csname glo@#1@symbolpluralaccess\endcsname
}
```

\showglodescaccess

```
\newcommand*{\showglodescaccess}[1]{%
  \expandafter\show\csname glo@#1@descaccess\endcsname
}
```

glodescpluralaccess

```
\newcommand*{\showglodescpluralaccess}[1]{%
  \expandafter\show\csname glo@#1@descpluralaccess\endcsname
}
```

\showgloshortaccess

```
\newcommand*{\showgloshortaccess}[1]{%
  \expandafter\show\csname glo@#1@shortaccess\endcsname
}
```

loshortpluralaccess

```
\newcommand*{\showgloshortpluralaccess}[1]{%
  \expandafter\show\csname glo@#1@shortpluralaccess\endcsname
}
```

\showglolongaccess

```
\newcommand*{\showglolongaccess}[1]{%
  \expandafter\show\csname glo@#1@longaccess\endcsname
}
```

glolongpluralaccess

```
\newcommand*{\showglolongpluralaccess}[1]{%
  \expandafter\show\csname glo@#1@longpluralaccess\endcsname
}
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex.

6.1 Babel Captions

Define captions if multi-lingual support is required, but the package is not loaded.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-babel}[2009/04/16 v1.2 (NLCT)]
```

English:

```
\@ifundefined{captionsenglish}{}{%
  \addto\captionsenglish{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}%
}
\@ifundefined{captionsamerican}{}{%
  \addto\captionsamerican{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}%
}
\@ifundefined{captionsaustralian}{}{%
  \addto\captionsaustralian{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}%
}
\@ifundefined{captionsbritish}{}{%
  \addto\captionsbritish{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}}%
\@ifundefined{captionscanadian}{}{%
```

```
  \addto\captionscanadian{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}%
}
\@ifundefined{captionsnewzealand}{}{%
  \addto\captionsnewzealand{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}%
}
\@ifundefined{captionsUKenglish}{}{%
  \addto\captionsUKenglish{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}%
}
\@ifundefined{captionsUSenglish}{}{%
  \addto\captionsUSenglish{%
    \renewcommand*{\glossaryname}{Glossary}%
    \renewcommand*{\acronymname}{Acronyms}%
    \renewcommand*{\entryname}{Notation}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Page List}%
    \renewcommand*{\glssymbolsgroupname}{Symbols}%
    \renewcommand*{\glsnumbersgroupname}{Numbers}%
}%
}
```

German (quite a few variations were suggested for German; I settled on the following):

```
\@ifundefined{captionsgerman}{}{%
  \addto\captionsgerman{%
    \renewcommand*{\glossaryname}{Glossar}%
    \renewcommand*{\acronymname}{Akronyme}%
    \renewcommand*{\entryname}{Bezeichnung}%
    \renewcommand*{\descriptionname}{Beschreibung}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Seiten}%
    \renewcommand*{\glssymbolsgroupname}{Symbole}%
    \renewcommand*{\glsnumbersgroupname}{Zahlen}}
}
```

ngerman is identical to German:

```
\@ifundefined{captionsngerman}{}{%
  \addto\captionsngerman{%
    \renewcommand*{\glossaryname}{Glossar}%
    \renewcommand*{\acronymname}{Akronyme}%
    \renewcommand*{\entryname}{Bezeichnung}%
    \renewcommand*{\descriptionname}{Beschreibung}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Seiten}%
    \renewcommand*{\glssymbolsgroupname}{Symbole}%
    \renewcommand*{\glsnumbersgroupname}{Zahlen}}
}
```

Italian:

```
\@ifundefined{captionsitalian}{}{%
  \addto\captionsitalian{%
    \renewcommand*{\glossaryname}{Glossario}%
    \renewcommand*{\acronymname}{Acronimi}%
    \renewcommand*{\entryname}{Nomenclatura}%
    \renewcommand*{\descriptionname}{Descrizione}%
    \renewcommand*{\symbolname}{Simbolo}%
    \renewcommand*{\pagelistname}{Elenco delle pagine}%
    \renewcommand*{\glssymbolsgroupname}{Simboli}%
    \renewcommand*{\glsnumbersgroupname}{Numeri}}
}
```

Dutch:

```
\@ifundefined{captionsdutch}{}{%
  \addto\captionsdutch{%
    \renewcommand*{\glossaryname}{Woordenlijst}%
    \renewcommand*{\acronymname}{Acroniemen}%
    \renewcommand*{\entryname}{Benaming}%
    \renewcommand*{\descriptionname}{Beschrijving}%
    \renewcommand*{\symbolname}{Symbool}%
    \renewcommand*{\pagelistname}{Pagina's}%
    \renewcommand*{\glssymbolsgroupname}{Symbolen}%
    \renewcommand*{\glsnumbersgroupname}{Cijfers}}
}
```

Spanish:

```
\@ifundefined{captionsspanish}{}{%
  \addto\captionsspanish{%
    \renewcommand*{\glossaryname}{Glosario}%
    \renewcommand*{\acronymname}{Siglas}%
    \renewcommand*{\entryname}{Entrada}%
    \renewcommand*{\descriptionname}{Descripci\'on}%
    \renewcommand*{\symbolname}{S\'{\i}mbolo}%
    \renewcommand*{\pagelistname}{Lista de p\'aginas}%
    \renewcommand*{\glssymbolsgroupname}{S\'{\i}mbolos}%
    \renewcommand*{\glsnumbersgroupname}{N\'umeros}}
}
```

French:

```
\@ifundefined{captionsfrench}{}{%
  \addto\captionsfrench{%
    \renewcommand*{\glossaryname}{Glossaire}%
    \renewcommand*{\acronymname}{Acronymes}%
    \renewcommand*{\entryname}{Terme}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbole}%
    \renewcommand*{\pagelistname}{Pages}%
    \renewcommand*{\glssymbolsgroupname}{Symboles}%
    \renewcommand*{\glsnumbersgroupname}{Nombres}}
}
\@ifundefined{captionsfrenchb}{}{%
  \addto\captionsfrenchb{%
    \renewcommand*{\glossaryname}{Glossaire}%
    \renewcommand*{\acronymname}{Acronymes}%
    \renewcommand*{\entryname}{Terme}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbole}%
    \renewcommand*{\pagelistname}{Pages}%
    \renewcommand*{\glssymbolsgroupname}{Symboles}%
    \renewcommand*{\glsnumbersgroupname}{Nombres}}
}
\@ifundefined{captionsfrancais}{}{%
  \addto\captionsfrancais{%
    \renewcommand*{\glossaryname}{Glossaire}%
    \renewcommand*{\acronymname}{Acronymes}%
    \renewcommand*{\entryname}{Terme}%
    \renewcommand*{\descriptionname}{Description}%
    \renewcommand*{\symbolname}{Symbole}%
    \renewcommand*{\pagelistname}{Pages}%
    \renewcommand*{\glssymbolsgroupname}{Symboles}%
    \renewcommand*{\glsnumbersgroupname}{Nombres}}
}
```

Danish:

```
\@ifundefined{captionsdanish}{}{%
  \addto\captionsdanish{%
    \renewcommand*{\glossaryname}{Ordliste}%
    \renewcommand*{\acronymname}{Akronymer}%
    \renewcommand*{\entryname}{Symbolforklaring}%
    \renewcommand*{\descriptionname}{Beskrivelse}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Side}%
    \renewcommand*{\glssymbolsgroupname}{Symboler}%
    \renewcommand*{\glsnumbersgroupname}{Tal}}
}
```

Irish:

```
\@ifundefined{captionsirish}{}{%
  \addto\captionsirish{%
    \renewcommand*{\glossaryname}{Gluais}%
    \renewcommand*{\acronymname}{Acrainmneacha}%
```

wasn't sure whether to go for Nóta (Note), Ciall ('Meaning', 'sense') or Brí ('Meaning'). In the end I chose Ciall.

```
    \renewcommand*{\entryname}{Ciall}%
    \renewcommand*{\descriptionname}{Tuairisc}%
```

Again, not sure whether to use Comhartha/Comharthaí or Siombail/Siombaile, so have chosen the former.

```
    \renewcommand*{\symbolname}{Comhartha}%
    \renewcommand*{\glssymbolsgroupname}{Comhartha\'{\i}}%
    \renewcommand*{\pagelistname}{Leathanaigh}%
    \renewcommand*{\glsnumbersgroupname}{Uimhreacha}}
}
```

Hungarian:

```
\@ifundefined{captionsmagyar}{}{%
  \addto\captionsmagyar{%
    \renewcommand*{\glossaryname}{Sz\'ojegyz\'ek}%
    \renewcommand*{\acronymname}{Bet\H uszavak}%
    \renewcommand*{\entryname}{Kifejez\'es}%
    \renewcommand*{\descriptionname}{Magyar\'azat}%
    \renewcommand*{\symbolname}{Jel\"ol\'es}%
    \renewcommand*{\pagelistname}{Oldalsz\'am}%
    \renewcommand*{\glssymbolsgroupname}{Jelek}%
    \renewcommand*{\glsnumbersgroupname}{Sz\'amjegyek}%
  }
}
\@ifundefined{captionshungarian}{}{%
  \addto\captionshungarian{%
    \renewcommand*{\glossaryname}{Sz\'ojegyz\'ek}%
    \renewcommand*{\acronymname}{Bet\H uszavak}%
    \renewcommand*{\entryname}{Kifejez\'es}%
    \renewcommand*{\descriptionname}{Magyar\'azat}%
    \renewcommand*{\symbolname}{Jel\"ol\'es}%
    \renewcommand*{\pagelistname}{Oldalsz\'am}%
    \renewcommand*{\glssymbolsgroupname}{Jelek}%
    \renewcommand*{\glsnumbersgroupname}{Sz\'amjegyek}%
  }
}
```

Polish

```
\@ifundefined{captionspolish}{}{%
  \addto\captionspolish{%
    \renewcommand*{\glossaryname}{S{\l}ownik termin\'ow}%
    \renewcommand*{\acronymname}{Skr\'ot}%
    \renewcommand*{\entryname}{Termin}%
    \renewcommand*{\descriptionname}{Opis}%
    \renewcommand*{\symbolname}{Symbol}%
    \renewcommand*{\pagelistname}{Strony}%
    \renewcommand*{\glssymbolsgroupname}{Symbole}%
    \renewcommand*{\glsnumbersgroupname}{Liczby}}
}
```

Brazilian

```
\@ifundefined{captionsbrazil}{}{%
  \addto\captionsbrazil{%
    \renewcommand*{\glossaryname}{Gloss\'ario}%
    \renewcommand*{\acronymname}{Siglas}%
    \renewcommand*{\entryname}{Nota\c c\~ao}%
    \renewcommand*{\descriptionname}{Descri\c c\~ao}%
    \renewcommand*{\symbolname}{S\'imbolo}%
    \renewcommand*{\pagelistname}{Lista de P\'aginas}%
    \renewcommand*{\glssymbolsgroupname}{S\'imbolos}%
    \renewcommand*{\glsnumbersgroupname}{N\'umeros}%
  }%
}
```

6.2 Polyglossia Captions

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-polyglossia}[2009/11/09 v1.0 (NLCT)]
```

English:

```
\@ifundefined{captionsenglish}{}{%
  \expandafter\toks@\expandafter{\captionsenglish
    \renewcommand*{\glossaryname}{\textenglish{Glossary}}%
    \renewcommand*{\acronymname}{\textenglish{Acronyms}}%
    \renewcommand*{\entryname}{\textenglish{Notation}}%
    \renewcommand*{\descriptionname}{\textenglish{Description}}%
    \renewcommand*{\symbolname}{\textenglish{Symbol}}%
    \renewcommand*{\pagelistname}{\textenglish{Page List}}%
    \renewcommand*{\glssymbolsgroupname}{\textenglish{Symbols}}%
    \renewcommand*{\glsnumbersgroupname}{\textenglish{Numbers}}%
  }%
  \edef\captionsenglish{\the\toks@}%
}
```

German:

```
\@ifundefined{captionsgerman}{}{%
  \expandafter\toks@\expandafter{\captionsgerman
    \renewcommand*{\glossaryname}{\textgerman{Glossar}}%
    \renewcommand*{\acronymname}{\textgerman{Akronyme}}%
    \renewcommand*{\entryname}{\textgerman{Bezeichnung}}%
    \renewcommand*{\descriptionname}{\textgerman{Beschreibung}}%
    \renewcommand*{\symbolname}{\textgerman{Symbol}}%
    \renewcommand*{\pagelistname}{\textgerman{Seiten}}%
    \renewcommand*{\glssymbolsgroupname}{\textgerman{Symbole}}%
    \renewcommand*{\glsnumbersgroupname}{\textgerman{Zahlen}}%
  }%
  \edef\captionsgerman{\the\toks@}%
}
```

Italian:

```
\@ifundefined{captionsitalian}{}{%
  \expandafter\toks@\expandafter{\captionsitalian
    \renewcommand*{\glossaryname}{\textitalian{Glossario}}%
    \renewcommand*{\acronymname}{\textitalian{Acronimi}}%
    \renewcommand*{\entryname}{\textitalian{Nomenclatura}}%
    \renewcommand*{\descriptionname}{\textitalian{Descrizione}}%
    \renewcommand*{\symbolname}{\textitalian{Simbolo}}%
    \renewcommand*{\pagelistname}{\textitalian{Elenco delle pagine}}%
    \renewcommand*{\glssymbolsgroupname}{\textitalian{Simboli}}%
    \renewcommand*{\glsnumbersgroupname}{\textitalian{Numeri}}%
  }%
  \edef\captionsitalian{\the\toks@}%
}
```

Dutch:

```
\@ifundefined{captionsdutch}{}{%
  \expandafter\toks@\expandafter{\captionsdutch
    \renewcommand*{\glossaryname}{\textdutch{Woordenlijst}}%
    \renewcommand*{\acronymname}{\textdutch{Acroniemen}}%
    \renewcommand*{\entryname}{\textdutch{Benaming}}%
    \renewcommand*{\descriptionname}{\textdutch{Beschrijving}}%
    \renewcommand*{\symbolname}{\textdutch{Symbool}}%
    \renewcommand*{\pagelistname}{\textdutch{Pagina's}}%
    \renewcommand*{\glssymbolsgroupname}{\textdutch{Symbolen}}%
    \renewcommand*{\glsnumbersgroupname}{\textdutch{Cijfers}}%
  }%
  \edef\captionsdutch{\the\toks@}%
}
```

Spanish:

```
\@ifundefined{captionsspanish}{}{%
  \expandafter\toks@\expandafter{\captionsspanish
    \renewcommand*{\glossaryname}{\textspanish{Glosario}}%
    \renewcommand*{\acronymname}{\textspanish{Siglas}}%
    \renewcommand*{\entryname}{\textspanish{Entrada}}%
    \renewcommand*{\descriptionname}{\textspanish{Descripci\'on}}%
    \renewcommand*{\symbolname}{\textspanish{S\'{\i}mbolo}}%
    \renewcommand*{\pagelistname}{\textspanish{Lista de p\'aginas}}%
    \renewcommand*{\glssymbolsgroupname}{\textspanish{S\'{\i}mbolos}}%
    \renewcommand*{\glsnumbersgroupname}{\textspanish{N\'umeros}}%
  }%
  \edef\captionsspanish{\the\toks@}%
}
```

French:

```
\@ifundefined{captionsfrench}{}{%
  \expandafter\toks@\expandafter{\captionsfrench
    \renewcommand*{\glossaryname}{\textfrench{Glossaire}}%
    \renewcommand*{\acronymname}{\textfrench{Acronymes}}%
    \renewcommand*{\entryname}{\textfrench{Terme}}%
    \renewcommand*{\descriptionname}{\textfrench{Description}}%
    \renewcommand*{\symbolname}{\textfrench{Symbole}}%
    \renewcommand*{\pagelistname}{\textfrench{Pages}}%
    \renewcommand*{\glssymbolsgroupname}{\textfrench{Symboles}}%
    \renewcommand*{\glsnumbersgroupname}{\textfrench{Nombres}}%
  }%
  \edef\captionsfrench{\the\toks@}%
}
```

Danish:

```
\@ifundefined{captionsdanish}{}{%
  \expandafter\toks@\expandafter{\captionsdanish
    \renewcommand*{\glossaryname}{\textdanish{Ordliste}}%
    \renewcommand*{\acronymname}{\textdanish{Akronymer}}%
    \renewcommand*{\entryname}{\textdanish{Symbolforklaring}}%
    \renewcommand*{\descriptionname}{\textdanish{Beskrivelse}}%
    \renewcommand*{\symbolname}{\textdanish{Symbol}}%
    \renewcommand*{\pagelistname}{\textdanish{Side}}%
    \renewcommand*{\glssymbolsgroupname}{\textdanish{Symboler}}%
    \renewcommand*{\glsnumbersgroupname}{\textdanish{Tal}}%
  }%
  \edef\captionsdanish{\the\toks@}%
}
```

Irish:

```
\@ifundefined{captionsirish}{}{%
  \expandafter\toks@\expandafter{\captionsirish
    \renewcommand*{\glossaryname}{\textirish{Gluais}}%
    \renewcommand*{\acronymname}{\textirish{Acrainmneacha}}%
    \renewcommand*{\entryname}{\textirish{Ciall}}%
    \renewcommand*{\descriptionname}{\textirish{Tuairisc}}%
    \renewcommand*{\symbolname}{\textirish{Comhartha}}%
    \renewcommand*{\glssymbolsgroupname}{\textirish{Comhartha\'{\i}}}%
    \renewcommand*{\pagelistname}{\textirish{Leathanaigh}}%
    \renewcommand*{\glsnumbersgroupname}{\textirish{Uimhreacha}}%
  }%
  \edef\captionsirish{\the\toks@}%
}
```

Hungarian:

```
\@ifundefined{captionsmagyar}{}{%
  \expandafter\toks@\expandafter{\captionsmagyar
    \renewcommand*{\glossaryname}{\textmagyar{Sz\'ojegyz\'ek}}%
    \renewcommand*{\acronymname}{\textmagyar{Bet\H uszavak}}%
    \renewcommand*{\entryname}{\textmagyar{Kifejez\'es}}%
    \renewcommand*{\descriptionname}{\textmagyar{Magyar\'azat}}%
    \renewcommand*{\symbolname}{\textmagyar{Jel\"ol\'es}}%
    \renewcommand*{\pagelistname}{\textmagyar{Oldalsz\'am}}%
    \renewcommand*{\glssymbolsgroupname}{\textmagyar{Jelek}}%
    \renewcommand*{\glsnumbersgroupname}{\textmagyar{Sz\'amjegyek}}%
  }%
  \edef\captionsmagyar{\the\toks@}%
}
```

Polish

```
\@ifundefined{captionspolish}{}{%
  \expandafter\toks@\expandafter{\captionspolish
    \renewcommand*{\glossaryname}{\textpolish{S{\l}ownik termin\'ow}}%
    \renewcommand*{\acronymname}{\textpolish{Skr\'ot}}%
    \renewcommand*{\entryname}{\textpolish{Termin}}%
    \renewcommand*{\descriptionname}{\textpolish{Opis}}%
    \renewcommand*{\symbolname}{\textpolish{Symbol}}%
    \renewcommand*{\pagelistname}{\textpolish{Strony}}%
    \renewcommand*{\glssymbolsgroupname}{\textpolish{Symbole}}%
    \renewcommand*{\glsnumbersgroupname}{\textpolish{Liczby}}%
  }%
  \edef\captionspolish{\the\toks@}%
}
```

Portugues

```
\@ifundefined{captionsportuges}{}{%
  \expandafter\toks@\expandafter{\captionsportuges
    \renewcommand*{\glossaryname}{\textportuges{Gloss\'ario}}%
    \renewcommand*{\acronymname}{\textportuges{Siglas}}%
    \renewcommand*{\entryname}{\textportuges{Nota\c c\~ao}}%
    \renewcommand*{\descriptionname}{\textportuges{Descri\c c\~ao}}%
    \renewcommand*{\symbolname}{\textportuges{S\'imbolo}}%
    \renewcommand*{\pagelistname}{\textportuges{Lista de P\'aginas}}%
    \renewcommand*{\glssymbolsgroupname}{\textportuges{S\'imbolos}}%
    \renewcommand*{\glsnumbersgroupname}{\textportuges{N\'umeros}}%
  }%
  \edef\captionsportuges{\the\toks@}%
}
```

6.3 Brazilian Dictionary

This is a dictionary file provided by Thiago de Melo for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Brazilian}
```

Provide Brazilian translations:

```
\providetranslation{Glossary}{Gloss\'ario}
\providetranslation{Acronyms}{Siglas}
\providetranslation{Notation (glossaries)}{Nota\c c\~ao}
\providetranslation{Description (glossaries)}{Descri\c c\~ao}
\providetranslation{Symbol (glossaries)}{S\'imbolo}
\providetranslation{Page List (glossaries)}{Lista de P\'aginas}
\providetranslation{Symbols (glossaries)}{S\'imbolos}
\providetranslation{Numbers (glossaries)}{N\'umeros}
```

6.4 Danish Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Danish}
```

Provide Danish translations:

```
\providetranslation{Glossary}{Ordliste}
\providetranslation{Acronyms}{Akronymer}
\providetranslation{Notation (glossaries)}{Symbolforklaring}
\providetranslation{Description (glossaries)}{Beskrivelse}
\providetranslation{Symbol (glossaries)}{Symbol}
\providetranslation{Page List (glossaries)}{Side}
\providetranslation{Symbols (glossaries)}{Symboler}
\providetranslation{Numbers (glossaries)}{Tal}
```

6.5 Dutch Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Dutch}
```

Provide Dutch translations:

```
\providetranslation{Glossary}{Woordenlijst}
\providetranslation{Acronyms}{Acroniemen}
\providetranslation{Notation (glossaries)}{Benaming}
\providetranslation{Description (glossaries)}{Beschrijving}
\providetranslation{Symbol (glossaries)}{Symbool}
\providetranslation{Page List (glossaries)}{Pagina's}
\providetranslation{Symbols (glossaries)}{Symbolen}
\providetranslation{Numbers (glossaries)}{Cijfers}
```

6.6 English Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{English}
```

Provide English translations:

```
\providetranslation{Glossary}{Glossary}
\providetranslation{Acronyms}{Acronyms}
\providetranslation{Notation (glossaries)}{Notation}
\providetranslation{Description (glossaries)}{Description}
\providetranslation{Symbol (glossaries)}{Symbol}
\providetranslation{Page List (glossaries)}{Page List}
\providetranslation{Symbols (glossaries)}{Symbols}
\providetranslation{Numbers (glossaries)}{Numbers}
```

6.7 French Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{French}
```

Provide French translations:

```
\providetranslation{Glossary}{Glossaire}
\providetranslation{Acronyms}{Acronymes}
\providetranslation{Notation (glossaries)}{Terme}
\providetranslation{Description (glossaries)}{Description}
\providetranslation{Symbol (glossaries)}{Symbole}
\providetranslation{Page List (glossaries)}{Pages}
\providetranslation{Symbols (glossaries)}{Symboles}
\providetranslation{Numbers (glossaries)}{Nombres}
```

6.8 German Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{German}
```

Provide German translations (quite a few variations were suggested for German; I settled on the following):

```
\providetranslation{Glossary}{Glossar}
\providetranslation{Acronyms}{Akronyme}
\providetranslation{Notation (glossaries)}{Bezeichnung}
\providetranslation{Description (glossaries)}{Beschreibung}
\providetranslation{Symbol (glossaries)}{Symbol}
\providetranslation{Page List (glossaries)}{Seiten}
\providetranslation{Symbols (glossaries)}{Symbole}
\providetranslation{Numbers (glossaries)}{Zahlen}
```

6.9 Irish Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Irish}
```

Provide Irish translations:

```
\providetranslation{Glossary}{Gluais}
\providetranslation{Acronyms}{Acrainmneacha}
```

```
\providetranslation{Notation (glossaries)}{Ciall}
\providetranslation{Description (glossaries)}{Tuairisc}
\providetranslation{Symbol (glossaries)}{Comhartha}
\providetranslation{Page List (glossaries)}{Leathanaigh}
\providetranslation{Symbols (glossaries)}{Comhartha\'{\i}}
\providetranslation{Numbers (glossaries)}{Uimhreacha}
```

6.10 Italian Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Italian}
```

Provide Italian translations:

```
\providetranslation{Glossary}{Glossario}
\providetranslation{Acronyms}{Acronimi}
\providetranslation{Notation (glossaries)}{Nomenclatura}
\providetranslation{Description (glossaries)}{Descrizione}
\providetranslation{Symbol (glossaries)}{Simbolo}
\providetranslation{Page List (glossaries)}{Elenco delle pagine}
\providetranslation{Symbols (glossaries)}{Simboli}
\providetranslation{Numbers (glossaries)}{Numeri}
```

6.11 Magyar Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Magyar}
```

Provide translations:

```
\providetranslation{Glossary}{Sz\'ojegyz\'ek}
\providetranslation{Acronyms}{Bet\H uszavak}
\providetranslation{Notation (glossaries)}{Kifejez\'es}
\providetranslation{Description (glossaries)}{Magyar\'azat}
\providetranslation{Symbol (glossaries)}{Jel\"ol\'es}
\providetranslation{Page List (glossaries)}{Oldalsz\'am}
\providetranslation{Symbols (glossaries)}{Jelek}
\providetranslation{Numbers (glossaries)}{Sz\'amjegyek}
```

6.12 Polish Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Polish}
```

Provide Polish translations:

```
\providetranslation{Glossary}{S{\l}ownik termin\'ow}
\providetranslation{Acronyms}{Skr\'ot}
\providetranslation{Notation (glossaries)}{Termin}
\providetranslation{Description (glossaries)}{Opis}
\providetranslation{Symbol (glossaries)}{Symbol}
\providetranslation{Page List (glossaries)}{Strony}
```

```
\providetranslation{Symbols (glossaries)}{Symbole}
\providetranslation{Numbers (glossaries)}{Liczby}
```

6.13 Serbian Dictionary

This dictionary was provided by Zoran Filipovic.

```
\ProvidesDictionary{glossaries-dictionary}{Serbian}
\providetranslation{Glossary}{Mali re\v cnik}
\providetranslation{Acronyms}{Skra\' cenice}
\providetranslation{Notation (glossaries)}{Oznaka}
\providetranslation{Description (glossaries)}{Opis}
\providetranslation{Symbol (glossaries)}{Simbol}
\providetranslation{Page List (glossaries)}{Stranica}
\providetranslation{Symbols (glossaries)}{Simboli}
\providetranslation{Numbers (glossaries)}{Brojevi}
```

6.14 Spanish Dictionary

This is a dictionary file provided for use with the package.

```
\ProvidesDictionary{glossaries-dictionary}{Spanish}
```

Provide Spanish translations:

```
\providetranslation{Glossary}{Glosario}
\providetranslation{Acronyms}{Siglas}
\providetranslation{Notation (glossaries)}{Entrada}
\providetranslation{Description (glossaries)}{Descripci\'on}
\providetranslation{Symbol (glossaries)}{S\'{\i}mbolo}
\providetranslation{Page List (glossaries)}{Lista de p\'aginas}
\providetranslation{Symbols (glossaries)}{S\'{\i}mbolos}
\providetranslation{Numbers (glossaries)}{N\'umeros}
```

Glossary

`makeindex` An indexing application. 17

`xindy` An flexible indexing application with multilingual support written in Perl. 17

Change History

1.01

General: Added range facility in format key 61

`\writeist`: Added spaces after 'delimN and 'delimR in ist file 118

1.03

`\makefirstuc`: changed 'protected@edef to 'def 170

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.