

Babel, a multilingual package for use with L^AT_EX's standard document classes*

Johannes Braams
Kersengarde 33
2723 BP Zoetermeer
The Netherlands
`babel@braams.xs4all.nl`

Printed October 27, 2010

Abstract

The standard distribution of L^AT_EX contains a number of document classes that are meant to be used, but also serve as examples for other users to create their own document classes. These document classes have become very popular among L^AT_EX users. But it should be kept in mind that they were designed for American tastes and typography. At one time they contained a number of hard-wired texts. This report describes `babel`, a package that makes use of the new capabilities of T_EX version 3 to provide an environment in which documents can be typeset in a language other than US English, or in more than one language.

Contents

1	The user interface	6
1.1	Languages supported by <code>Babel</code>	7
1.2	Workarounds	8
2	Changes for L^AT_EX 2_ε	8
3	Changes in <code>Babel</code> version 3.7	8
4	Changes in <code>Babel</code> version 3.6	9
5	Changes in <code>Babel</code> version 3.5	10
6	The interface between the core of <code>babel</code> and the language definition files	11
6.1	Support for active characters	12
6.2	Support for saving macro definitions	13
6.3	Support for extending macros	13
6.4	Macros common to a number of languages	13
7	Compatibility with <code>german.sty</code>	13
8	Compatibility with <code>ngerman.sty</code>	14
9	Compatibility with the <code>french</code> package	14

*During the development ideas from Nico Poppelier, Piet van Oostrum and many others have been used. Bernd Raichle has provided many helpful suggestions.

10 Identification	14
11 The Package File	15
11.1 Language options	15
12 The Kernel of Babel	18
12.1 Encoding issues (part 1)	19
12.2 Multiple languages	19
12.3 Support for active characters	32
12.4 Shorthands	33
12.5 Language attributes	42
12.6 Support for saving macro definitions	44
12.7 Support for extending macros	45
12.8 Macros common to a number of languages	45
12.9 Making glyphs available	46
12.10 Quotation marks	46
12.11 Letters	47
12.12 Shorthands for quotation marks	48
12.13 Umlauts and trema's	49
12.14 The redefinition of the style commands	50
12.14.1 Redefinition of macros	51
12.15 Cross referencing macros	55
12.16 marks	58
12.17 Encoding issues (part 2)	59
12.18 Preventing clashes with other packages	59
12.18.1 <code>ifthen</code>	59
12.18.2 <code>varioref</code>	60
12.18.3 <code>hhline</code>	61
12.18.4 <code>hyperref</code>	61
12.18.5 General	61
13 Local Language Configuration	62
14 Driver files for the documented source code	63
15 Conclusion	67
16 Acknowledgements	67
17 References	67
18 The Esperanto language	68
19 The Interlingua language	71
20 The Dutch language	73
21 The English language	77
22 The German language	81
23 The German language – new orthography	86
24 The Breton language	89
25 The Welsh language	92
26 The Irish language	94
27 The Scottish language	96

28 The Greek language	98
28.1 Typing conventions	98
28.2 Greek numbering	98
29 The French language	106
29.1 Basic interface	106
29.2 Customisation	107
29.3 Hyphenation checks	109
29.4 Changes	110
29.5 File frenchb.cfg	111
30 T_EXnical details	112
30.1 Initial setup	112
30.2 Punctuation	113
30.3 French quotation marks	116
30.4 Date in French	117
30.5 Extra utilities	117
30.6 Formatting numbers	120
30.7 Caption names	121
30.8 French lists	123
30.9 French indentation of sections	125
30.10 Formatting footnotes	125
30.11 Global layout	127
30.12 Dots	127
30.13 Setup options: keyval stuff	128
30.14 Clean up and exit	136
31 The Italian language	137
31.1 Support for etymological hyphenation	139
31.2 Facilities required by the ISO 31/XI regulations	141
31.3 Accents	142
31.4 <i>Caporali</i> or French double quotes	142
31.5 Finishing commands	145
31.6 References	145
32 The Latin language	146
33 Latin shortcuts	149
34 Etymological hyphenation	151
35 The Portuguese language	153
36 The Spanish language	157
36.1 The Code	161
37 The Catalan language	177
38 This file	184
39 The Galician language	184
39.1 The Code	186
40 The Basque language	203
41 The Romanian language	206
42 The Danish language	208

43 The Icelandic language	211
43.1 Overview	211
43.2 References	211
43.3 TeXnical details	212
43.4 Captionnames and date	213
43.5 Icelandic quotation marks	214
43.6 Old Icelandic	214
43.7 Formatting numbers	215
43.8 Extra utilities	217
44 The Norwegian language	219
45 The Swedish language	223
46 The North Sami language	227
46.1 The code of <code>samin.dtx</code>	227
47 The Finnish language	229
48 The Hungarian language	232
49 The Estonian language	247
49.1 Implementation	247
50 The Albanian language	251
51 The Croatian language	254
52 The Czech Language	256
52.1 Usage	256
52.2 Compatibility	256
52.3 Implementation	257
53 The Polish language	268
54 The Serbocroatian language	273
55 The Slovak language	276
55.1 Compatibility	277
55.2 Implementation	277
56 The Slovenian language	289
57 The Russian language	291
58 The Bulgarian language	303
59 The Ukrainian language	314
60 The Lower Sorbian language	326
61 The Upper Sorbian language	328
62 The Turkish language	331
63 The Hebrew language	334
63.1 Acknowledgement	334
63.2 The DOCSTRIP modules	334
63.3 Hebrew language definitions	335
63.3.1 Hebrew numerals	338
63.4 Right to left support	344

63.4.1	Switching from LR to RL mode and back	344
63.4.2	Counters	347
63.4.3	Preserving logos	348
63.4.4	List environments	348
63.4.5	Tables of moving stuff	349
63.4.6	Two-column mode	353
63.4.7	Footnotes	354
63.4.8	Headings and two-side support	354
63.4.9	Postscript Problems	357
63.4.10	Miscellaneous internal L ^A T _E X macros	358
63.4.11	Bibliography and citations	359
63.4.12	Additional bidirectional commands	361
63.5	Hebrew calendar	362
63.5.1	Introduction	363
63.5.2	Registers, Commands, Formatting Macros	363
63.5.3	Auxiliary Macros	365
63.5.4	Gregorian Part	366
63.5.5	Hebrew Part	367
64	Hebrew input encodings	371
64.1	Default definitions for characters	372
64.2	The SI-960 encoding	373
64.3	The ISO 8859-8 encoding and the MS Windows cp1255 encoding	373
64.4	The IBM code page 862	375
65	Hebrew font encodings	377
65.1	THIS SECTION IS OUT OF DATE. UPDATE DOCS TO MATCH HE8 ENCODING	377
65.2	The DOCSTRIP modules	378
65.3	The LHEencoding definition file	378
65.4	The font definition files (in LHE encoding)	380
65.4.1	Hebrew default font	380
65.4.2	Hebrew sans-serif font	380
65.4.3	Hebrew typewriter font	381
65.4.4	Hebrew classic font	381
65.4.5	Hebrew shalom fonts	382
65.4.6	Hebrew frank-ruehl font	382
65.4.7	Hebrew carmel font	383
65.4.8	Hebrew redis font	383
65.5	The HE8encoding definition file	384
65.5.1	CHECK HERE FOR HE8 UPDATES	384
65.6	The font definition files (in HE8 encoding)	386
65.6.1	Hebrew default font	386
65.6.2	Hebrew sans-serif font	386
65.6.3	Hebrew typewriter font	387
65.6.4	8Bit OmegaHebrew font	387
65.6.5	8Bit Aharoni font	387
65.6.6	8Bit David font	388
65.6.7	8Bit Drugulin font	388
65.6.8	8Bit Ellinia font	388
65.6.9	8Bit FrankRuehl font	388
65.6.10	8Bit KtavYad font	389
65.6.11	8Bit MiriamMono font	389
65.6.12	8Bit Nachlieli font	389
65.6.13	Hebrew font switching commands	389

66 Hebrew in L^AT_EX 2.09 compatibility mode	392
66.1 The DOCSTRIP modules	392
66.2 Obsolete style files	392
67 The Bahasa Indonesian language	394
68 The Bahasa Malaysia language	396
69 Not renaming hyphen.tex	398
70 Support for formats based on PLAIN_TE_X	399

1 The user interface

The user interface of this package is quite simple. It consists of a set of commands that switch from one language to another, and a set of commands that deal with shorthands. It is also possible to find out what the current language is.

<code>\selectlanguage</code>	When a user wants to switch from one language to another he can do so using the macro <code>\selectlanguage</code> . This macro takes the language, defined previously by a language definition file, as its argument. It calls several macros that should be defined in the language definition files to activate the special definitions for the language chosen.
<code>otherlanguage</code>	The environment <code>otherlanguage</code> does basically the same as <code>\selectlanguage</code> , except the language change is local to the environment. This environment is required for intermixing left-to-right typesetting with right-to-left typesetting. The language to switch to is specified as an argument to <code>\begin{otherlanguage}</code> .
<code>\foreignlanguage</code>	The command <code>\foreignlanguage</code> takes two arguments; the second argument is a phrase to be typeset according to the rules of the language named in its first argument. This command only switches the extra definitions and the hyphenation rules for the language, <i>not</i> the names and dates.
<code>otherlanguage*</code>	In the environment <code>otherlanguage*</code> only the typesetting is done according to the rules of the other language, but the text-strings such as ‘figure’, ‘table’, etc. are left as they were set outside this environment.
<code>hyphenrules</code>	The environment <code>hyphenrules</code> can be used to select <i>only</i> the hyphenation rules to be used. This can for instance be used to select ‘nohyphenation’, provided that in <code>language.dat</code> the ‘language’ nohyphenation is defined by loading <code>serohyph.tex</code> .
<code>\language</code>	The control sequence <code>\language</code> contains the name of the current language.
<code>\iflanguage</code>	If more than one language is used, it might be necessary to know which language is active at a specific time. This can be checked by a call to <code>\iflanguage</code> . This macro takes three arguments. The first argument is the name of a language; the second and third arguments are the actions to take if the result of the test is <code>true</code> or <code>false</code> respectively.
<code>\useshorthands</code>	The command <code>\useshorthands</code> initiates the definition of user-defined shorthand sequences. It has one argument, the character that starts these personal shorthands.
<code>\defineshorthand</code>	The command <code>\defineshorthand</code> takes two arguments: the first is a one- or two-character shorthand sequence, and the second is the code the shorthand should expand to.
<code>\aliasshorthand</code>	The command <code>\aliasshorthand</code> can be used to let another character perform the same functions as the default shorthand character. If one prefers for example to use the character / over " in typing polish texts, this can be achieved by entering <code>\aliasshorthand{"}{/}</code> . <i>Please note</i> that the substitute shorthand character must have been declared in the preamble of your document, using a command such as <code>\useshorthands{/}</code> in this example.
<code>\languageshorthands</code>	The command <code>\languageshorthands</code> can be used to switch the shorthands on

the language level. It takes one argument, the name of a language. Note that for this to work the language should have been specified as an option when loading the `babel` package.

`\shorthandon` It is sometimes necessary to switch a shorthand character off temporarily, because it must be used in an entirely different way. For this purpose, the user commands `\shorthandoff` and `\shorthandon` are provided. They each take a list of characters as their arguments. The command `\shorthandoff` sets the `\catcode` for each of the characters in its argument to other (12); the command `\shorthandon` sets the `\catcode` to active (13). Both commands only work on ‘known’ shorthand characters. If a character is not known to be a shorthand character its category code will be left unchanged.

`\languageattribute` This is a user-level command, to be used in the preamble of a document (after `\usepackage[...]{babel}`), that declares which attributes are to be used for a given language. It takes two arguments: the first is the name of the language; the second, a (list of) attribute(s) to used. The command checks whether the language is known in this document and whether the attribute(s) are known for this language.

1.1 Languages supported by Babel

In the following table all the languages supported by `Babel` are listed, together with the names of the options with which you can load `babel` for each language.

Language	Option(s)
Afrikaans	afrikaans
Bahasa	bahasa, indonesian, indon, bahasai, bahasam, malay, meyalu
Basque	basque
Breton	breton
Bulgarian	bulgarian
Catalan	catalan
Croatian	croatian
Czech	czech
Danish	danish
Dutch	dutch
English	english, USenglish, american, UKenglish, british, canadian, australian, newzealand
Esperanto	esperanto
Estonian	estonian
Finnish	finnish
French	french, francais, canadien, acadian
Galician	galician
German	austrian, german, germanb, ngerman, naustrian
Greek	greek, polutonikogreek
Hebrew	hebrew
Hungarian	magyar, hungarian
Icelandic	icelandic
Interlingua	interlingua
Irish Gaelic	irish
Italian	italian
Latin	latin
Lower Sorbian	lowersorbian
North Sami	samin
Norwegian	norsk, nynorsk
Polish	polish
Portuguese	portuges, portuguese, brazilian, brazil

Language	Option(s)
Romanian	romanian
Russian	russian
Scottish Gaelic	scottish
Spanish	spanish
Slovakian	slovak
Slovenian	slovene
Swedish	swedish
Serbian	serbian
Turkish	turkish
Ukrainian	ukrainian
Upper Sorbian	uppersorbian
Welsh	welsh

For some languages **babel** supports the options `activeacute` and `activegrave`; for typesetting Russian texts, **babel** knows about the options `LWN` and `LCY` to specify the fontencoding of the cyrillic font used. Currently only `LWN` is supported.

1.2 Workarounds

If you use the document class `book` *and* you use `\ref` inside the argument of `\chapter`, **L^AT_EX** will keep complaining about an undefined label. The reason is that the argument of `\ref` is passed through `\uppercase` at some time during processing. To prevent such problems, you could revert to using uppercase labels, or you can use `\lowercase{\ref{foo}}` inside the argument of `\chapter`.

2 Changes for L^AT_EX 2_ε

With the advent of L^AT_EX 2_ε the interface to **babel** in the preamble of the document has changed. With L^AT_EX 2.09 one used to call up the **babel** system with a line such as:

```
\documentstyle[dutch,english]{article}
```

which would tell L^AT_EX that the document would be written in two languages, Dutch and English, and that English would be the first language in use.

The L^AT_EX 2_ε way of providing the same information is:

```
\documentclass{article}
\usepackage[dutch,english]{babel}
```

or, making `dutch` and `english` global options in order to let other packages detect and use them:

```
\documentclass[dutch,english]{article}
\usepackage{babel}
\usepackage{varioref}
```

In this last example, the package **varioref** will also see the options and will be able to use them.

3 Changes in Babel version 3.7

In Babel version 3.7 a number of bugs that were found in version 3.6 are fixed. Also a number of changes and additions have occurred:

- Shorthands are expandable again. The disadvantage is that one has to type `'{a}` when the acute accent is used as a shorthand character. The advantage is that a number of other problems (such as the breaking of ligatures, etc.) have vanished.

- Two new commands, `\shorthandon` and `\shorthandoff` have been introduced to enable to temporarily switch off one or more shorthands.
- Support for typesetting Greek has been enhanced. Code from the `kdgreek` package (suggested by the author) was added and `\greeknumeral` has been added.
- Support for typesetting Basque is now available thanks to Juan Aguirregabiria.
- Support for typesetting Serbian with Latin script is now available thanks to Dejan Muhamedagić and Jankovic Slobodan.
- Support for typesetting Hebrew (and potential support for typesetting other right-to-left written languages) is now available thanks to Rama Porrat and Boris Lavva.
- Support for typesetting Bulgarian is now available thanks to Georgi Boshnakov.
- Support for typesetting Latin is now available, thanks to Claudio Beccari and Krzysztof Konrad Żelechowski.
- Support for typesetting North Sami is now available, thanks to Regnor Jernsletten.
- The options `canadian`, `canadien` and `acadien` have been added for Canadian English and French use.
- A language attribute has been added to the `\mark...` commands in order to make sure that a Greek header line comes out right on the last page before a language switch.
- Hyphenation pattern files are now read *inside a group*; therefore any changes a pattern file needs to make to lowercase codes, uppercase codes, and category codes are kept local to that group. If they are needed for the language, these changes will need to be repeated and stored in `\extras...`
- The concept of language attributes is introduced. It is intended to give the user some control over the features a language-definition file provides. Its first use is for the Greek language, where the user can choose the πολυτονικό (“Polutoniko” or multi-accented) Greek way of typesetting texts. These attributes will possibly find wider use in future releases.
- The environment `hyphenrules` is introduced.
- The syntax of the file `language.dat` has been extended to allow (optionally) specifying the font encoding to be used while processing the patterns file.
- The command `\providehyphenmins` should now be used in language definition files in order to be able to keep any settings provided by the pattern file.

4 Changes in Babel version 3.6

In Babel version 3.6 a number of bugs that were found in version 3.5 are fixed. Also a number of changes and additions have occurred:

- A new environment `otherlanguage*` is introduced. it only switches the ‘specials’, but leaves the ‘captions’ untouched.

- The shorthands are no longer fully expandable. Some problems could only be solved by peeking at the token following an active character. The advantage is that `'{a}` works as expected for languages that have the `'` active.
- Support for typesetting french texts is much enhanced; the file `francais.ldf` is now replaced by `frenchb.ldf` which is maintained by Daniel Flipo.
- Support for typesetting the russian language is again available. The language definition file was originally developed by Olga Lapko from CyrTUG. The fonts needed to typeset the russian language are now part of the `babel` distribution. The support is not yet up to the level which is needed according to Olga, but this is a start.
- Support for typesetting greek texts is now also available. What is offered in this release is a first attempt; it will be enhanced later on by Yannis Haralambous.
- in `babel 3.6j` some hooks have been added for the development of support for Hebrew typesetting.
- Support for typesetting texts in Afrikaans (a variant of Dutch, spoken in South Africa) has been added to `dutch.ldf`.
- Support for typesetting Welsh texts is now available.
- A new command `\aliasshorthand` is introduced. It seems that in Poland various conventions are used to type the necessary Polish letters. It is now possible to use the character `/` as a shorthand character instead of the character `"`, by issuing the command `\aliasshorthand{"}{/}`.
- The shorthand mechanism now deals correctly with characters that are already active.
- Shorthand characters are made active at `\begin{document}`, not earlier. This is to prevent problems with other packages.
- A *preambleonly* command `\substitutefontfamily` has been added to create `.fd` files on the fly when the font families of the Latin text differ from the families used for the Cyrillic or Greek parts of the text.
- Three new commands `\LdfInit`, `\ldf@quit` and `\ldf@finish` are introduced that perform a number of standard tasks.
- In `babel 3.6k` the language Ukrainian has been added and the support for Russian typesetting has been adapted to the package `'cyrillic'` to be released with the December 1998 release of $\text{\LaTeX 2}_{\epsilon}$.

5 Changes in Babel version 3.5

In `Babel` version 3.5 a lot of changes have been made when compared with the previous release. Here is a list of the most important ones:

- the selection of the language is delayed until `\begin{document}`, which means you must add appropriate `\selectlanguage` commands if you include `\hyphenation` lists in the preamble of your document.
- `babel` now has a `language` environment and a new command `\foreignlanguage`;
- the way active characters are dealt with is completely changed. They are called ‘shorthands’; one can have three levels of shorthands: on the user level, the language level, and on ‘system level’. A consequence of the new way of handling active characters is that they are now written to auxiliary files ‘`verbatim`’;

- A language change now also writes information in the `.aux` file, as the change might also affect typesetting the table of contents. The consequence is that an `.aux` file generated by a LaTeX format with babel preloaded gives errors when read with a LaTeX format without babel; but I think this probably doesn't occur;
- `babel` is now compatible with the `inputenc` and `fontenc` packages;
- the language definition files now have a new extension, `ldf`;
- the syntax of the file `language.dat` is extended to be compatible with the `french` package by Bernard Gaulle;
- each language definition file looks for a configuration file which has the same name, but the extension `.cfg`. It can contain any valid \LaTeX code.

6 The interface between the core of `babel` and the language definition files

In the core of the `babel` system, several macros are defined for use in language definition files. Their purpose is to make a new language known.

`\addlanguage` The macro `\addlanguage` is a non-outer version of the macro `\newlanguage`, defined in `plain.tex` version 3.x. For older versions of `plain.tex` and `lplain.tex` a substitute definition is used.

`\adddialect` The macro `\adddialect` can be used when two languages can (or must) use the same hyphenation patterns. This can also be useful for languages for which no patterns are preloaded in the format. In such cases the default behaviour of the `babel` system is to define this language as a 'dialect' of the language for which the patterns were loaded as `\language0`.

The language definition files must conform to a number of conventions, because these files have to fill in the gaps left by the common code in `babel.def`, i.e., the definitions of the macros that produce texts. Also the language-switching possibility which has been built into the `babel` system has its implications.

The following assumptions are made:

- Some of the language-specific definitions might be used by plain \TeX users, so the files have to be coded so that they can be read by both \LaTeX and plain \TeX . The current format can be checked by looking at the value of the macro `\fmtname`.
- The common part of the `babel` system redefines a number of macros and environments (defined previously in the document style) to put in the names of macros that replace the previously hard-wired texts. These macros have to be defined in the language definition files.
- The language definition files define five macros, used to activate and deactivate the language-specific definitions. These macros are `\<lang>hyphenmins`, `\<lang>captions`, `\<lang>date`, `\<lang>extras` and `\<lang>noextras`; where `\<lang>` is either the name of the language definition file or the name of the \LaTeX option that is to be used. These macros and their functions are discussed below.
- When a language definition file is loaded, it can define `\l@<lang>` to be a dialect of `\language0` when `\l@<lang>` is undefined.
- The language definition files can be read in the preamble of the document, but also in the middle of document processing. This means that they have to function independently of the current `\catcode` of the `@` sign.

<code>\providehyphenmins</code>	The macro <code>\providehyphenmins</code> should be used in the language definition files to set the <code>\lefthyphenmin</code> and <code>\righthyphenmin</code> . This macro will check whether these parameters were provided by the hyphenation file before it takes any action.
<code>\langhyphenmins</code>	The macro <code>\<lang>hyphenmins</code> is used to store the values of the <code>\lefthyphenmin</code> and <code>\righthyphenmin</code> .
<code>\captionslang</code>	The macro <code>\captions<lang></code> defines the macros that hold the texts to replace the original hard-wired texts.
<code>\datelang</code>	The macro <code>\date<lang></code> defines <code>\today</code> and
<code>\extraslang</code>	The macro <code>\extras<lang></code> contains all the extra definitions needed for a specific language.
<code>\noextraslang</code>	Because we want to let the user switch between languages, but we do not know what state \TeX might be in after the execution of <code>\extras<lang></code> , a macro that brings \TeX into a predefined state is needed. It will be no surprise that the name of this macro is <code>\noextras<lang></code> .
<code>\bbl@declare@ttribute</code>	This is a command to be used in the language definition files for declaring a language attribute. It takes three arguments: the name of the language, the attribute to be defined, and the code to be executed when the attribute is to be used.
<code>\main@language</code>	To postpone the activation of the definitions needed for a language until the beginning of a document, all language definition files should use <code>\main@language</code> instead of <code>\selectlanguage</code> . This will just store the name of the language, and the proper language will be activated at the start of the document.
<code>\ProvidesLanguage</code>	The macro <code>\ProvidesLanguage</code> should be used to identify the language definition files. Its syntax is similar to the syntax of the \LaTeX command <code>\ProvidesPackage</code> .
<code>\LdfInit</code>	The macro <code>\LdfInit</code> performs a couple of standard checks that must be made at the beginning of a language definition file, such as checking the category code of the <code>@</code> -sign, preventing the <code>.ldf</code> file from being processed twice, etc.
<code>\ldf@quit</code>	The macro <code>\ldf@quit</code> does work needed if a <code>.ldf</code> file was processed earlier. This includes resetting the category code of the <code>@</code> -sign, preparing the language to be activated at <code>\begin{document}</code> time, and ending the input stream.
<code>\ldf@finish</code>	The macro <code>\ldf@finish</code> does work needed at the end of each <code>.ldf</code> file. This includes resetting the category code of the <code>@</code> -sign, loading a local configuration file, and preparing the language to be activated at <code>\begin{document}</code> time.
<code>\loadlocalcfg</code>	After processing a language definition file, \LaTeX can be instructed to load a local configuration file. This file can, for instance, be used to add strings to <code>\captions<lang></code> to support local document classes. The user will be informed that this configuration file has been loaded. This macro is called by <code>\ldf@finish</code> .
<code>\substitutefontfamily</code>	This command takes three arguments, a font encoding and two font family names. It creates a font description file for the first font in the given encoding. This <code>.fd</code> file will instruct \LaTeX to use a font from the second family when a font from the first family in the given encoding seems to be needed.

6.1 Support for active characters

In quite a number of language definition files, active characters are introduced. To facilitate this, some support macros are provided.

<code>\initiate@active@char</code>	The internal macro <code>\initiate@active@char</code> is used in language definition files to instruct \LaTeX to give a character the category code ‘active’. When a character has been made active it will remain that way until the end of the document. Its definition may vary.
<code>\bbl@activate</code>	The command <code>\bbl@activate</code> is used to change the way an active character expands. <code>\bbl@activate</code> ‘switches on’ the active behaviour of the character. <code>\bbl@deactivate</code> lets the active character expand to its former (mostly) non-active self.
<code>\bbl@deactivate</code>	
<code>\declare@shorthand</code>	The macro <code>\declare@shorthand</code> is used to define the various shorthands. It takes three arguments: the name for the collection of shorthands this definition

belongs to; the character (sequence) that makes up the shorthand, i.e. `~` or `"a`; and the code to be executed when the shorthand is encountered.

`\bbl@add@special`
`\bbl@remove@special`

The `TEX`book states: “Plain `TEX` includes a macro called `\dospecials` that is essentially a set macro, representing the set of all characters that have a special category code.” [1, p. 380] It is used to set text ‘verbatim’. To make this work if more characters get a special category code, you have to add this character to the macro `\dospecials`. `LATEX` adds another macro called `\@sanitize` representing the same character set, but without the curly braces. The macros `\bbl@add@special⟨char⟩` and `\bbl@remove@special⟨char⟩` add and remove the character `⟨char⟩` to these two sets.

6.2 Support for saving macro definitions

Language definition files may want to *redefine* macros that already exist. Therefore a mechanism for saving (and restoring) the original definition of those macros is provided. We provide two macros for this¹.

`\babel@save`
`\babel@savevariable`

To save the current meaning of any control sequence, the macro `\babel@save` is provided. It takes one argument, `⟨csize⟩`, the control sequence for which the meaning has to be saved.

A second macro is provided to save the current value of a variable. In this context, anything that is allowed after the `\the` primitive is considered to be a variable. The macro takes one argument, the `⟨variable⟩`.

The effect of the preceding macros is to append a piece of code to the current definition of `\originalTeX`. When `\originalTeX` is expanded, this code restores the previous definition of the control sequence or the previous value of the variable.

6.3 Support for extending macros

`\addto` The macro `\addto{⟨control sequence⟩}{⟨TEX code⟩}` can be used to extend the definition of a macro. The macro need not be defined. This macro can, for instance, be used in adding instructions to a macro like `\extrasenglish`.

6.4 Macros common to a number of languages

`\allowhyphens`
`\set@low@box`
`\save@sf@q`
`\bbl@frenchspacing`
`\bbl@nonfrenchspacing`

In a couple of European languages compound words are used. This means that when `TEX` has to hyphenate such a compound word, it only does so at the ‘-’ that is used in such words. To allow hyphenation in the rest of such a compound word, the macro `\allowhyphens` can be used.

For some languages, quotes need to be lowered to the baseline. For this purpose the macro `\set@low@box` is available. It takes one argument and puts that argument in an `\hbox`, at the baseline. The result is available in `\box0` for further processing.

Sometimes it is necessary to preserve the `\spacefactor`. For this purpose the macro `\save@sf@q` is available. It takes one argument, saves the current spacefactor, executes the argument, and restores the spacefactor.

The commands `\bbl@frenchspacing` and `\bbl@nonfrenchspacing` can be used to properly switch French spacing on and off.

7 Compatibility with `german.sty`

The file `german.sty` has been one of the sources of inspiration for the `babel` system. Because of this I wanted to include `german.sty` in the `babel` system. To be able to do that I had to allow for one incompatibility: in the definition of the macro `\selectlanguage` in `german.sty` the argument is used as the `⟨number⟩` for an `\ifcase`. So in this case a call to `\selectlanguage` might look like `\selectlanguage{⟨german⟩}`.

¹This mechanism was introduced by Bernd Raichle.

In the definition of the macro `\selectlanguage` in `babel.def` the argument is used as a part of other macronames, so a call to `\selectlanguage` now looks like `\selectlanguage{german}`. Notice the absence of the escape character. As of version 3.1a of `babel` both syntaxes are allowed.

All other features of the original `german.sty` have been copied into a new file, called `germanb.sty`².

Although the `babel` system was developed to be used with L^AT_EX, some of the features implemented in the language definition files might be needed by plain T_EX users. Care has been taken that all files in the system can be processed by plain T_EX.

8 Compatibility with `ngerman.sty`

When used with the options `ngerman` or `naustrian`, `babel` will provide all features of the package `ngerman`. There is however one exception: The commands for special hyphenation of double consonants ("ff etc.) and ck ("ck), which are no longer required with the new German orthography, are undefined. With the `ngerman` package, however, these commands will generate appropriate warning messages only.

9 Compatibility with the french package

It has been reported to me that the package `french` by Bernard Gaulle (`gaulle@idris.fr`) works together with `babel`. On the other hand, it seems *not* to work well together with a lot of other packages. Therefore I have decided to no longer load `french.ldf` by default. Instead, when you want to use the package by Bernard Gaulle, you will have to request it specifically, by passing either `frenchle` or `frenchpro` as an option to `babel`.

10 Identification

The file `babel.sty`³ is meant for L^AT_EX 2_ε, therefor we make sure that the format file used is the right one.

`\ProvidesLanguage` The identification code for each file is something that was introduced in L^AT_EX 2_ε. When the command `\ProvidesFile` does not exist, a dummy definition is provided temporarily. For use in the language definition file the command `\ProvidesLanguage` is defined by `babel`.

```

10.1 <!*package>
10.2 \ifx\ProvidesFile\@undefined
10.3   \def\ProvidesFile#1[#2 #3 #4]{%
10.4     \wlog{File: #1 #4 #3 <#2>}}%
10.5 <*kernel & patterns>
10.6   \toks8{Babel <#3> and hyphenation patterns for }%
10.7 </kernel & patterns>
10.8   \let\ProvidesFile\@undefined
10.9   }
```

As an alternative for `\ProvidesFile` we define `\ProvidesLanguage` here to be used in the language definition files.

```

10.10 <*kernel>
10.11   \def\ProvidesLanguage#1[#2 #3 #4]{%
10.12     \wlog{Language: #1 #4 #3 <#2>}}%
10.13   }
10.14 \else
```

²The 'b' is added to the name to distinguish the file from Partls' file.

³The file described in this section is called `babel.dtx`, has version number v3.8l and was last revised on 2008/03/16.

In this case we save the original definition of `\ProvidesFile` in `\bbl@tempa` and restore it after we have stored the version of the file in `\toks8`.

```

10.15 <*kernel & patterns>
10.16   \let\bbl@tempa\ProvidesFile
10.17   \def\ProvidesFile#1[#2 #3 #4]{%
10.18     \toks8{Babel <#3> and hyphenation patterns for }%
10.19     \bbl@tempa#1[#2 #3 #4]%
10.20     \let\ProvidesFile\bbl@tempa}
10.21 </kernel & patterns>

```

When `\ProvidesFile` is defined we give `\ProvidesLanguage` a similar definition.

```

10.22   \def\ProvidesLanguage#1{%
10.23     \begingroup
10.24       \catcode'\ 10 %
10.25       \@makeother\/%
10.26       \@ifnextchar[%
10.27         {\@provideslanguage{#1}}{\@provideslanguage{#1}[]}}
10.28   \def\@provideslanguage#1[#2]{%
10.29     \wlog{Language: #1 #2}%
10.30     \expandafter\xdef\csname ver@#1.1df\endcsname{#2}%
10.31     \endgroup}
10.32 </kernel>
10.33 \fi
10.34 </!package>

```

Identify each file that is produced from this source file.

```

10.35 <+package>\ProvidesPackage{babel}
10.36 <+core>\ProvidesFile{babel.def}
10.37 <+kernel & patterns>\ProvidesFile{hyphen.cfg}
10.38 <+kernel&!patterns>\ProvidesFile{switch.def}
10.39 <+driver&!user>\ProvidesFile{babel.drv}
10.40 <+driver & user>\ProvidesFile{user.drv}
10.41                                     [2008/07/06 v3.81 %
10.42 <+package>       The Babel package]
10.43 <+core>          Babel common definitions]
10.44 <+kernel>        Babel language switching mechanism]
10.45 <+driver>]

```

11 The Package File

In order to make use of the features of $\text{\LaTeX} 2_{\epsilon}$, the `babel` system contains a package file, `babel.sty`. This file is loaded by the `\usepackage` command and defines all the language options known in the `babel` system. It also takes care of a number of compatibility issues with other packages.

11.1 Language options

```

11.1 <*package>
11.2 \ifx\LdfInit\@undefined\input babel.def\relax\fi

```

For all the languages supported we need to declare an option.

```

11.3 \DeclareOption{acadian}{\input{frenchb.1df}}
11.4 \DeclareOption{albanian}{\input{albanian.1df}}
11.5 \DeclareOption{afrikaans}{\input{dutch.1df}}
11.6 \DeclareOption{american}{\input{english.1df}}
11.7 \DeclareOption{australian}{\input{english.1df}}

Austrian is really a dialect of German.
11.8 \DeclareOption{austrian}{\input{germanb.1df}}

11.9 \DeclareOption{bahasa}{\input{bahasai.1df}}
11.10 \DeclareOption{indonesian}{\input{bahasai.1df}}
11.11 \DeclareOption{indon}{\input{bahasai.1df}}

```

```

11.12 \DeclareOption{bahasai}{\input{bahasai.ldf}}
11.13 \DeclareOption{malay}{\input{bahasam.ldf}}
11.14 \DeclareOption{meyalu}{\input{bahasam.ldf}}
11.15 \DeclareOption{bahasam}{\input{bahasam.ldf}}
11.16 \DeclareOption{basque}{\input{basque.ldf}}
11.17 \DeclareOption{brazil}{\input{portuges.ldf}}
11.18 \DeclareOption{brazilian}{\input{portuges.ldf}}
11.19 \DeclareOption{breton}{\input{breton.ldf}}
11.20 \DeclareOption{british}{\input{english.ldf}}
11.21 \DeclareOption{bulgarian}{\input{bulgarian.ldf}}
11.22 \DeclareOption{canadian}{\input{english.ldf}}
11.23 \DeclareOption{canadien}{\input{frenchb.ldf}}
11.24 \DeclareOption{catalan}{\input{catalan.ldf}}
11.25 \DeclareOption{croatian}{\input{croatian.ldf}}
11.26 \DeclareOption{czech}{\input{czech.ldf}}
11.27 \DeclareOption{danish}{\input{danish.ldf}}
11.28 \DeclareOption{dutch}{\input{dutch.ldf}}
11.29 \DeclareOption{english}{\input{english.ldf}}
11.30 \DeclareOption{esperanto}{\input{esperanto.ldf}}
11.31 \DeclareOption{estonian}{\input{estonian.ldf}}
11.32 \DeclareOption{finnish}{\input{finnish.ldf}}

```

The babel support for French used to be stored in `francais.ldf`; therefor the \LaTeX 2.09 option used to be `francais`. The hyphenation patterns may be loaded as either ‘french’ or as ‘francais’.

```

11.33 \DeclareOption{francais}{\input{frenchb.ldf}}
11.34 \DeclareOption{frenchb}{\input{frenchb.ldf}}

```

With \LaTeX 2_ε we can now also use the option `french` and still call the file `frenchb.ldf`.

```

11.35 \DeclareOption{french}{\input{frenchb.ldf}}%
11.36 \DeclareOption{galician}{\input{galician.ldf}}
11.37 \DeclareOption{german}{\input{germanb.ldf}}
11.38 \DeclareOption{germanb}{\input{germanb.ldf}}
11.39 \DeclareOption{greek}{\input{greek.ldf}}
11.40 \DeclareOption{polutonikogreek}{%
11.41   \input{greek.ldf}}%
11.42   \languageattribute{greek}{polutoniko}}
11.43 \DeclareOption{hebrew}{%
11.44   \input{rlbabel.def}}%
11.45   \input{hebrew.ldf}}

```

`hungarian` is just a synonym for `magyar`

```

11.46 \DeclareOption{hungarian}{\input{magyar.ldf}}
11.47 \DeclareOption{icelandic}{\input{icelandic.ldf}}
11.48 \DeclareOption{interlingua}{\input{interlingua.ldf}}
11.49 \DeclareOption{irish}{\input{irish.ldf}}
11.50 \DeclareOption{italian}{\input{italian.ldf}}
11.51 \DeclareOption{latin}{\input{latin.ldf}}
11.52 \DeclareOption{lowersorbian}{\input{lsorbian.ldf}}
11.53 %^A\DeclareOption{kannada}{\input{kannada.ldf}}
11.54 \DeclareOption{magyar}{\input{magyar.ldf}}
11.55 %^A\DeclareOption{nagari}{\input{nagari.ldf}}

```

‘New’ German orthography, including Austrian variant:

```

11.56 \DeclareOption{naustrian}{\input{ngermanb.ldf}}
11.57 \DeclareOption{newzealand}{\input{english.ldf}}
11.58 \DeclareOption{ngerman}{\input{ngermanb.ldf}}
11.59 \DeclareOption{norsk}{\input{norsk.ldf}}
11.60 \DeclareOption{samin}{\input{samin.ldf}}

```

For Norwegian two spelling variants are provided.

```

11.61 \DeclareOption{nynorsk}{\input{norsk.ldf}}

```



```

11.62 \DeclareOption{polish}{\input{polish.ldf}}
11.63 \DeclareOption{portuges}{\input{portuges.ldf}}
11.64 \DeclareOption{portuguese}{\input{portuges.ldf}}
11.65 \DeclareOption{romanian}{\input{romanian.ldf}}
11.66 \DeclareOption{russian}{\input{russianb.ldf}}

11.67 %^A\DeclareOption{sanskrit}{\input{sanskrit.ldf}}
11.68 \DeclareOption{scottish}{\input{scottish.ldf}}
11.69 \DeclareOption{serbian}{\input{serbian.ldf}}
11.70 \DeclareOption{slovak}{\input{slovak.ldf}}
11.71 \DeclareOption{slovene}{\input{slovene.ldf}}
11.72 \DeclareOption{spanish}{\input{spanish.ldf}}
11.73 \DeclareOption{swedish}{\input{swedish.ldf}}
11.74 %^A\DeclareOption{tamil}{\input{tamil.ldf}}
11.75 \DeclareOption{turkish}{\input{turkish.ldf}}
11.76 \DeclareOption{ukrainian}{\input{ukraineb.ldf}}
11.77 \DeclareOption{uppersorbian}{\input{usorbian.ldf}}
11.78 \DeclareOption{welsh}{\input{welsh.ldf}}

11.79 \DeclareOption{UKenglish}{\input{english.ldf}}
11.80 \DeclareOption{USenglish}{\input{english.ldf}}

```

For all those languages for which the option name is the same as the name of the language specific file we specify a default option, which tries to load the file specified. If this doesn't succeed an error is signalled.

```

11.81 \DeclareOption*{%
11.82   \InputIfFileExists{\CurrentOption.ldf}{}{%
11.83     \PackageError{babel}{%
11.84       Language definition file \CurrentOption.ldf not found}{%
11.85       Maybe you misspelled the language option?}}%
11.86 }

```

Another way to extend the list of 'known' options for babel is to create the file `bblopts.cfg` in which one can add option declarations.

```

11.87 \InputIfFileExists{bblopts.cfg}{%
11.88   \typeout{*****~J%
11.89     * Local config file bblopts.cfg used~J%
11.90     *}%
11.91 }{}

```

Apart from all the language options we also have a few options that influence the behaviour of language definition files.

The following options don't do anything themselves, they are just defined in order to make it possible for language definition files to check if one of them was specified by the user.

```

11.92 \DeclareOption{activeacute}{}
11.93 \DeclareOption{activegrave}{}

```

The next option tells babel to leave shorthand characters active at the end of processing the package. This is *not* the default as it can cause problems with other packages, but for those who want to use the shorthand characters in the preamble of their documents this can help.

```

11.94 \DeclareOption{KeepShorthandsActive}{}

```

The options have to be processed in the order in which the user specified them:

```

11.95 \ProcessOptions*

```

In order to catch the case where the user forgot to specify a language we check whether `\bbl@main@language`, has become defined. If not, no language has been loaded and an error message is displayed.

```

11.96 \ifx\bbl@main@language\@undefined
11.97   \PackageError{babel}{%
11.98     You haven't specified a language option}{%
11.99     You need to specify a language, either as a global
11.100    option\MessageBreak
11.101    or as an optional argument to the \string\usepackage\space
11.102    command; \MessageBreak

```

11.103 You shouldn't try to proceed from here, type x to quit.}

To prevent undefined command errors when the user insists on continuing we load `babel.def` here. He should expect more errors though.

11.104

11.105 \fi

`\substitutefontfamily` The command `\substitutefontfamily` creates an `.fd` file on the fly. The first argument is an encoding mnemonic, the second and third arguments are font family names.

```
11.106 \def\substitutefontfamily#1#2#3{%
11.107   \lowercase{\immediate\openout15=#1#2.fd\relax}%
11.108   \immediate\write15{%
11.109     \string\ProvidesFile{#1#2.fd}%
11.110     [the\year/\two@digits{\the\month}/\two@digits{\the\day}
11.111     \space generated font description file]^^J
11.112     \string\DeclareFontFamily{#1}{#2}{-}^^J
11.113     \string\DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{-}^^J
11.114     \string\DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{-}^^J
11.115     \string\DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{-}^^J
11.116     \string\DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{-}^^J
11.117     \string\DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/bx/n}{-}^^J
11.118     \string\DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/bx/it}{-}^^J
11.119     \string\DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/bx/sl}{-}^^J
11.120     \string\DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/bx/sc}{-}^^J
11.121   }%
11.122   \closeout15
11.123 }
```

This command should only be used in the preamble of a document.

11.124 \@onlypreamble\substitutefontfamily

11.125 \end{package}

12 The Kernel of Babel

The kernel of the `babel` system is stored in either `hyphen.cfg` or `switch.def` and `babel.def`. The file `hyphen.cfg` is a file that can be loaded into the format, which is necessary when you want to be able to switch hyphenation patterns. The file `babel.def` contains some \TeX code that can be read in at run time. When `babel.def` is loaded it checks if `hyphen.cfg` is in the format; if not the file `switch.def` is loaded.

Because plain \TeX users might want to use some of the features of the `babel` system too, care has to be taken that plain \TeX can process the files. For this reason the current format will have to be checked in a number of places. Some of the code below is common to plain \TeX and \LaTeX , some of it is for the \LaTeX case only.

When the command `\AtBeginDocument` doesn't exist we assume that we are dealing with a plain-based format. In that case the file `plain.def` is needed.

```
12.1 (*kernel | core)
12.2 \ifx\AtBeginDocument\@undefined
```

But we need to use the second part of `plain.def` (when we load it from `switch.def`) which we can do by defining `\adddialect`.

```
12.3 (kernel&!patterns) \def\adddialect{}
12.4   \input plain.def\relax
12.5 \fi
12.6 \end{kernel | core}
```

Check the presence of the command `\iflanguage`, if it is undefined read the file `switch.def`.

```
12.7 (*core)
```

```

12.8 \ifx\iflanguage\@undefined
12.9 \input switch.def\relax
12.10 \fi
12.11 \</core>

```

12.1 Encoding issues (part 1)

The first thing we need to do is to determine, at `\begin{document}`, which latin fontencoding to use.

`\latinencoding` When text is being typeset in an encoding other than ‘latin’ (OT1 or T1), it would be nice to still have Roman numerals come out in the Latin encoding. So we first assume that the current encoding at the end of processing the package is the Latin encoding.

```

12.12 \<core>
12.13 \AtEndOfPackage{\edef\latinencoding{\cf@encoding}}

```

But this might be overruled with a later loading of the package `fontenc`. Therefore we check at the execution of `\begin{document}` whether it was loaded with the T1 option. The normal way to do this (using `\@ifpackageloaded`) is disabled for this package. Now we have to revert to parsing the internal macro `\@filelist` which contains all the filenames loaded.

```

12.14 \AtBeginDocument{%
12.15   \gdef\latinencoding{OT1}%
12.16   \ifx\cf@encoding\bbl@t@one
12.17     \xdef\latinencoding{\bbl@t@one}%
12.18   \else
12.19     \@ifl@aded{def}{t1enc}{\xdef\latinencoding{\bbl@t@one}}{}%
12.20   \fi
12.21 }

```

`\latintext` Then we can define the command `\latintext` which is a declarative switch to a latin font-encoding.

```

12.22 \DeclareRobustCommand{\latintext}{%
12.23   \fontencoding{\latinencoding}\selectfont
12.24   \def\encodingdefault{\latinencoding}}

```

`\textlatin` This command takes an argument which is then typeset using the requested font encoding. In order to avoid many encoding switches it operates in a local scope.

```

12.25 \ifx\@undefined\DeclareTextFontCommand
12.26   \DeclareRobustCommand{\textlatin}[1]{\leavevmode{\latintext #1}}
12.27 \else
12.28   \DeclareTextFontCommand{\textlatin}{\latintext}
12.29 \fi
12.30 \</core>

```

We also need to redefine a number of commands to ensure that the right font encoding is used, but this can’t be done before `babel.def` is loaded.

12.2 Multiple languages

With T_EX version 3.0 it has become possible to load hyphenation patterns for more than one language. This means that some extra administration has to be taken care of. The user has to know for which languages patterns have been loaded, and what values of `\language` have been used.

Some discussion has been going on in the T_EX world about how to use `\language`. Some have suggested to set a fixed standard, i. e., patterns for each language should *always* be loaded in the same location. It has also been suggested to use the ISO list for this purpose. Others have pointed out that the ISO list contains more than 256 languages, which have *not* been numbered consecutively.

I think the best way to use `\language`, is to use it dynamically. This code implements an algorithm to do so. It uses an external file in which the person who maintains a \TeX environment has to record for which languages he has hyphenation patterns *and* in which files these are stored⁴. When hyphenation exceptions are stored in a separate file this can be indicated by naming that file *after* the file with the hyphenation patterns.

This “configuration file” can contain empty lines and comments, as well as lines which start with an equals (=) sign. Such a line will instruct \LaTeX that the hyphenation patterns just processed have to be known under an alternative name. Here is an example:

```
% File      : language.dat
% Purpose   : tell iniTeX what files with patterns to load.
english    english.hyphenations
=british

dutch      hyphen.dutch exceptions.dutch % Nederlands
german     hyphen.ger
```

As the file `switch.def` needs to be read only once, we check whether it was read before. If it was, the command `\iflanguage` is already defined, so we can stop processing.

```
12.31 (*kernel)
12.32 (*!patterns)
12.33 \expandafter\ifx\csname iflanguage\endcsname\relax \else
12.34 \expandafter\endinput
12.35 \fi
12.36 \!/patterns)
```

\language Plain \TeX version 3.0 provides the primitive `\language` that is used to store the current language. When used with a pre-3.0 version this function has to be implemented by allocating a counter.

```
12.37 \ifx\language\@undefined
12.38   \csname newcount\endcsname\language
12.39 \fi
```

\last@language Another counter is used to store the last language defined. For pre-3.0 formats an extra counter has to be allocated,

```
12.40 \ifx\newlanguage\@undefined
12.41   \csname newcount\endcsname\last@language
      plain  $\text{\TeX}$  version 3.0 uses \count 19 for this purpose.
12.42 \else
12.43   \countdef\last@language=19
12.44 \fi
```

\addlanguage To add languages to \TeX ’s memory plain \TeX version 3.0 supplies `\newlanguage`, in a pre-3.0 environment a similar macro has to be provided. For both cases a new macro is defined here, because the original `\newlanguage` was defined to be `\outer`.

For a format based on plain version 2.x, the definition of `\newlanguage` can not be copied because `\count 19` is used for other purposes in these formats. Therefor `\addlanguage` is defined using a definition based on the macros used to define `\newlanguage` in plain \TeX version 3.0.

```
12.45 \ifx\newlanguage\@undefined
12.46   \def\addlanguage#1{%
12.47     \global\advance\last@language \ne
```

⁴This is because different operating systems sometimes use *very* different file-naming conventions.

```

12.48 \ifnum\last@language<\@ccclvi
12.49 \else
12.50 \errmessage{No room for a new \string\language!}%
12.51 \fi
12.52 \global\chardef#1\last@language
12.53 \wlog{\string#1 = \string\language\the\last@language}}

```

For formats based on plain version 3.0 the definition of `\newlanguage` can be simply copied, removing `\outer`.

```

12.54 \else
12.55 \def\addlanguage{\alloc@9\language\chardef\@ccclvi}
12.56 \fi

```

`\adddialect` The macro `\adddialect` can be used to add the name of a dialect or variant language, for which an already defined hyphenation table can be used.

```

12.57 \def\adddialect#1#2{%
12.58 \global\chardef#1#2\relax
12.59 \wlog{\string#1 = a dialect from \string\language#2}}

```

`\iflanguage` Users might want to test (in a private package for instance) which language is currently active. For this we provide a test macro, `\iflanguage`, that has three arguments. It checks whether the first argument is a known language. If so, it compares the first argument with the value of `\language`. Then, depending on the result of the comparison, it executes either the second or the third argument.

```

12.60 \def\iflanguage#1{%
12.61 \expandafter\ifx\csname l@#1\endcsname\relax
12.62 \@nolanerr{#1}%
12.63 \else
12.64 \bbl@afterfi{\ifnum\csname l@#1\endcsname=\language
12.65 \expandafter\@firstoftwo
12.66 \else
12.67 \expandafter\@secondoftwo
12.68 \fi}%
12.69 \fi}

```

`\selectlanguage` The macro `\selectlanguage` checks whether the language is already defined before it performs its actual task, which is to update `\language` and activate language-specific definitions.

To allow the call of `\selectlanguage` either with a control sequence name or with a simple string as argument, we have to use a trick to delete the optional escape character.

To convert a control sequence to a string, we use the `\string` primitive. Next we have to look at the first character of this string and compare it with the escape character. Because this escape character can be changed by setting the internal integer `\escapechar` to a character number, we have to compare this number with the character of the string. To do this we have to use TeX's backquote notation to specify the character as a number.

If the first character of the `\string`'ed argument is the current escape character, the comparison has stripped this character and the rest in the 'then' part consists of the rest of the control sequence name. Otherwise we know that either the argument is not a control sequence or `\escapechar` is set to a value outside of the character range 0–255.

If the user gives an empty argument, we provide a default argument for `\string`. This argument should expand to nothing.

```

12.70 \edef\selectlanguage{%
12.71 \noexpand\protect
12.72 \expandafter\noexpand\csname selectlanguage \endcsname
12.73 }

```

Because the command `\selectlanguage` could be used in a moving argument it expands to `\protect\selectlanguage␣`. Therefor, we have to make sure that a macro `\protect` exists. If it doesn't it is `\let` to `\relax`.

```
12.74 \ifx\@undefined\protect\let\protect\relax\fi
```

As L^AT_EX 2.09 writes to files *expanded* whereas L^AT_EX 2_ε takes care *not* to expand the arguments of `\write` statements we need to be a bit clever about the way we add information to `.aux` files. Therefor we introduce the macro `\xstring` which should expand to the right amount of `\string`'s.

```
12.75 \ifx\documentclass\@undefined
12.76   \def\xstring{\string\string\string}
12.77 \else
12.78   \let\xstring\string
12.79 \fi
```

Since version 3.5 `babel` writes entries to the auxiliary files in order to typeset table of contents etc. in the correct language environment.

`\bbl@pop@language` But when the language change happens *inside* a group the end of the group doesn't write anything to the auxiliary files. Therefor we need T_EX's `aftergroup` mechanism to help us. The command `\aftergroup` stores the token immediately following it to be executed when the current group is closed. So we define a temporary control sequence `\bbl@pop@language` to be executed at the end of the group. It calls `\bbl@set@language` with the name of the current language as its argument.

`\bbl@language@stack` The previous solution works for one level of nesting groups, but as soon as more levels are used it is no longer adequate. For that case we need to keep track of the nested languages using a stack mechanism. This stack is called `\bbl@language@stack` and initially empty.

```
12.80 \xdef\bbl@language@stack{}
```

When using a stack we need a mechanism to push an element on the stack and to retrieve the information afterwards.

`\bbl@push@language` The stack is simply a list of languagenames, separated with a '+' sign; the push
`\bbl@pop@language` function can be simple:

```
12.81 \def\bbl@push@language{%
12.82   \xdef\bbl@language@stack{\language+\bbl@language@stack}%
12.83 }
```

Retrieving information from the stack is a little bit less simple, as we need to remove the element from the stack while storing it in the macro `\language`. For this we first define a helper function.

`\bbl@pop@lang` This macro stores its first element (which is delimited by the '+'-sign) in `\language` and stores the rest of the string (delimited by '-') in its third argument.

```
12.84 \def\bbl@pop@lang#1+#2-#3{%
12.85   \def\language{#1}\xdef#3{#2}%
12.86 }
```

The reason for the somewhat weird arrangement of arguments to the helper function is the fact it is called in the following way:

```
12.87 \def\bbl@pop@language{%
12.88   \expandafter\bbl@pop@lang\bbl@language@stack-\bbl@language@stack
```

This means that before `\bbl@pop@lang` is executed T_EX first *expands* the stack, stored in `\bbl@language@stack`. The result of that is that the argument string of `\bbl@pop@lang` contains one or more language names, each followed by a '+'-sign (zero language names won't occur as this macro will only be called after something has been pushed on the stack) followed by the '-'-sign and finally the reference to the stack.

```
12.89 $$
```

```

12.90 \expandafter\babel@set@language\expandafter{\language}%
12.91 }

```

Once the name of the previous language is retrieved from the stack, it is fed to `\babel@set@language` to do the actual work of switching everything that needs switching.

```

12.92 \expandafter\def\csname selectlanguage \endcsname#1{%
12.93 \babel@push@language
12.94 \aftergroup\babel@pop@language
12.95 \babel@set@language{#1}}

```

`\babel@set@language` The macro `\babel@set@language` takes care of switching the language environment *and* of writing entries on the auxiliary files.

```

12.96 \def\babel@set@language#1{%
12.97 \edef\language{%
12.98 \ifnum\escapechar=\expandafter'\string#1\@empty
12.99 \else \string#1\@empty\fi}%
12.100 \select@language{\language}%

```

We also write a command to change the current language in the auxiliary files.

```

12.101 \if@filesw
12.102 \protected@write\@auxout{}\string\select@language{\language}}%
12.103 \addtocontents{toc}{\xstring\select@language{\language}}%
12.104 \addtocontents{lof}{\xstring\select@language{\language}}%
12.105 \addtocontents{lot}{\xstring\select@language{\language}}%
12.106 \fi}

```

First, check if the user asks for a known language. If so, update the value of `\language` and call `\originalTeX` to bring $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ in a certain pre-defined state.

```

12.107 \def\select@language#1{%
12.108 \expandafter\ifx\csname l@#1\endcsname\relax
12.109 \@nolanerr{#1}%
12.110 \else
12.111 \expandafter\ifx\csname date#1\endcsname\relax
12.112 \@noopterr{#1}%
12.113 \else
12.114 \babel@patterns{\language}%
12.115 \originalTeX

```

The name of the language is stored in the control sequence `\language`. The contents of this control sequence could be tested in the following way:

```

\edef\tmp{\string english}
\ifx\language\tmp
...
\else
...
\fi

```

The construction with `\string` is necessary because `\language` returns the name with characters of category code 12 (other). Then we have to redefine `\originalTeX` to compensate for the things that have been activated. To save memory space for the macro definition of `\originalTeX`, we construct the control sequence name for the `\noextras⟨lang⟩` command at definition time by expanding the `\csname` primitive.

```

12.116 \expandafter\def\expandafter\originalTeX
12.117 \expandafter{\csname noextras#1\endcsname
12.118 \let\originalTeX\@empty}%
12.119 \languageshorthands{none}%
12.120 \babel@beginsave

```

Now activate the language-specific definitions. This is done by constructing the names of three macros by concatenating three words with the argument of `\selectlanguage`, and calling these macros.

```
12.121 \csname captions#1\endcsname
12.122 \csname date#1\endcsname
12.123 \csname extras#1\endcsname\relax
```

The switching of the values of `\lefthyphenmin` and `\righthyphenmin` is somewhat different. First we save their current values, then we check if `\langle lang \rangle hyphenmins` is defined. If it is not, we set default values (2 and 3), otherwise the values in `\langle lang \rangle hyphenmins` will be used.

```
12.124 \babel@savevariable\lefthyphenmin
12.125 \babel@savevariable\righthyphenmin
12.126 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
12.127 \set@hyphenmins\tw@\thr@@\relax
12.128 \else
12.129 \expandafter\expandafter\expandafter\set@hyphenmins
12.130 \csname #1hyphenmins\endcsname\relax
12.131 \fi
12.132 \fi
12.133 \fi}
```

otherlanguage The `otherlanguage` environment can be used as an alternative to using the `\selectlanguage` declarative command. When you are typesetting a document which mixes left-to-right and right-to-left typesetting you have to use this environment in order to let things work as you expect them to.

The first thing this environment does is store the name of the language in `\language`; it then calls `\selectlanguage` to switch on everything that is needed for this language. The `\ignorespaces` command is necessary to hide the environment when it is entered in horizontal mode.

```
12.134 \long\def\otherlanguage#1{%
12.135 \csname selectlanguage \endcsname{#1}%
12.136 \ignorespaces
12.137 }
```

The `\endotherlanguage` part of the environment calls `\originalTeX` to restore (most of) the settings and tries to hide itself when it is called in horizontal mode.

```
12.138 \long\def\endotherlanguage{%
12.139 \originalTeX
12.140 \global\@ignoretrue\ignorespaces
12.141 }
```

otherlanguage* The `otherlanguage` environment is meant to be used when a large part of text from a different language needs to be typeset, but without changing the translation of words such as ‘figure’.

This environment makes use of `\foreign@language`.

```
12.142 \expandafter\def\csname otherlanguage*\endcsname#1{%
12.143 \foreign@language{#1}%
12.144 }
```

At the end of the environment we need to switch off the extra definitions. The grouping mechanism of the environment will take care of resetting the correct hyphenation rules.

```
12.145 \expandafter\def\csname endotherlanguage*\endcsname{%
12.146 \csname noextras\language\endcsname
12.147 }
```

\foreignlanguage The `\foreignlanguage` command is another substitute for the `\selectlanguage` command. This command takes two arguments, the first argument is the name of the language to use for typesetting the text specified in the second argument.

Unlike `\selectlanguage` this command doesn’t switch *everything*, it only switches the hyphenation rules and the extra definitions for the language specified.

It does this within a group and assumes the `\extras⟨lang⟩` command doesn't make any `\global` changes. The coding is very similar to part of `\selectlanguage`.

```

12.148 \def\foreignlanguage{\protect\csname foreignlanguage \endcsname}
12.149 \expandafter\def\csname foreignlanguage \endcsname#1#2{%
12.150   \begingroup
12.151     \originalTeX
12.152     \foreign@language{#1}%
12.153     #2%
12.154     \csname noextras#1\endcsname
12.155   \endgroup
12.156 }

```

`\foreign@language` This macro does the work for `\foreignlanguage` and the `otherlanguage*` environment.

```

12.157 \def\foreign@language#1{%

```

First we need to store the name of the language and check that it is a known language.

```

12.158   \def\language#1{%
12.159   \expandafter\ifx\csname l@#1\endcsname\relax
12.160     \@nolanerr{#1}%
12.161   \else

```

If it is we can select the proper hyphenation table and switch on the extra definitions for this language.

```

12.162     \bbl@patterns{\language}%
12.163     \languageshorthands{none}%

```

Then we set the left- and right hyphenmin variables.

```

12.164     \csname extras#1\endcsname
12.165     \expandafter\ifx\csname #1hyphenmins\endcsname\relax
12.166       \set@hyphenmins\tw@\thr@@\relax
12.167     \else
12.168       \expandafter\expandafter\expandafter\set@hyphenmins
12.169         \csname #1hyphenmins\endcsname\relax
12.170     \fi
12.171   \fi
12.172 }

```

`\bbl@patterns` This macro selects the hyphenation patterns by changing the `\language` register. If special hyphenation patterns are available specifically for the current font encoding, use them instead of the default.

```

12.173 \def\bbl@patterns#1{%
12.174   \language=\expandafter\ifx\csname l@#1:f@encoding\endcsname\relax
12.175     \csname l@#1\endcsname
12.176   \else
12.177     \csname l@#1:f@encoding\endcsname
12.178   \fi\relax
12.179 }

```

`hyphenrules` The environment `hyphenrules` can be used to select *just* the hyphenation rules. This environment does *not* change `\language` and when the hyphenation rules specified were not loaded it has no effect.

```

12.180 \def\hyphenrules#1{%
12.181   \expandafter\ifx\csname l@#1\endcsname\@undefined
12.182     \@nolanerr{#1}%
12.183   \else
12.184     \bbl@patterns{#1}%
12.185     \languageshorthands{none}%
12.186     \expandafter\ifx\csname #1hyphenmins\endcsname\relax
12.187       \set@hyphenmins\tw@\thr@@\relax
12.188     \else

```

```

12.189      \expandafter\expandafter\expandafter\set@hyphenmins
12.190      \csname #1hyphenmins\endcsname\relax
12.191      \fi
12.192  \fi
12.193  }
12.194 \def\endhyphenrules{}
```

\providehyphenmins The macro `\providehyphenmins` should be used in the language definition files to provide a *default* setting for the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`. If the macro `\langhyphenmins` is already defined this command has no effect.

```

12.195 \def\providehyphenmins#1#2{%
12.196   \expandafter\ifx\csname #1hyphenmins\endcsname\relax
12.197     \@namedef{#1hyphenmins}{#2}%
12.198   \fi}
```

\set@hyphenmins This macro sets the values of `\lefthyphenmin` and `\righthyphenmin`. It expects two values as its argument.

```

12.199 \def\set@hyphenmins#1#2{\lefthyphenmin#1\righthyphenmin#2}
```

\LdfInit This macro is defined in two versions. The first version is to be part of the ‘kernel’ of `babel`, ie. the part that is loaded in the format; the second version is defined in `babel.def`. The version in the format just checks the category code of the ampersand and then loads `babel.def`.

```

12.200 \def\LdfInit{%
12.201   \chardef\atcatcode=\catcode'\@
12.202   \catcode'\@=11\relax
12.203   \input babel.def\relax

   The category code of the ampersand is restored and the macro calls itself again
   with the new definition from babel.def

12.204   \catcode'\@=\atcatcode \let\atcatcode\relax
12.205   \LdfInit}
12.206 \kernel}
```

The second version of this macro takes two arguments. The first argument is the name of the language that will be defined in the language definition file; the second argument is either a control sequence or a string from which a control sequence should be constructed. The existence of the control sequence indicates that the file has been processed before.

At the start of processing a language definition file we always check the category code of the ampersand. We make sure that it is a ‘letter’ during the processing of the file.

```

12.207 (*core)
12.208 \def\LdfInit#1#2{%
12.209   \chardef\atcatcode=\catcode'\@
12.210   \catcode'\@=11\relax
```

Another character that needs to have the correct category code during processing of language definition files is the equals sign, ‘=’, because it is sometimes used in constructions with the `\let` primitive. Therefore we store its current catcode and restore it later on.

```

12.211   \chardef\eqcatcode=\catcode'\=
12.212   \catcode'\==12\relax
```

Now we check whether we should perhaps stop the processing of this file. To do this we first need to check whether the second argument that is passed to `\LdfInit` is a control sequence. We do that by looking at the first token after passing `#2` through `string`. When it is equal to `\@backslashchar` we are dealing with a control sequence which we can compare with `\@undefined`.

```

12.213   \let\bbl@tempa\relax
12.214   \expandafter\if\expandafter\@backslashchar
```

```

12.215         \expandafter\@car\string#2\@nil
12.216     \ifx#2\@undefined
12.217     \else

```

If so, we call `\ldf@quit` (but after the end of this `\if` construction) to set the main language, restore the category code of the `@`-sign and call `\endinput`.

```

12.218         \def\bbl@tempa{\ldf@quit{#1}}
12.219     \fi
12.220 \else

```

When `#2` was *not* a control sequence we construct one and compare it with `\relax`.

```

12.221     \expandafter\ifx\csname#2\endcsname\relax
12.222     \else
12.223         \def\bbl@tempa{\ldf@quit{#1}}
12.224     \fi
12.225 \fi
12.226 \bbl@tempa

```

Finally we check `\originalTeX`.

```

12.227 \ifx\originalTeX\@undefined
12.228     \let\originalTeX\@empty
12.229 \else
12.230     \originalTeX
12.231 \fi}

```

`\ldf@quit` This macro interrupts the processing of a language definition file.

```

12.232 \def\ldf@quit#1{%
12.233     \expandafter\main@language\expandafter{#1}%
12.234     \catcode'\@=\atcatcode \let\atcatcode\relax
12.235     \catcode'\==\eqcatcode \let\eqcatcode\relax
12.236     \endinput
12.237 }

```

`\ldf@finish` This macro takes one argument. It is the name of the language that was defined in the language definition file.

We load the local configuration file if one is present, we set the main language (taking into account that the argument might be a control sequence that needs to be expanded) and reset the category code of the `@`-sign.

```

12.238 \def\ldf@finish#1{%
12.239     \loadlocalcfg{#1}%
12.240     \expandafter\main@language\expandafter{#1}%
12.241     \catcode'\@=\atcatcode \let\atcatcode\relax
12.242     \catcode'\==\eqcatcode \let\eqcatcode\relax
12.243 }

```

After the preamble of the document the commands `\LdfInit`, `\ldf@quit` and `\ldf@finish` are no longer needed. Therefor they are turned into warning messages in `LATEX`.

```

12.244 \@onlypreamble\LdfInit
12.245 \@onlypreamble\ldf@quit
12.246 \@onlypreamble\ldf@finish

```

`\main@language` This command should be used in the various language definition files. It stores its argument in `\bbl@main@language`; to be used to switch to the correct language at the beginning of the document.

```

12.247 \def\main@language#1{%
12.248     \def\bbl@main@language{#1}%
12.249     \let\language\main@language
12.250     \bbl@patterns{\language}%
12.251 }

```

The default is to use English as the main language.

```
12.252 \ifx\l@english\@undefined
12.253   \let\l@english\z@
12.254 \fi
12.255 \main@language{english}
```

We also have to make sure that some code gets executed at the beginning of the document.

```
12.256 \AtBeginDocument{%
12.257   \expandafter\selectlanguage\expandafter{\bbl@main@language}}
12.258 \</core>
```

`\originalTeX` The macro `\originalTeX` should be known to \TeX at this moment. As it has to be expandable we `\let` it to `\@empty` instead of `\relax`.

```
12.259 (*kernel)
12.260 \ifx\originalTeX\@undefined\let\originalTeX\@empty\fi
```

Because this part of the code can be included in a format, we make sure that the macro which initialises the save mechanism, `\babel@beginsave`, is not considered to be undefined.

```
12.261 \ifx\babel@beginsave\@undefined\let\babel@beginsave\relax\fi
```

`\@nolanerr` The babel package will signal an error when a documents tries to select a language that hasn't been defined earlier. When a user selects a language for which no hyphenation patterns were loaded into the format he will be given a warning about that fact. We revert to the patterns for `\language=0` in that case. In most formats that will be (US)english, but it might also be empty.

`\@noopterr` When the package was loaded without options not everything will work as expected. An error message is issued in that case.

When the format knows about `\PackageError` it must be $\text{\LaTeX 2}_{\epsilon}$, so we can safely use its error handling interface. Otherwise we'll have to 'keep it simple'.

```
12.262 \ifx\PackageError\@undefined
12.263   \def\@nolanerr#1{%
12.264     \errhelp{Your command will be ignored, type <return> to proceed}%
12.265     \errmessage{You haven't defined the language #1\space yet}}
12.266   \def\@nopatterns#1{%
12.267     \message{No hyphenation patterns were loaded for}%
12.268     \message{the language '#1'}%
12.269     \message{I will use the patterns loaded for \string\language=0
12.270       instead}}
12.271   \def\@noopterr#1{%
12.272     \errmessage{The option #1 was not specified in \string\usepackage}
12.273     \errhelp{You may continue, but expect unexpected results}}
12.274   \def\@activated#1{%
12.275     \wlog{Package babel Info: Making #1 an active character}}
12.276 \else
12.277   \newcommand*{\@nolanerr}[1]{%
12.278     \PackageError{babel}%
12.279       {You haven't defined the language #1\space yet}%
12.280       {Your command will be ignored, type <return> to proceed}}
12.281   \newcommand*{\@nopatterns}[1]{%
12.282     \PackageWarningNoLine{babel}%
12.283       {No hyphenation patterns were loaded for\MessageBreak
12.284         the language '#1'\MessageBreak
12.285         I will use the patterns loaded for \string\language=0
12.286         instead}}
12.287   \newcommand*{\@noopterr}[1]{%
12.288     \PackageError{babel}%
12.289       {You haven't loaded the option #1\space yet}%
12.290       {You may proceed, but expect unexpected results}}
12.291   \newcommand*{\@activated}[1]{%
```

```

12.292 \PackageInfo{babel}{%
12.293 Making #1 an active character}}
12.294 \fi

```

The following code is meant to be read by `iniTEX` because it should instruct `TEX` to read hyphenation patterns. To this end the `docstrip` option `patterns` can be used to include this code in the file `hyphen.cfg`.

```
12.295 (*patterns)
```

`\process@line` Each line in the file `language.dat` is processed by `\process@line` after it is read. The first thing this macro does is to check whether the line starts with `=`. When the first token of a line is an `=`, the macro `\process@synonym` is called; otherwise the macro `\process@language` will continue.

```

12.296 \def\process@line#1#2 #3/{%
12.297 \ifx=#1
12.298 \process@synonym#2 /
12.299 \else
12.300 \process@language#1#2 #3/%
12.301 \fi
12.302 }

```

`\process@synonym` This macro takes care of the lines which start with an `=`. It needs an empty token register to begin with.

```

12.303 \toks@{}
12.304 \def\process@synonym#1 /{%
12.305 \ifnum\last@language=\m@ne

```

When no languages have been loaded yet, the name following the `=` will be a synonym for hyphenation register 0.

```

12.306 \expandafter\chardef\csname l@#1\endcsname0\relax
12.307 \wlog{\string\l@#1=\string\language0}

```

As no hyphenation patterns are read in yet, we can not yet set the `hyphenmin` parameters. Therefor a commands to do so is stored in a token register and executed when the first pattern file has been processed.

```

12.308 \toks@\expandafter{\the\toks@
12.309 \expandafter\let\csname #1hyphenmins\expandafter\endcsname
12.310 \csname\language\hyphenmins\endcsname}%
12.311 \else

```

Otherwise the name will be a synonym for the language loaded last.

```

12.312 \expandafter\chardef\csname l@#1\endcsname\last@language
12.313 \wlog{\string\l@#1=\string\language\the\last@language}

```

We also need to copy the `hyphenmin` parameters for the synonym.

```

12.314 \expandafter\let\csname #1hyphenmins\expandafter\endcsname
12.315 \csname\language\hyphenmins\endcsname
12.316 \fi
12.317 }

```

`\process@language` The macro `\process@language` is used to process a non-empty line from the ‘configuration file’. It has three arguments, each delimited by white space. The third argument is optional, so a `/` character is expected to delimit the last argument. The first argument is the ‘name’ of a language; the second is the name of the file that contains the patterns. The optional third argument is the name of a file containing hyphenation exceptions.

The first thing to do is call `\addlanguage` to allocate a pattern register and to make that register ‘active’.

```

12.318 \def\process@language#1 #2 #3/{%
12.319 \expandafter\addlanguage\csname l@#1\endcsname
12.320 \expandafter\language\csname l@#1\endcsname
12.321 \def\language{#1}%

```

Then the ‘name’ of the language that will be loaded now is added to the token register `\toks8`. and finally the pattern file is read.

```
12.322 \global\toks8\expandafter{\the\toks8#1,}%
```

For some hyphenation patterns it is needed to load them with a specific font encoding selected. This can be specified in the file `language.dat` by adding for instance ‘:T1’ to the name of the language. The macro `\bbl@get@enc` extracts the font encoding from the language name and stores it in `\bbl@hyph@enc`.

```
12.323 \begingroup
12.324 \bbl@get@enc#1:\@@@
12.325 \ifx\bbl@hyph@enc\@empty
12.326 \else
12.327 \fontencoding{\bbl@hyph@enc}\selectfont
12.328 \fi
```

Some pattern files contain assignments to `\lefthyphenmin` and `\righthyphenmin`. \TeX does not keep track of these assignments. Therefore we try to detect such assignments and store them in the `\(lang)hyphenmins` macro. When no assignments were made we provide a default setting.

```
12.329 \lefthyphenmin\m@ne
```

Some pattern files contain changes to the `\lccode` en `\uccode` arrays. Such changes should remain local to the language; therefore we process the pattern file in a group; the `\patterns` command acts globally so its effect will be remembered.

```
12.330 \input #2\relax
```

Now we globally store the settings of `\lefthyphenmin` and `\righthyphenmin` and close the group.

```
12.331 \ifnum\lefthyphenmin=\m@ne
12.332 \else
12.333 \expandafter\xdef\csname #1hyphenmins\endcsname%
12.334 \the\lefthyphenmin\the\righthyphenmin}%
12.335 \fi
12.336 \endgroup
```

If the counter `\language` is still equal to zero we set the hyphenmin parameters to the values for the language loaded on pattern register 0.

```
12.337 \ifnum\the\language=\z@
12.338 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
12.339 \set@hyphenmins\tw@\thr@@\relax
12.340 \else
12.341 \expandafter\expandafter\expandafter\set@hyphenmins
12.342 \csname #1hyphenmins\endcsname
12.343 \fi
```

Now execute the contents of token register zero as it may contain commands which set the hyphenmin parameters for synonyms that were defined before the first pattern file is read in.

```
12.344 \the\toks@
12.345 \fi
```

Empty the token register after use.

```
12.346 \toks@{}}%
```

When the hyphenation patterns have been processed we need to see if a file with hyphenation exceptions needs to be read. This is the case when the third argument is not empty and when it does not contain a space token.

```
12.347 \def\bbl@tempa{#3}%
12.348 \ifx\bbl@tempa\@empty
12.349 \else
12.350 \ifx\bbl@tempa\space
12.351 \else
12.352 \input #3\relax
```

```

12.353     \fi
12.354     \fi
12.355 }

```

`\bbl@get@enc` The macro `\bbl@get@enc` extracts the font encoding from the language name and `\bbl@hyph@enc` stores it in `\bbl@hyph@enc`. It uses delimited arguments to achieve this.

```

12.356 \def\bbl@get@enc#1:#2\@@@{

```

First store both arguments in temporary macros,

```

12.357 \def\bbl@tempa{#1}%
12.358 \def\bbl@tempb{#2}%

```

then, if the second argument was empty, no font encoding was specified and we're done.

```

12.359 \ifx\bbl@tempb\@empty
12.360     \let\bbl@hyph@enc\@empty
12.361 \else

```

But if the second argument was *not* empty it will now have a superfluous colon attached to it which we need to remove. This done by feeding it to `\bbl@get@enc`. The string that we are after will then be in the first argument and be stored in `\bbl@tempa`.

```

12.362     \bbl@get@enc#2\@@@
12.363     \edef\bbl@hyph@enc{\bbl@tempa}%
12.364 \fi}

```

`\readconfigfile` The configuration file can now be opened for reading.

```

12.365 \openin1 = language.dat

```

See if the file exists, if not, use the default hyphenation file `hyphen.tex`. The user will be informed about this.

```

12.366 \ifeof1
12.367     \message{I couldn't find the file language.dat,\space
12.368             I will try the file hyphen.tex}
12.369     \input hyphen.tex\relax
12.370 \else

```

Pattern registers are allocated using count register `\last@language`. Its initial value is 0. The definition of the macro `\newlanguage` is such that it first increments the count register and then defines the language. In order to have the first patterns loaded in pattern register number 0 we initialize `\last@language` with the value `-1`.

```

12.371 \last@language\m@ne

```

We now read lines from the file until the end is found

```

12.372 \loop

```

While reading from the input, it is useful to switch off recognition of the end-of-line character. This saves us stripping off spaces from the contents of the control sequence.

```

12.373     \endlinechar\m@ne
12.374     \read1 to \bbl@line
12.375     \endlinechar'\^M

```

Empty lines are skipped.

```

12.376     \ifx\bbl@line\@empty
12.377     \else

```

Now we add a space and a `/` character to the end of `\bbl@line`. This is needed to be able to recognize the third, optional, argument of `\process@language` later on.

```

12.378     \edef\bbl@line{\bbl@line\space/}%
12.379     \expandafter\process@line\bbl@line
12.380 \fi

```

Check for the end of the file. To avoid a new `if` control sequence we create the necessary `\iftrue` or `\iffalse` with the help of `\csname`. But there is one complication with this approach: when skipping the `loop...repeat` \TeX has to read `\if/\fi` pairs. So we have to insert a ‘dummy’ `\iftrue`.

```
12.381 \iftrue \csname fi\endcsname
12.382 \csname if\ifeof1 false\else true\fi\endcsname
12.383 \repeat
```

Reactivate the default patterns,

```
12.384 \language=0
12.385 \fi
```

and close the configuration file.

```
12.386 \closein1
```

Also remove some macros from memory

```
12.387 \let\process@language\@undefined
12.388 \let\process@synonym\@undefined
12.389 \let\process@line\@undefined
12.390 \let\bbl@tempa\@undefined
12.391 \let\bbl@tempb\@undefined
12.392 \let\bbl@eq\@undefined
12.393 \let\bbl@line\@undefined
12.394 \let\bbl@get@enc\@undefined
```

We add a message about the fact that `babel` is loaded in the format and with which language patterns to the `\everyjob` register.

```
12.395 \ifx\addto@hook\@undefined
12.396 \else
12.397 \expandafter\addto@hook\expandafter\everyjob\expandafter{%
12.398 \expandafter\typeout\expandafter{\the\toks8 loaded.}}
12.399 \fi
```

Here the code for `ini \TeX` ends.

```
12.400 \</patterns>
12.401 \</kernel>
```

12.3 Support for active characters

`\bbl@add@special` The macro `\bbl@add@special` is used to add a new character (or single character control sequence) to the macro `\dospecials` (and `\@sanitize` if \LaTeX is used).

To keep all changes local, we begin a new group. Then we redefine the macros `\do` and `\@makeother` to add themselves and the given character without expansion.

```
12.402 (*core | shorthands)
12.403 \def\bbl@add@special#1{\begingroup
12.404 \def\do{\noexpand\do\noexpand}%
12.405 \def\@makeother{\noexpand\@makeother\noexpand}%
```

To add the character to the macros, we expand the original macros with the additional character inside the redefinition of the macros. Because `\@sanitize` can be undefined, we put the definition inside a conditional.

```
12.406 \edef\x{\endgroup
12.407 \def\noexpand\dospecials{\dospecials\do#1}%
12.408 \expandafter\ifx\csname @sanitize\endcsname\relax \else
12.409 \def\noexpand\@sanitize{\@sanitize\@makeother#1}%
12.410 \fi}%
```

The macro `\x` contains at this moment the following:

```
\endgroup\def\dospecials{old contents \do{char}}.
```

If `\@sanitize` is defined, it contains an additional definition of this macro. The last thing we have to do, is the expansion of `\x`. Then `\endgroup` is executed,

which restores the old meaning of `\x`, `\do` and `\@makeother`. After the group is closed, the new definition of `\dospecials` (and `\@sanitize`) is assigned.

12.411 `\x}`

`\bbl@remove@special` The companion of the former macro is `\bbl@remove@special`. It is used to remove a character from the set macros `\dospecials` and `\@sanitize`.

To keep all changes local, we begin a new group. Then we define a help macro `\x`, which expands to empty if the characters match, otherwise it expands to its nonexpandable input. Because TeX inserts a `\relax`, if the corresponding `\else` or `\fi` is scanned before the comparison is evaluated, we provide a ‘stop sign’ which should expand to nothing.

```
12.412 \def\bbl@remove@special#1{\begingroup
12.413   \def\x##1##2{\ifnum'#1='##2\noexpand\@empty
12.414     \else\noexpand##1\noexpand##2\fi}%
```

With the help of this macro we define `\do` and `\make@other`.

```
12.415   \def\do{x\do}%
12.416   \def\makeother{x\@makeother}%
```

The rest of the work is similar to `\bbl@add@special`.

```
12.417   \edef\x{\endgroup
12.418     \def\noexpand\dospecials{\dospecials}%
12.419     \expandafter\ifx\csname @sanitize\endcsname\relax \else
12.420       \def\noexpand\@sanitize{\@sanitize}%
12.421     \fi}%
12.422   \x}
```

12.4 Shorthands

`\initiate@active@char` A language definition file can call this macro to make a character active. This macro takes one argument, the character that is to be made active. When the character was already active this macro does nothing. Otherwise, this macro defines the control sequence `\normal@char⟨char⟩` to expand to the character in its ‘normal state’ and it defines the active character to expand to `\normal@char⟨char⟩` by default (`⟨char⟩` being the character to be made active). Later its definition can be changed to expand to `\active@char⟨char⟩` by calling `\bbl@activate{⟨char⟩}`.

For example, to make the double quote character active one could have the following line in a language definition file:

```
\initiate@active@char{"}
```

`\bbl@afterelse` Because the code that is used in the handling of active characters may need to look ahead, we take extra care to ‘throw’ it over the `\else` and `\fi` parts of an `\if`-statement⁵. These macros will break if another `\if... \fi` statement appears in one of the arguments.

`\bbl@afterfi`

```
12.423 \long\def\bbl@afterelse#1\else#2\fi{\fi#1}
12.424 \long\def\bbl@afterfi#1\fi{\fi#1}
```

`\peek@token` To prevent error messages when a shorthand, which normally takes an argument, sees a `\par`, or `}`, or similar tokens, we need to be able to ‘peek’ at what is coming up next in the input stream. Depending on the category code of the token that is seen, we need to either continue the code for the active character, or insert the non-active version of that character in the output. The macro `\peek@token` therefore takes two arguments, with which it constructs the control sequence to expand next. It \let’s `\bbl@nexta` and `\bbl@nextb` to the two possible macros. This is necessary for `\bbl@test@token` to take the right decision.

⁵This code is based on code presented in TUGboat vol. 12, no2, June 1991 in “An expansion Power Lemma” by Sonja Maus.

```

12.425 %\def\peek@token#1#2{%
12.426 %  \expandafter\let\expandafter\bbl@nexta\csname #1\string#2\endcsname
12.427 %  \expandafter\let\expandafter\bbl@nextb
12.428 %    \csname system@active\string#2\endcsname
12.429 %  \futurelet\bbl@token\bbl@test@token}

```

`\bbl@test@token` When the result of peeking at the next token has yielded a token with category ‘letter’, ‘other’ or ‘active’ it is safe to proceed with evaluating the code for the shorthand. When a token is found with any other category code proceeding is unsafe and therefor the original shorthand character is inserted in the output. The macro that calls `\bbl@test@token` needs to setup `\bbl@nexta` and `\bbl@nextb` in order to achieve this.

```

12.430 %\def\bbl@test@token{%
12.431 %  \let\bbl@next\bbl@nexta
12.432 %  \ifcat\noexpand\bbl@token a%
12.433 %    \else
12.434 %      \ifcat\noexpand\bbl@token=%
12.435 %        \else
12.436 %          \ifcat\noexpand\bbl@token\noexpand\bbl@next
12.437 %            \else
12.438 %              \let\bbl@next\bbl@nextb
12.439 %            \fi
12.440 %          \fi
12.441 %        \fi
12.442 %      \bbl@next}

```

The macro `\initiate@active@char` takes all the necessary actions to make its argument a shorthand character. The real work is performed once for each character.

```

12.443 \def\initiate@active@char#1{%
12.444   \expandafter\ifx\csname active@char\string##1\endcsname\relax
12.445     \bbl@afterfi{\@initiate@active@char{#1}}%
12.446   \fi}

```

Note that the definition of `\@initiate@active@char` needs an active character, for this the `~` is used. Some of the changes we need, do not have to become available later on, so we do it inside a group.

```

12.447 \begingroup
12.448   \catcode'\~\active
12.449   \def\x{\endgroup
12.450     \def\@initiate@active@char##1{%

```

If the character is already active we provide the default expansion under this shorthand mechanism.

```

12.451       \ifcat\noexpand##1\noexpand~\relax
12.452       \ifundefined{normal@char\string##1}{%
12.453         \expandafter\let\csname normal@char\string##1\endcsname##1%
12.454         \expandafter\gdef
12.455           \expandafter##1%
12.456           \expandafter{%
12.457             \expandafter\active@prefix\expandafter##1%
12.458             \csname normal@char\string##1\endcsname}}}%
12.459       \else

```

Otherwise we write a message in the transcript file,

```

12.460         \@activated{##1}%

```

and define `\normal@char⟨char⟩` to expand to the character in its default state.

```

12.461         \@namedef{normal@char\string##1}{##1}%

```

If we are making the right quote active we need to change `\pr@m@s` as well.

```

12.462         \ifx##1'%
12.463         \let\prim@s\bbl@prim@s

```

Also, make sure that a single ' in math mode 'does the right thing'.

```
12.464      \namedef{normal@char\string##1}{%
12.465      \textormath{##1}{~\bgroup\prim@s}}}%
12.466      \fi
```

If we are using the caret as a shorthand character special care should be taken to make sure math still works. Therefor an extra level of expansion is introduced with a check for math mode on the upper level.

```
12.467      \ifx##1~%
12.468      \gdef\bbl@act@caret{%
12.469      \ifmmode
12.470      \csname normal@char\string~\endcsname
12.471      \else
12.472      \bbl@afterfi
12.473      {\ifsafe@actives
12.474      \bbl@afterelse\csname normal@char\string##1\endcsname
12.475      \else
12.476      \bbl@afterfi\csname user@active\string##1\endcsname
12.477      \fi}%
12.478      \fi}
12.479      \fi
```

To prevent problems with the loading of other packages after babel we reset the catcode of the character at the end of the package.

```
12.480      \ifpackagewith{babel}{KeepShorthandsActive}{%
12.481      \edef\bbl@tempa{\catcode'\noexpand##1\the\catcode'##1}%
12.482      \expandafter\AtEndOfPackage\expandafter{\bbl@tempa}}%
```

Now we set the lowercase code of the ~ equal to that of the character to be made active and execute the rest of the code inside a \lowercase 'environment'.

```
12.483      \@tempcnta=\lccode'\~
12.484      \lccode'\~='##1%
12.485      \lowercase{%
```

Make the character active and add it to \dospecials and \@sanitize.

```
12.486      \catcode'\~\active
12.487      \expandafter\bbl@add@special
12.488      \csname \string##1\endcsname
```

Also re-activate it again at \begin{document}.

```
12.489      \AtBeginDocument{%
12.490      \catcode'##1\active
```

We also need to make sure that the shorthands are active during the processing of the .aux file. Otherwise some citations may give unexpected results in the printout when a shorthand was used in the optional argument of \bibitem for example.

```
12.491      \if@files
12.492      \immediate\write\@mainaux{%
12.493      \string\catcode'##1\string\active}%
12.494      \fi}%
```

Define the character to expand to

`\active@prefix <char> \normal@char<char>`

(where \active@char<char> is one control sequence!).

```
12.495      \expandafter\gdef
12.496      \expandafter~%
12.497      \expandafter{%
12.498      \expandafter\active@prefix\expandafter##1%
12.499      \csname normal@char\string##1\endcsname}}%
12.500      \lccode'\~\@tempcnta
12.501      \fi
```

For the active caret we first expand to `\bbl@act@caret` in order to be able to handle math mode correctly.

```
12.502 \ifx##1~%
12.503 \namedef{active@char\string##1}{\bbl@act@caret}%
12.504 \else
```

We define the first level expansion of `\active@char⟨char⟩` to check the status of the `@safe@actives` flag. If it is set to true we expand to the ‘normal’ version of this character, otherwise we call `\@active@char⟨char⟩`.

```
12.505 \namedef{active@char\string##1}{%
12.506 \if@safe@actives
12.507 \bbl@afterelse\csname normal@char\string##1\endcsname
12.508 \else
12.509 \bbl@afterfi\csname user@active\string##1\endcsname
12.510 \fi}%
12.511 \fi
```

The next level of the code checks whether a user has defined a shorthand for himself with this character. First we check for a single character shorthand. If that doesn’t exist we check for a shorthand with an argument.

```
12.512 \namedef{user@active\string##1}{%
12.513 \expandafter\ifx
12.514 \csname \user@group @sh@\string##1@\endcsname
12.515 \relax
12.516 \bbl@afterelse\bbl@sh@select\user@group##1%
12.517 {user@active@arg\string##1}{language@active\string##1}%
12.518 \else
12.519 \bbl@afterfi\csname \user@group @sh@\string##1@\endcsname
12.520 \fi}%
```

When there is also no user-level shorthand with an argument we will check whether there is a language defined shorthand for this active character. Before the next token is absorbed as argument we need to make sure that this is safe. Therefore `\peek@token` is called to decide that.

```
12.521 \long\namedef{user@active@arg\string##1}####1{%
12.522 \expandafter\ifx
12.523 \csname \user@group @sh@\string##1@\string####1@\endcsname
12.524 \relax
12.525 \bbl@afterelse
12.526 \csname language@active\string##1\endcsname####1%
12.527 \else
12.528 \bbl@afterfi
12.529 \csname \user@group @sh@\string##1@\string####10%
12.530 \endcsname
12.531 \fi}%
```

In order to do the right thing when a shorthand with an argument is used by itself at the end of the line we provide a definition for the case of an empty argument. For that case we let the shorthand character expand to its non-active self.

```
12.532 \namedef{\user@group @sh@\string##1@@}{%
12.533 \csname normal@char\string##1\endcsname}
```

Like the shorthands that can be defined by the user, a language definition file can also define shorthands with and without an argument, so we need two more macros to check if they exist.

```
12.534 \namedef{language@active\string##1}{%
12.535 \expandafter\ifx
12.536 \csname \language@group @sh@\string##1@\endcsname
12.537 \relax
12.538 \bbl@afterelse\bbl@sh@select\language@group##1%
12.539 {language@active@arg\string##1}{system@active\string##1}%
12.540 \else
12.541 \bbl@afterfi
```

```

12.542      \csname \language@group @sh@\string##1\endcsname
12.543      \fi}%

12.544      \long\@namedef{language@active@arg\string##1}####1{%
12.545      \expandafter\ifx
12.546      \csname \language@group @sh@\string##1\string####1\endcsname
12.547      \relax
12.548      \bbl@afterelse
12.549      \csname system@active\string##1\endcsname####1%
12.550      \else
12.551      \bbl@afterfi
12.552      \csname \language@group @sh@\string##1\string####1%
12.553      \endcsname
12.554      \fi}%

```

And the same goes for the system level.

```

12.555      \@namedef{system@active\string##1}{%
12.556      \expandafter\ifx
12.557      \csname \system@group @sh@\string##1\endcsname
12.558      \relax
12.559      \bbl@afterelse\bbl@sh@select\system@group##1%
12.560      {system@active@arg\string##1}{normal@char\string##1}%
12.561      \else
12.562      \bbl@afterfi\csname \system@group @sh@\string##1\endcsname
12.563      \fi}%

```

When no shorthands were found the ‘normal’ version of the active character is inserted.

```

12.564      \long\@namedef{system@active@arg\string##1}####1{%
12.565      \expandafter\ifx
12.566      \csname \system@group @sh@\string##1\string####1\endcsname
12.567      \relax
12.568      \bbl@afterelse\csname normal@char\string##1\endcsname####1%
12.569      \else
12.570      \bbl@afterfi
12.571      \csname \system@group @sh@\string##1\string####1\endcsname
12.572      \fi}%

```

When a shorthand combination such as ‘’ ends up in a heading T_EX would see `\protect\protect`. To prevent this from happening a shorthand needs to be defined at user level.

```

12.573      \@namedef{user@sh@\string##1\string\protect}{%
12.574      \csname user@active\string##1\endcsname}%
12.575      }%
12.576      }\x

```

`\bbl@sh@select` This command helps the shorthand supporting macros to select how to proceed. Note that this macro needs to be expandable as do all the shorthand macros in order for them to work in expansion-only environments such as the argument of `\hyphenation`.

This macro expects the name of a group of shorthands in its first argument and a shorthand character in its second argument. It will expand to either `\bbl@firstcs` or `\bbl@scndcs`. Hence two more arguments need to follow it.

```

12.577 \def\bbl@sh@select#1#2{%
12.578   \expandafter\ifx\csname#1@sh@\string#2@sel\endcsname\relax
12.579   \bbl@afterelse\bbl@scndcs
12.580   \else
12.581   \bbl@afterfi\csname#1@sh@\string#2@sel\endcsname
12.582   \fi
12.583 }

```

`\active@prefix` The command `\active@prefix` which is used in the expansion of active characters has a function similar to `\OT1-cmd` in that it `\protects` the active character whenever `\protect` is *not* `\@typeset@protect`.

```
12.584 \def\active@prefix#1{%
12.585   \ifx\protect\@typeset@protect
12.586   \else
```

When `\protect` is set to `\@unexpandable@protect` we make sure that the active character is *als not* expanded by inserting `\noexpand` in front of it. The `\@gobble` is needed to remove a token such as `\activechar:` (when the double colon was the active character to be dealt with).

```
12.587   \ifx\protect\@unexpandable@protect
12.588     \bbl@afterelse\bbl@afterfi\noexpand#1\@gobble
12.589   \else
12.590     \bbl@afterfi\bbl@afterfi\protect#1\@gobble
12.591   \fi
12.592 \fi}
```

`\if@safe@actives` In some circumstances it is necessary to be able to change the expansion of an active character on the fly. For this purpose the switch `@safe@actives` is available. The setting of this switch should be checked in the first level expansion of `\active@char⟨char⟩`.

```
12.593 \newif\if@safe@actives
12.594 \@safe@activesfalse
```

`\bbl@restore@actives` When the output routine kicks in while the active characters were made “safe” this must be undone in the headers to prevent unexpected typeset results. For this situation we define a command to make them “unsafe” again.

```
12.595 \def\bbl@restore@actives{\if@safe@actives\@safe@activesfalse\fi}
```

`\bbl@activate` This macro takes one argument, like `\initiate@active@char`. The macro is used to change the definition of an active character to expand to `\active@char⟨char⟩` instead of `\normal@char⟨char⟩`.

```
12.596 \def\bbl@activate#1{%
12.597   \expandafter\def
12.598   \expandafter#1\expandafter{%
12.599     \expandafter\active@prefix
12.600     \expandafter#1\csname active@char\string#1\endcsname}%
12.601 }
```

`\bbl@deactivate` This macro takes one argument, like `\bbl@activate`. The macro doesn’t really make a character non-active; it changes its definition to expand to `\normal@char⟨char⟩`.

```
12.602 \def\bbl@deactivate#1{%
12.603   \expandafter\def
12.604   \expandafter#1\expandafter{%
12.605     \expandafter\active@prefix
12.606     \expandafter#1\csname normal@char\string#1\endcsname}%
12.607 }
```

`\bbl@firstcs` These macros have two arguments. They use one of their arguments to build a `\bbl@scndcs` control sequence from.

```
12.608 \def\bbl@firstcs#1#2{\csname#1\endcsname}
12.609 \def\bbl@scndcs#1#2{\csname#2\endcsname}
```

`\declare@shorthand` The command `\declare@shorthand` is used to declare a shorthand on a certain level. It takes three arguments:

1. a name for the collection of shorthands, i.e. ‘system’, or ‘dutch’;
2. the character (sequence) that makes up the shorthand, i.e. `~` or `"a`;

3. the code to be executed when the shorthand is encountered.

```

12.610 \def\declare@shorthand#1#2{\@decl@short{#1}#2\@nil}
12.611 \def\@decl@short#1#2#3\@nil#4{%
12.612   \def\bbl@tempa{#3}%
12.613   \ifx\bbl@tempa\@empty
12.614     \expandafter\let\csname #1@sh@\string#2@sel\endcsname\bbl@scndcs
12.615     \@namedef{#1@sh@\string#2@}{#4}%
12.616   \else
12.617     \expandafter\let\csname #1@sh@\string#2@sel\endcsname\bbl@firstcs
12.618     \@namedef{#1@sh@\string#2@\string#3@}{#4}%
12.619   \fi}

```

`\textormath` Some of the shorthands that will be declared by the language definition files have to be usable in both text and mathmode. To achieve this the helper macro `\textormath` is provided.

```

12.620 \def\textormath#1#2{%
12.621   \ifmmode
12.622     \bbl@afterelse#2%
12.623   \else
12.624     \bbl@afterfi#1%
12.625   \fi}

```

`\user@group` The current concept of ‘shorthands’ supports three levels or groups of shorthands.
`\language@group` For each level the name of the level or group is stored in a macro. The default is
`\system@group` to have a user group; use language group ‘english’ and have a system group called ‘system’.

```

12.626 \def\user@group{user}
12.627 \def\language@group{english}
12.628 \def\system@group{system}

```

`\useshorthands` This is the user level command to tell L^AT_EX that user level shorthands will be used in the document. It takes one argument, the character that starts a shorthand.

```

12.629 \def\useshorthands#1{%

```

First note that this is user level.

```

12.630   \def\user@group{user}%

```

Then initialize the character for use as a shorthand character.

```

12.631   \initiate@active@char{#1}%

```

Now that T_EX has seen the character its category code is fixed, but for the actions of `\bbl@activate` to succeed we need it to be active. Hence the trick with the `\lccode` to circumvent this.

```

12.632   \@tempcnta\lccode'\~
12.633   \lccode'\~='#1%
12.634   \lowercase{\catcode'\~\active\bbl@activate{~}}%
12.635   \lccode'\~\@tempcnta}

```

`\defineshorthand` Currently we only support one group of user level shorthands, called ‘user’.

```

12.636 \def\defineshorthand{\declare@shorthand{user}}

```

`\languageshorthands` A user level command to change the language from which shorthands are used.

```

12.637 \def\languageshorthands#1{\def\language@group{#1}}

```

`\aliasshorthand`

```

12.638 \def\aliasshorthand#1#2{%

```

First the new shorthand needs to be initialized,

```

12.639   \expandafter\ifx\csname active@char\string#2\endcsname\relax
12.640     \ifx\document\@notprerr
12.641       \@notshorthand{#2}
12.642     \else
12.643       \initiate@active@char{#2}%

```

Then we need to use the `\lccode` trick to make the new shorthand behave like the old one. Therefore we save the current `\lccode` of the `~`-character and restore it later. Then we `\let` the new shorthand character be equal to the original.

```

12.644     \@tempcnta\lccode'\~
12.645     \lccode'\~='#2%
12.646     \lowercase{\let~#1}%
12.647     \lccode'\~\@tempcnta
12.648     \fi
12.649     \fi
12.650 }

```

`\@notshorthand`

```

12.651 \def\@notshorthand#1{%
12.652     \PackageError{babel}{%
12.653         The character '\string #1' should be made
12.654         a shorthand character;\MessageBreak
12.655         add the command \string\usesshorthands\string{#1\string} to
12.656         the preamble.\MessageBreak
12.657         I will ignore your instruction}}{%
12.658 }

```

`\shorthandon` The first level definition of these macros just passes the argument on to `\shorthandoff` `\bbl@switch@sh`, adding `\@nil` at the end to denote the end of the list of characters.

```

12.659 \newcommand*\shorthandon[1]{\bbl@switch@sh{on}#1\@nil}
12.660 \newcommand*\shorthandoff[1]{\bbl@switch@sh{off}#1\@nil}

```

`\bbl@switch@sh` The macro `\bbl@switch@sh` takes the list of characters apart one by one and subsequently switches the category code of the shorthand character according to the first argument of `\bbl@switch@sh`.

```

12.661 \def\bbl@switch@sh#1#2#3\@nil{%

```

But before any of this switching takes place we make sure that the character we are dealing with is known as a shorthand character. If it is, a macro such as `\active@char` should exist.

```

12.662     \ifundefined{active@char\string#2}{%
12.663         \PackageError{babel}{%
12.664             The character '\string #2' is not a shorthand character
12.665             in \language}%
12.666             Maybe you made a typing mistake?\MessageBreak
12.667             I will ignore your instruction}}{%
12.668         \csname bbl@switch@sh@#1\endcsname#2}%

```

Now that, as the first character in the list has been taken care of, we pass the rest of the list back to `\bbl@switch@sh`.

```

12.669     \ifx#3\@empty\else
12.670         \bbl@afterfi\bbl@switch@sh{#1}#3\@nil
12.671     \fi}

```

`\bbl@switch@sh@off` All that is left to do is define the actual switching macros. Switching off is easy, we just set the category code to 'other' (12).

```

12.672 \def\bbl@switch@sh@off#1{\catcode'#112\relax}

```

`\bbl@switch@sh@on` But switching the shorthand character back on is a bit more tricky. It involves making sure that we have an active character to begin with when the macro is being defined. It also needs the use of `\lowercase` and `\lccode` trickery to get everything to work out as expected. And to keep things local that need to remain local a group is opened, which is closed as soon as `\x` gets executed.

```

12.673 \begingroup
12.674     \catcode'\~\active
12.675     \def\x{\endgroup

```



```

12.676 \def\bbl@switch@sh@on##1{%
12.677 \begingroup
12.678 \lccode'~='##1%
12.679 \lowercase{\endgroup
12.680 \catcode'\active
12.681 }%
12.682 }%
12.683 }

```

The next operation makes the above definition effective.

```

12.684 \x
12.685 %

```

To prevent problems with constructs such as `\char"01A` when the double quote is made active, we define a shorthand on system level.

```

12.686 \declare@shorthand{system}{"}{\csname normal@char\string\endcsname}

```

When the right quote is made active we need to take care of handling it correctly in mathmode. Therefore we define a shorthand at system level to make it expand to a non-active right quote in textmode, but expand to its original definition in mathmode. (Note that the right quote is ‘active’ in mathmode because of its mathcode.)

```

12.687 \declare@shorthand{system}{'}{%
12.688 \textormath{\csname normal@char\string'\endcsname}%
12.689 {\sp\bgroup\prim@s}}

```

When the left quote is made active we need to take care of handling it correctly when it is followed by for instance an open brace token. Therefore we define a shorthand at system level to make it expand to a non-active left quote.

```

12.690 \declare@shorthand{system}{'}{\csname normal@char\string'\endcsname}

```

`\bbl@prim@s` One of the internal macros that are involved in substituting `\prime` for each right quote in mathmode is `\prim@s`. This checks if the next character is a right quote. When the right quote is active, the definition of this macro needs to be adapted to look for an active right quote.

```

12.691 \def\bbl@prim@s{%
12.692 \prime\futurelet\@let@token\bbl@pr@m@s}
12.693 \begingroup
12.694 \catcode'\'\active\let'\relax
12.695 \def\x{\endgroup
12.696 \def\bbl@pr@m@s{%
12.697 \ifx'\@let@token
12.698 \expandafter\pr@@@s
12.699 \else
12.700 \ifx^\@let@token
12.701 \expandafter\expandafter\expandafter\pr@@@t
12.702 \else
12.703 \egroup
12.704 \fi
12.705 \fi}%
12.706 }
12.707 \x

```

```

12.708 </core | shorthands>

```

Normally the `~` is active and expands to `\penalty\@M__`. When it is written to the `.aux` file it is written expanded. To prevent that and to be able to use the character `~` as a start character for a shorthand, it is redefined here as a one character shorthand on system level.

```

12.709 <*core>
12.710 \initiate@active@char{~}
12.711 \declare@shorthand{system}{~}{\leavevmode\nobreak\ }
12.712 \bbl@activate{~}

```

`\OT1dqpos` The position of the double quote character is different for the OT1 and T1 encodings. It will later be selected using the `\f@encoding` macro. Therefor we define two macros here to store the position of the character in these encodings.

```
12.713 \expandafter\def\csname OT1dqpos\endcsname{127}
12.714 \expandafter\def\csname T1dqpos\endcsname{4}
```

When the macro `\f@encoding` is undefined (as it is in plain T_EX) we define it here to expand to OT1

```
12.715 \ifx\f@encoding\@undefined
12.716   \def\f@encoding{OT1}
12.717 \fi
```

12.5 Language attributes

Language attributes provide a means to give the user control over which features of the language definition files he wants to enable.

`\languageattribute` The macro `\languageattribute` checks whether its arguments are valid and then activates the selected language attribute.

```
12.718 \newcommand\languageattribute[2]{%
```

First check whether the language is known.

```
12.719   \expandafter\ifx\csname l@#1\endcsname\relax
12.720     \@nolanerr{#1}%
12.721   \else
```

Then process each attribute in the list.

```
12.722     \@for\bb1@attr:=#2\do{%
```

We want to make sure that each attribute is selected only once; therefor we store the already selected attributes in `\bb1@known@attribs`. When that control sequence is not yet defined this attribute is certainly not selected before.

```
12.723       \ifx\bb1@known@attribs\@undefined
12.724         \in@false
12.725       \else
```

Now we need to see if the attribute occurs in the list of already selected attributes.

```
12.726         \edef\bb1@tempa{\noexpand\in@{, #1-\bb1@attr,}%
12.727           {,\bb1@known@attribs,}}%
12.728         \bb1@tempa
12.729       \fi
```

When the attribute was in the list we issue a warning; this might not be the users intention.

```
12.730         \ifin@
12.731         \PackageWarning{Babel}{%
12.732           You have more than once selected the attribute
12.733           '\bb1@attr'\MessageBreak for language #1}%
12.734       \else
```

When we end up here the attribute is not selected before. So, we add it to the list of selected attributes and execute the associated T_EX-code.

```
12.735         \edef\bb1@tempa{%
12.736           \noexpand\bb1@add@list\noexpand\bb1@known@attribs{#1-\bb1@attr}}%
12.737         \bb1@tempa
12.738         \edef\bb1@tempa{#1-\bb1@attr}%
12.739         \expandafter\bb1@ifknown@ttrib\expandafter{\bb1@tempa}\bb1@attributes%
12.740         {\csname#1@attr@\bb1@attr\endcsname}%
12.741         {\@attrerr{#1}{\bb1@attr}}%
12.742       \fi
12.743     }
12.744 \fi}
```

This command should only be used in the preamble of a document.

```
12.745 \@onlypreamble\languageattribute
```

The error text to be issued when an unknown attribute is selected.

```
12.746 \newcommand*{\@attrerr}[2]{%
12.747   \PackageError{babel}%
12.748     {The attribute #2 is unknown for language #1.}%
12.749     {Your command will be ignored, type <return> to proceed}}
```

`\bbl@declare@ttribute` This command adds the new language/attribute combination to the list of known attributes.

```
12.750 \def\bbl@declare@ttribute#1#2#3{%
12.751   \bbl@add@list\bbl@attributes{#1-#2}%
```

Then it defines a control sequence to be executed when the attribute is used in a document. The result of this should be that the macro `\extras...` for the current language is extended, otherwise the attribute will not work as its code is removed from memory at `\begin{document}`.

```
12.752   \expandafter\def\csname#1@attr@#2\endcsname{#3}%
12.753 }
```

`\bbl@ifattributeset` This internal macro has 4 arguments. It can be used to interpret \TeX code based on whether a certain attribute was set. This command should appear inside the argument to `\AtBeginDocument` because the attributes are set in the document preamble, *after* `babel` is loaded.

The first argument is the language, the second argument the attribute being checked, and the third and fourth arguments are the true and false clauses.

```
12.754 \def\bbl@ifattributeset#1#2#3#4{%
```

First we need to find out if any attributes were set; if not we're done.

```
12.755   \ifx\bbl@known@attribs\@undefined
12.756     \in@false
12.757   \else
```

The we need to check the list of known attributes.

```
12.758   \edef\bbl@tempa{\noexpand\in@{,#1-#2,}%
12.759     {,\bbl@known@attribs,}}%
12.760   \bbl@tempa
12.761   \fi
```

When we're this far `\ifin@` has a value indicating if the attribute in question was set or not. Just to be safe the code to be executed is 'thrown over the `\fi`'.

```
12.762   \ifin@
12.763     \bbl@afterelse#3%
12.764   \else
12.765     \bbl@afterfi#4%
12.766   \fi
12.767 }
```

`\bbl@add@list` This internal macro adds its second argument to a comma separated list in its first argument. When the list is not defined yet (or empty), it will be initiated

```
12.768 \def\bbl@add@list#1#2{%
12.769   \ifx#1\@undefined
12.770     \def#1{#2}%
12.771   \else
12.772     \ifx#1\@empty
12.773       \def#1{#2}%
12.774     \else
12.775       \edef#1{#1,#2}%
12.776     \fi
12.777   \fi
12.778 }
```

`\bbl@ifknown@ttrib` An internal macro to check whether a given language/attribute is known. The macro takes 4 arguments, the language/attribute, the attribute list, the \TeX -code

to be executed when the attribute is known and the \TeX -code to be executed otherwise.

```
12.779 \def\bbl@ifknown@ttrib#1#2{%
```

We first assume the attribute is unknown.

```
12.780 \let\bbl@tempa\@secondoftwo
```

Then we loop over the list of known attributes, trying to find a match.

```
12.781 \@for\bbl@tempb:=#2\do{%
```

```
12.782 \expandafter\in@\expandafter{\expandafter,\bbl@tempb,}\{,#1,}%
```

```
12.783 \ifin@
```

When a match is found the definition of `\bbl@tempa` is changed.

```
12.784 \let\bbl@tempa\@firstoftwo
```

```
12.785 \else
```

```
12.786 \fi}%
```

Finally we execute `\bbl@tempa`.

```
12.787 \bbl@tempa
```

```
12.788 }
```

`\bbl@clear@ttribs` This macro removes all the attribute code from \LaTeX 's memory at `\begin{document}` time (if any is present).

```
12.789 \def\bbl@clear@ttribs{%
```

```
12.790 \ifx\bbl@attributes\@undefined\else
```

```
12.791 \@for\bbl@tempa:=\bbl@attributes\do{%
```

```
12.792 \expandafter\bbl@clear@ttrib\bbl@tempa.
```

```
12.793 }%
```

```
12.794 \let\bbl@attributes\@undefined
```

```
12.795 \fi
```

```
12.796 }
```

```
12.797 \def\bbl@clear@ttrib#1-#2.{%
```

```
12.798 \expandafter\let\csname#1@attr@#2\endcsname\@undefined}
```

```
12.799 \AtBeginDocument{\bbl@clear@ttribs}
```

12.6 Support for saving macro definitions

To save the meaning of control sequences using `\babel@save`, we use temporary control sequences. To save hash table entries for these control sequences, we don't use the name of the control sequence to be saved to construct the temporary name. Instead we simply use the value of a counter, which is reset to zero each time we begin to save new values. This works well because we release the saved meanings before we begin to save a new set of control sequence meanings (see `\selectlanguage` and `\originalTeX`).

`\babel@savecnt` The initialization of a new save cycle: reset the counter to zero.

```
\babel@beginsave12.800 \def\babel@beginsave{\babel@savecnt\z@}
```

Before it's forgotten, allocate the counter and initialize all.

```
12.801 \newcount\babel@savecnt
```

```
12.802 \babel@beginsave
```

`\babel@save` The macro `\babel@save{csname}` saves the current meaning of the control sequence `csname` to `\originalTeX`⁶. To do this, we let the current meaning to a temporary control sequence, the restore commands are appended to `\originalTeX` and the counter is incremented.

```
12.803 \def\babel@save#1{%
```

```
12.804 \expandafter\let\csname babel@\number\babel@savecnt\endcsname #1\relax
```

```
12.805 \begingroup
```

```
12.806 \toks@\expandafter{\originalTeX \let#1=}%
```

```
12.807 \edef\x{\endgroup
```

⁶`\originalTeX` has to be expandable, i. e. you shouldn't let it to `\relax`.

```

12.808      \def\noexpand\originalTeX{\the\toks@ \expandafter\noexpand
12.809          \csname babel@number\babel@savecnt\endcsname\relax}}%
12.810      \x
12.811      \advance\babel@savecnt\@ne}

```

`\babel@savevariable` The macro `\babel@savevariable⟨variable⟩` saves the value of the variable. `⟨variable⟩` can be anything allowed after the `\the` primitive.

```

12.812 \def\babel@savevariable#1{\begingroup
12.813     \toks@\expandafter{\originalTeX #1=}%
12.814     \edef\x{\endgroup
12.815         \def\noexpand\originalTeX{\the\toks@ \the#1\relax}}%
12.816     \x}

```

`\bbl@frenchspacing` Some languages need to have `\frenchspacing` in effect. Others don't want that.
`\bbl@nonfrenchspacing` The command `\bbl@frenchspacing` switches it on when it isn't already in effect and `\bbl@nonfrenchspacing` switches it off if necessary.

```

12.817 \def\bbl@frenchspacing{%
12.818     \ifnum\the\sfcode'\.=\@m
12.819         \let\bbl@nonfrenchspacing\relax
12.820     \else
12.821         \frenchspacing
12.822         \let\bbl@nonfrenchspacing\nonfrenchspacing
12.823     \fi}
12.824 \let\bbl@nonfrenchspacing\nonfrenchspacing

```

12.7 Support for extending macros

`\addto` For each language four control sequences have to be defined that control the language-specific definitions. To be able to add something to these macro once they have been defined the macro `\addto` is introduced. It takes two arguments, a `⟨control sequence⟩` and T_EX-code to be added to the `⟨control sequence⟩`.

If the `⟨control sequence⟩` has not been defined before it is defined now.

```

12.825 \def\addto#1#2{%
12.826     \ifx#1\@undefined
12.827         \def#1{#2}%
12.828     \else

```

The control sequence could also expand to `\relax`, in which case a circular definition results. The net result is a stack overflow.

```

12.829     \ifx#1\relax
12.830         \def#1{#2}%
12.831     \else

```

Otherwise the replacement text for the `⟨control sequence⟩` is expanded and stored in a token register, together with the T_EX-code to be added. Finally the `⟨control sequence⟩` is redefined, using the contents of the token register.

```

12.832         {\toks@\expandafter{#1#2}%
12.833         \xdef#1{\the\toks@}}%
12.834     \fi
12.835 \fi
12.836 }

```

12.8 Macros common to a number of languages

`\allowhyphens` This macro makes hyphenation possible. Basically its definition is nothing more than `\nobreak \hskip 0pt plus 0pt`⁷.

```

12.837 \def\bbl@t@one{T1}
12.838 \def\allowhyphens{%
12.839     \ifx\cf@encoding\bbl@t@one\else\bbl@allowhyphens\fi}
12.840 \def\bbl@allowhyphens{\nobreak\hskip\z@skip}

```

⁷T_EX begins and ends a word for hyphenation at a glue node. The penalty prevents a linebreak at this glue node.

`\set@low@box` The following macro is used to lower quotes to the same level as the comma. It prepares its argument in box register 0.

```
12.841 \def\set@low@box#1{\setbox\tw@\hbox{,}\setbox\z@\hbox{#1}%
12.842   \dimen\z@\ht\z@ \advance\dimen\z@ -\ht\tw@%
12.843   \setbox\z@\hbox{\lower\dimen\z@ \box\z@}\ht\z@\ht\tw@ \dp\z@\dp\tw@}
```

`\save@sf@q` The macro `\save@sf@q` is used to save and reset the current space factor.

```
12.844 \def\save@sf@q #1{\leavevmode
12.845   \begingroup
12.846   \edef\@SF{\spacefactor \the\spacefactor}#1\@SF
12.847   \endgroup
12.848 }
```

`\bbl@disc` For some languages the macro `\bbl@disc` is used to ease the insertion of discretionary for letters that behave ‘abnormally’ at a breakpoint.

```
12.849 \def\bbl@disc#1#2{%
12.850   \nobreak\discretionary{#2-}{#1}\allowhyphens}
```

12.9 Making glyphs available

The file `babel.dtx`⁸ makes a number of glyphs available that either do not exist in the OT1 encoding and have to be ‘faked’, or that are not accessible through `T1enc.def`.

12.10 Quotation marks

`\quotedblbase` In the T1 encoding the opening double quote at the baseline is available as a separate character, accessible via `\quotedblbase`. In the OT1 encoding it is not available, therefore we make it available by lowering the normal open quote character to the baseline.

```
12.851 \ProvideTextCommand{\quotedblbase}{OT1}{%
12.852   \save@sf@q{\set@low@box{\textquotedblright\}}%
12.853   \box\z@\kern-.04em\allowhyphens}}
```

Make sure that when an encoding other than OT1 or T1 is used this glyph can still be typeset.

```
12.854 \ProvideTextCommandDefault{\quotedblbase}{%
12.855   \UseTextSymbol{OT1}{\quotedblbase}}
```

`\quotesinglbase` We also need the single quote character at the baseline.

```
12.856 \ProvideTextCommand{\quotesinglbase}{OT1}{%
12.857   \save@sf@q{\set@low@box{\textquoteright\}}%
12.858   \box\z@\kern-.04em\allowhyphens}}
```

Make sure that when an encoding other than OT1 or T1 is used this glyph can still be typeset.

```
12.859 \ProvideTextCommandDefault{\quotesinglbase}{%
12.860   \UseTextSymbol{OT1}{\quotesinglbase}}
```

`\guillemotleft` The guillemet characters are not available in OT1 encoding. They are faked.

```
\guillemotright12.861 \ProvideTextCommand{\guillemotleft}{OT1}{%
12.862   \ifmmode
12.863     \ll
12.864   \else
12.865     \save@sf@q{\nobreak
12.866       \raise.2ex\hbox{$\scriptscriptstyle\ll$}\allowhyphens}%
12.867     \fi}
12.868 \ProvideTextCommand{\guillemotright}{OT1}{%
```

⁸The file described in this section has version number v3.8l, and was last revised on 2008/03/16.

```

12.869 \ifmmode
12.870 \gg
12.871 \else
12.872 \save@sf@q{\nobreak
12.873 \raise.2ex\hbox{$\scriptscriptstyle\gg$}\allowhyphens}%
12.874 \fi}

```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.

```

12.875 \ProvideTextCommandDefault{\guillemotleft}{\%
12.876 \UseTextSymbol{OT1}{\guillemotleft}}
12.877 \ProvideTextCommandDefault{\guillemotright}{\%
12.878 \UseTextSymbol{OT1}{\guillemotright}}

```

`\guilsinglleft` The single guillemets are not available in OT1 encoding. They are faked.

```

\guilsinglright12.879 \ProvideTextCommand{\guilsinglleft}{OT1}{\%
12.880 \ifmmode
12.881 <%
12.882 \else
12.883 \save@sf@q{\nobreak
12.884 \raise.2ex\hbox{$\scriptscriptstyle<$}\allowhyphens}%
12.885 \fi}
12.886 \ProvideTextCommand{\guilsinglright}{OT1}{\%
12.887 \ifmmode
12.888 >%
12.889 \else
12.890 \save@sf@q{\nobreak
12.891 \raise.2ex\hbox{$\scriptscriptstyle>$}\allowhyphens}%
12.892 \fi}

```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.

```

12.893 \ProvideTextCommandDefault{\guilsinglleft}{\%
12.894 \UseTextSymbol{OT1}{\guilsinglleft}}
12.895 \ProvideTextCommandDefault{\guilsinglright}{\%
12.896 \UseTextSymbol{OT1}{\guilsinglright}}

```

12.11 Letters

`\ij` The dutch language uses the letter ‘ij’. It is available in T1 encoded fonts, but not `\IJ` in the OT1 encoded fonts. Therefore we fake it for the OT1 encoding.

```

12.897 \DeclareTextCommand{\ij}{OT1}{\%
12.898 \allowhyphens i\kern-0.02em j\allowhyphens}
12.899 \DeclareTextCommand{\IJ}{OT1}{\%
12.900 \allowhyphens I\kern-0.02em J\allowhyphens}
12.901 \DeclareTextCommand{\ij}{T1}{\char188}
12.902 \DeclareTextCommand{\IJ}{T1}{\char156}

```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.

```

12.903 \ProvideTextCommandDefault{\ij}{\%
12.904 \UseTextSymbol{OT1}{\ij}}
12.905 \ProvideTextCommandDefault{\IJ}{\%
12.906 \UseTextSymbol{OT1}{\IJ}}

```

`\dj` The croatian language needs the letters `\dj` and `\DJ`; they are available in the T1 `\DJ` encoding, but not in the OT1 encoding by default.

Some code to construct these glyphs for the OT1 encoding was made available to me by Stipcevic Mario, (stipcevic@olimp.irb.hr).

```

12.907 \def\crrtic@{\hrule height0.1ex width0.3em}
12.908 \def\crttic@{\hrule height0.1ex width0.33em}
12.909 %

```

```

12.910 \def\ddj@{%
12.911   \setbox0\hbox{d}\dimen@=\ht0
12.912   \advance\dimen@1ex
12.913   \dimen@.45\dimen@
12.914   \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
12.915   \advance\dimen@ii.5ex
12.916   \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crrtic@}}}}
12.917 \def\DDJ@{%
12.918   \setbox0\hbox{D}\dimen@=.55\ht0
12.919   \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
12.920   \advance\dimen@ii.15ex % correction for the dash position
12.921   \advance\dimen@ii-.15\fontdimen7\font % correction for cmtt font
12.922   \dimen\thr@@\expandafter\rem@pt\the\fontdimen7\font\dimen@
12.923   \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crrtic@}}}}
12.924 %
12.925 \DeclareTextCommand{\dj}{OT1}{\ddj@ d}
12.926 \DeclareTextCommand{\DJ}{OT1}{\DDJ@ D}

    Make sure that when an encoding other than OT1 or T1 is used these glyphs can
    still be typeset.

12.927 \ProvideTextCommandDefault{\dj}{%
12.928   \UseTextSymbol{OT1}{\dj}}
12.929 \ProvideTextCommandDefault{\DJ}{%
12.930   \UseTextSymbol{OT1}{\DJ}}

```

\SS For the T1 encoding \SS is defined and selects a specific glyph from the font, but for other encodings it is not available. Therefor we make it available here.

```

12.931 \DeclareTextCommand{\SS}{OT1}{SS}
12.932 \ProvideTextCommandDefault{\SS}{\UseTextSymbol{OT1}{\SS}}

```

12.12 Shorthands for quotation marks

Shorthands are provided for a number of different quotation marks, which make them usable both outside and inside mathmode.

\glq The ‘german’ single quotes.

```

\grq12.933 \ProvideTextCommand{\glq}{OT1}{%
12.934   \textormath{\quotesinglbase}{\mbox{\quotesinglbase}}}
12.935 \ProvideTextCommand{\glq}{T1}{%
12.936   \textormath{\quotesinglbase}{\mbox{\quotesinglbase}}}
12.937 \ProvideTextCommandDefault{\glq}{\UseTextSymbol{OT1}\glq}

```

The definition of \grq depends on the fontencoding. With T1 encoding no extra kerning is needed.

```

12.938 \ProvideTextCommand{\grq}{T1}{%
12.939   \textormath{\textquoteleft}{\mbox{\textquoteleft}}}
12.940 \ProvideTextCommand{\grq}{OT1}{%
12.941   \save@sf@q{\kern-.0125em%
12.942   \textormath{\textquoteleft}{\mbox{\textquoteleft}}}%
12.943   \kern.07em\relax}}
12.944 \ProvideTextCommandDefault{\grq}{\UseTextSymbol{OT1}\grq}

```

\glqq The ‘german’ double quotes.

```

\grqq12.945 \ProvideTextCommand{\glqq}{OT1}{%
12.946   \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
12.947 \ProvideTextCommand{\glqq}{T1}{%
12.948   \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
12.949 \ProvideTextCommandDefault{\glqq}{\UseTextSymbol{OT1}\glqq}

```

The definition of \grqq depends on the fontencoding. With T1 encoding no extra kerning is needed.

```

12.950 \ProvideTextCommand{\grqq}{T1}{%
12.951   \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}

```



```

12.952 \ProvideTextCommand{\grqq}{OT1}{%
12.953   \save@sf@q{\kern-.07em%
12.954   \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}%
12.955   \kern.07em\relax}}
12.956 \ProvideTextCommandDefault{\grqq}{\UseTextSymbol{OT1}\grqq}

```

`\flq` The ‘french’ single guillemets.

```

\frq12.957 \ProvideTextCommand{\flq}{OT1}{%
12.958   \textormath{\guilsinglleft}{\mbox{\guilsinglleft}}}
12.959 \ProvideTextCommand{\flq}{T1}{%
12.960   \textormath{\guilsinglleft}{\mbox{\guilsinglleft}}}
12.961 \ProvideTextCommandDefault{\flq}{\UseTextSymbol{OT1}\flq}

12.962 \ProvideTextCommand{\frq}{OT1}{%
12.963   \textormath{\guilsinglright}{\mbox{\guilsinglright}}}
12.964 \ProvideTextCommand{\frq}{T1}{%
12.965   \textormath{\guilsinglright}{\mbox{\guilsinglright}}}
12.966 \ProvideTextCommandDefault{\frq}{\UseTextSymbol{OT1}\frq}

```

`\flqq` The ‘french’ double guillemets.

```

\frqq12.967 \ProvideTextCommand{\flqq}{OT1}{%
12.968   \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
12.969 \ProvideTextCommand{\flqq}{T1}{%
12.970   \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
12.971 \ProvideTextCommandDefault{\flqq}{\UseTextSymbol{OT1}\flqq}

12.972 \ProvideTextCommand{\frqq}{OT1}{%
12.973   \textormath{\guillemotright}{\mbox{\guillemotright}}}
12.974 \ProvideTextCommand{\frqq}{T1}{%
12.975   \textormath{\guillemotright}{\mbox{\guillemotright}}}
12.976 \ProvideTextCommandDefault{\frqq}{\UseTextSymbol{OT1}\frqq}

```

12.13 Umlauts and trema’s

The command `\` needs to have a different effect for different languages. For German for instance, the ‘umlaut’ should be positioned lower than the default position for placing it over the letters a, o, u, A, O and U. When placed over an e, i, E or I it can retain its normal position. For Dutch the same glyph is always placed in the lower position.

`\umlauthigh` To be able to provide both positions of `\` we provide two commands to switch the positioning, the default will be `\umlauthigh` (the normal positioning).

```

\umlautlow
12.977 \def\umlauthigh{%
12.978   \def\bbl@umlauta##1{\leavevmode\bgroup%
12.979     \expandafter\accent\csname\f@encoding dqpos\endcsname
12.980     ##1\allowhyphens\egroup}%
12.981   \let\bbl@umlaute\bbl@umlauta}
12.982 \def\umlautlow{%
12.983   \def\bbl@umlauta{\protect\lower@umlaut}}
12.984 \def\umlautelow{%
12.985   \def\bbl@umlaute{\protect\lower@umlaut}}
12.986 \umlauthigh

```

`\lower@umlaut` The command `\lower@umlaut` is used to position the `\` closer to the letter.

We want the umlaut character lowered, nearer to the letter. To do this we need an extra *<dimen>* register.

```

12.987 \expandafter\ifx\csname U@D\endcsname\relax
12.988   \csname newdimen\endcsname U@D
12.989 \fi

```

The following code fools T_EX’s `make_accent` procedure about the current x-height of the font to force another placement of the umlaut character.

```

12.990 \def\lower@umlaut#1{%

```

First we have to save the current x-height of the font, because we'll change this font dimension and this is always done globally.

```
12.991 \leavevmode\bgroup
12.992 \U@D 1ex%
```

Then we compute the new x-height in such a way that the umlaut character is lowered to the base character. The value of `.45ex` depends on the METAFONT parameters with which the fonts were built. (Just try out, which value will look best.)

```
12.993 {\setbox\z@\hbox{%
12.994 \expandafter\char\csname\f@encoding dqpos\endcsname}%
12.995 \dimen@ -.45ex\advance\dimen@\ht\z@
```

If the new x-height is too low, it is not changed.

```
12.996 \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
```

Finally we call the `\accent` primitive, reset the old x-height and insert the base character in the argument.

```
12.997 \expandafter\accent\csname\f@encoding dqpos\endcsname
12.998 \fontdimen5\font\U@D #1%
12.999 \egroup}
```

For all vowels we declare `\` to be a composite command which uses `\bbl@umlauta` or `\bbl@umlaute` to position the umlaut character. We need to be sure that these definitions override the ones that are provided when the package `fontenc` with option `OT1` is used. Therefore these declarations are postponed until the beginning of the document.

```
12.1000 \AtBeginDocument{%
12.1001 \DeclareTextCompositeCommand{"}{OT1}{a}{\bbl@umlauta{a}}%
12.1002 \DeclareTextCompositeCommand{"}{OT1}{e}{\bbl@umlaute{e}}%
12.1003 \DeclareTextCompositeCommand{"}{OT1}{i}{\bbl@umlaute{i}}%
12.1004 \DeclareTextCompositeCommand{"}{OT1}{\i}{\bbl@umlaute{i}}%
12.1005 \DeclareTextCompositeCommand{"}{OT1}{o}{\bbl@umlauta{o}}%
12.1006 \DeclareTextCompositeCommand{"}{OT1}{u}{\bbl@umlauta{u}}%
12.1007 \DeclareTextCompositeCommand{"}{OT1}{A}{\bbl@umlauta{A}}%
12.1008 \DeclareTextCompositeCommand{"}{OT1}{E}{\bbl@umlaute{E}}%
12.1009 \DeclareTextCompositeCommand{"}{OT1}{I}{\bbl@umlaute{I}}%
12.1010 \DeclareTextCompositeCommand{"}{OT1}{O}{\bbl@umlauta{O}}%
12.1011 \DeclareTextCompositeCommand{"}{OT1}{U}{\bbl@umlauta{U}}%
12.1012 }
```

12.14 The redefinition of the style commands

The rest of the code in this file can only be processed by L^AT_EX, so we check the current format. If it is plain T_EX, processing should stop here. But, because of the need to limit the scope of the definition of `\format`, a macro that is used locally in the following `\if` statement, this comparison is done inside a group. To prevent T_EX from complaining about an unclosed group, the processing of the command `\endinput` is deferred until after the group is closed. This is accomplished by the command `\aftergroup`.

```
12.1013 {\def\format{lplain}
12.1014 \ifx\fmtname\format
12.1015 \else
12.1016 \def\format{LaTeX2e}
12.1017 \ifx\fmtname\format
12.1018 \else
12.1019 \aftergroup\endinput
12.1020 \fi
12.1021 \fi}
```

Now that we're sure that the code is seen by L^AT_EX only, we have to find out what the main (primary) document style is because we want to redefine some

macros. This is only necessary for releases of L^AT_EX dated before December 1991. Therefor this part of the code can optionally be included in `babel.def` by specifying the `docstrip` option `names`.

12.1022 `(*names)`

The standard styles can be distinguished by checking whether some macros are defined. In table 1 an overview is given of the macros that can be used for this purpose.

article	:	both the <code>\chapter</code> and <code>\opening</code> macros are undefined
report and book	:	the <code>\chapter</code> macro is defined and the <code>\opening</code> is undefined
letter	:	the <code>\chapter</code> macro is undefined and the <code>\opening</code> is defined

Table 1: How to determine the main document style

The macros that have to be redefined for the `report` and `book` document styles happen to be the same, so there is no need to distinguish between those two styles.

`\doc@style` First a parameter `\doc@style` is defined to identify the current document style. This parameter might have been defined by a document style that already uses macros instead of hard-wired texts, such as `artikel1.sty` [6], so the existence of `\doc@style` is checked. If this macro is undefined, i.e., if the document style is unknown and could therefore contain hard-wired texts, `\doc@style` is defined to the default value ‘0’.

12.1023 `\ifx\@undefined\doc@style`

12.1024 `\def\doc@style{0}%`

This parameter is defined in the following `if` construction (see table 1):

```

12.1025 \ifx\@undefined\opening
12.1026 \ifx\@undefined\chapter
12.1027 \def\doc@style{1}%
12.1028 \else
12.1029 \def\doc@style{2}%
12.1030 \fi
12.1031 \else
12.1032 \def\doc@style{3}%
12.1033 \fi%
12.1034 \fi%
```

12.14.1 Redefinition of macros

Now here comes the real work: we start to redefine things and replace hard-wired texts by macros. These redefinitions should be carried out conditionally, in case it has already been done.

For the `figure` and `table` environments we have in all styles:

```

12.1035 \@ifundefined{figurename}{\def\fnun@figure{\figurename} \thefigure}}{}
12.1036 \@ifundefined{tablename}{\def\fnun@table{tablename} \thetable}}{}%
```

The rest of the macros have to be treated differently for each style. When `\doc@style` still has its default value nothing needs to be done.

12.1037 `\ifcase \doc@style\relax`

12.1038 `\or`

This means that `babel.def` is read after the `article` style, where no `\chapter` and `\opening` commands are defined⁹.

⁹A fact that was pointed out to me by Nico Poppelier and was already used in Piet van Oostrum’s document style option `n1`.

First we have the \tableofcontents, \listoffigures and \listoftables:

```

12.1039 \@ifundefined{contentsname}%
12.1040   {\def\tableofcontents{\section*{\contentsname\@mkboth
12.1041     {\uppercase{\contentsname}}{\uppercase{\contentsname}}}%
12.1042     \@starttoc{toc}}}%
12.1043
12.1044 \@ifundefined{listfigurename}%
12.1045   {\def\listoffigures{\section*{\listfigurename\@mkboth
12.1046     {\uppercase{\listfigurename}}{\uppercase{\listfigurename}}}%
12.1047     \@starttoc{lof}}}%
12.1048
12.1049 \@ifundefined{listtablename}%
12.1050   {\def\listoftables{\section*{\listtablename\@mkboth
12.1051     {\uppercase{\listtablename}}{\uppercase{\listtablename}}}%
12.1052     \@starttoc{lot}}}%

```

Then the \thebibliography and \theindex environments.

```

12.1053 \@ifundefined{refname}%
12.1054   {\def\thebibliography#1{\section*{\refname
12.1055     \@mkboth{\uppercase{\refname}}{\uppercase{\refname}}}%
12.1056     \list{[arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
12.1057       \leftmargin\labelwidth
12.1058       \advance\leftmargin\labelsep
12.1059       \usecounter{enumi}}%
12.1060     \def\newblock{\hskip.11em plus.33em minus.07em}%
12.1061     \sloppy\clubpenalty4000\widowpenalty\clubpenalty
12.1062     \sfcode'\.=1000\relax}}%
12.1063
12.1064 \@ifundefined{indexname}%
12.1065   {\def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
12.1066     \columnseprule \z@
12.1067     \columnsep 35pt\twocolumn[\section*{\indexname}]%
12.1068     \@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
12.1069     \thispagestyle{plain}%
12.1070     \parskip\z@ plus.3pt\parindent\z@\let\item\@idxitem}}%

```

The abstract environment:

```

12.1071 \@ifundefined{abstractname}%
12.1072   {\def\abstract{\if@twocolumn
12.1073     \section*{\abstractname}%
12.1074     \else \small
12.1075     \begin{center}%
12.1076     {\bf \abstractname\vspace{-.5em}\vspace{\z@}}%
12.1077     \end{center}%
12.1078     \quotation
12.1079     \fi}}%

```

And last but not least, the macro \part:

```

12.1080 \@ifundefined{partname}%
12.1081   {\def\@part[#1]#2{\ifnum \c@secnumdepth >\m@ne
12.1082     \refstepcounter{part}%
12.1083     \addcontentsline{toc}{part}{\thepart
12.1084       \hspace{1em}#1}\else
12.1085     \addcontentsline{toc}{part}{#1}\fi
12.1086     {\parindent\z@ \raggedright
12.1087     \ifnum \c@secnumdepth >\m@ne
12.1088       \Large \bf \partname{} \thepart
12.1089       \par \nobreak
12.1090     \fi
12.1091     \huge \bf
12.1092     #2\markboth{}{}\par}%
12.1093     \nobreak
12.1094     \vskip 3ex\@afterheading}%

```

12.1095 }{}

This is all that needs to be done for the `article` style.

12.1096 \or

The next case is formed by the two styles `book` and `report`. Basically we have to do the same as for the `article` style, except now we must also change the `\chapter` command.

The tables of contents, figures and tables:

```
12.1097 \@ifundefined{contentsname}%
12.1098   {\def\tableofcontents{\@restonecolfalse
12.1099     \if@twocolumn\@restonecoltrue\onecolumn
12.1100     \fi\chapter*{\contentsname\@mkboth
12.1101       {\uppercase{\contentsname}}{\uppercase{\contentsname}}}%
12.1102     \@starttoc{toc}%
12.1103     \csname if@restonecol\endcsname\twocolumn
12.1104     \csname fi\endcsname}}{}
12.1105
12.1106 \@ifundefined{listfigurename}%
12.1107   {\def\listoffigures{\@restonecolfalse
12.1108     \if@twocolumn\@restonecoltrue\onecolumn
12.1109     \fi\chapter*{\listfigurename\@mkboth
12.1110       {\uppercase{\listfigurename}}{\uppercase{\listfigurename}}}%
12.1111     \@starttoc{lof}%
12.1112     \csname if@restonecol\endcsname\twocolumn
12.1113     \csname fi\endcsname}}{}
12.1114
12.1115 \@ifundefined{listtablename}%
12.1116   {\def\listoftables{\@restonecolfalse
12.1117     \if@twocolumn\@restonecoltrue\onecolumn
12.1118     \fi\chapter*{\listtablename\@mkboth
12.1119       {\uppercase{\listtablename}}{\uppercase{\listtablename}}}%
12.1120     \@starttoc{lot}%
12.1121     \csname if@restonecol\endcsname\twocolumn
12.1122     \csname fi\endcsname}}{}
12.1123
```

Again, the bibliography and index environments; notice that in this case we use `\bibname` instead of `\refname` as in the definitions for the `article` style. The reason for this is that in the `article` document style the term ‘References’ is used in the definition of `\thebibliography`. In the `report` and `book` document styles the term ‘Bibliography’ is used.

```
12.1123 \@ifundefined{bibname}%
12.1124   {\def\thebibliography#1{\chapter*{\bibname
12.1125     \@mkboth{\uppercase{\bibname}}{\uppercase{\bibname}}}%
12.1126     \list{[arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
12.1127     \leftmargin\labelwidth \advance\leftmargin\labelsep
12.1128     \usecounter{enumi}}}%
12.1129     \def\newblock{\hspace{11em plus.33em minus.07em}%
12.1130     \sloppy\clubpenalty4000\widowpenalty\clubpenalty
12.1131     \sfcode‘\.=1000\relax}}{}
12.1132
12.1133 \@ifundefined{indexname}%
12.1134   {\def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
12.1135     \columnseprule \z@
12.1136     \columnsep 35pt\twocolumn[\@makeschapterhead{\indexname}]%
12.1137     \@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
12.1138     \thispagestyle{plain}%
12.1139     \parskip\z@ plus.3pt\parindent\z@ \let\item\@idxitem}}{}
12.1140
```

Here is the abstract environment:

```
12.1140 \@ifundefined{abstractname}%
12.1141   {\def\abstract{\titlepage
12.1142     \null\vfil
```

```

12.1143 \begin{center}%
12.1144 {\bf \abstractname}%
12.1145 \end{center}}{}

And last but not least the \chapter, \appendix and \part macros.

12.1146 \ifundefined{chaptername}{\def\@chapapp{\chaptername}}{}
12.1147 %
12.1148 \ifundefined{appendixname}%
12.1149 {\def\appendix{\par
12.1150 \setcounter{chapter}{0}%
12.1151 \setcounter{section}{0}%
12.1152 \def\@chapapp{\appendixname}%
12.1153 \def\thechapter{\Alph{chapter}}}}{}
12.1154 %
12.1155 \ifundefined{partname}%
12.1156 {\def\@part[#1]#2{\ifnum \c@secnumdepth >-2\relax
12.1157 \refstepcounter{part}%
12.1158 \addcontentsline{toc}{part}{\thepart
12.1159 \hspace{1em}#1}\else
12.1160 \addcontentsline{toc}{part}{#1}\fi
12.1161 \markboth{}{}}%
12.1162 {\centering
12.1163 \ifnum \c@secnumdepth >-2\relax
12.1164 \huge\bf \partname{} \thepart
12.1165 \par
12.1166 \vskip 20pt \fi
12.1167 \Huge \bf
12.1168 #1\par}\@endpart}}{}%
12.1169 \or

```

Now we address the case where `babel.def` is read after the `letter` style. The `letter` document style defines the macro `\opening` and some other macros that are specific to `letter`. This means that we have to redefine other macros, compared to the previous two cases.

First two macros for the material at the end of a letter, the `\cc` and `\encl` macros.

```

12.1170 \ifundefined{ccname}%
12.1171 {\def\cc#1{\par\noindent
12.1172 \parbox[t]{\textwidth}%
12.1173 {\@hangfrom{\rm \ccname : }\ignorespaces #1\strut}\par}}{}
12.1174
12.1175 \ifundefined{enclname}%
12.1176 {\def\encl#1{\par\noindent
12.1177 \parbox[t]{\textwidth}%
12.1178 {\@hangfrom{\rm \enclname : }\ignorespaces #1\strut}\par}}{}

```

The last thing we have to do here is to redefine the `headings` pagestyle:

```

12.1179 \ifundefined{headtoname}%
12.1180 {\def\ps@headings{%
12.1181 \def\@oddhead{\sl \headtoname{} \ignorespaces\toname \hfil
12.1182 \@date \hfil \pagename{} \thepage}%
12.1183 \def\@oddfoot{}}{}

```

This was the last of the four standard document styles, so if `\doc@style` has another value we do nothing and just close the `if` construction.

```

12.1184 \fi

```

Here ends the code that can be optionally included when a version of L^AT_EX is in use that is dated *before* December 1991.

```

12.1185 </names>
12.1186 </core>

```

12.15 Cross referencing macros

The L^AT_EX book states:

The *key* argument is any sequence of letters, digits, and punctuation symbols; upper- and lowercase letters are regarded as different.

When the above quote should still be true when a document is typeset in a language that has active characters, special care has to be taken of the category codes of these characters when they appear in an argument of the cross referencing macros.

When a cross referencing command processes its argument, all tokens in this argument should be character tokens with category ‘letter’ or ‘other’.

The only way to accomplish this in most cases is to use the trick described in the T_EXbook [1] (Appendix D, page 382). The primitive `\meaning` applied to a token expands to the current meaning of this token. For example, ‘`\meaning\A`’ with `\A` defined as ‘`\def\A#1{\B}`’ expands to the characters ‘`macro:#1->\B`’ with all category codes set to ‘other’ or ‘space’.

\bbl@redefine To redefine a command, we save the old meaning of the macro. Then we redefine it to call the original macro with the ‘sanitized’ argument. The reason why we do it this way is that we don’t want to redefine the L^AT_EX macros completely in case their definitions change (they have changed in the past).

Because we need to redefine a number of commands we define the command `\bbl@redefine` which takes care of this. It creates a new control sequence, `\org@...`

```
12.1187 \core | shorthands
12.1188 \def\bbl@redefine#1{%
12.1189   \edef\bbl@tempa{\expandafter\@gobble\string#1}%
12.1190   \expandafter\let\csname org@\bbl@tempa\endcsname#1
12.1191   \expandafter\def\csname\bbl@tempa\endcsname}
```

This command should only be used in the preamble of the document.

```
12.1192 \@onlypreamble\bbl@redefine
```

\bbl@redefine@long This version of `\babel@redefine` can be used to redefine `\long` commands such as `\ifthenelse`.

```
12.1193 \def\bbl@redefine@long#1{%
12.1194   \edef\bbl@tempa{\expandafter\@gobble\string#1}%
12.1195   \expandafter\let\csname org@\bbl@tempa\endcsname#1
12.1196   \expandafter\long\expandafter\def\csname\bbl@tempa\endcsname}
12.1197 \@onlypreamble\bbl@redefine@long
```

\bbl@redefineroobust For commands that are redefined, but which *might* be robust we need a slightly more intelligent macro. A robust command `foo` is defined to expand to `\protect\foo_`. So it is necessary to check whether `\foo_` exists.

```
12.1198 \def\bbl@redefineroobust#1{%
12.1199   \edef\bbl@tempa{\expandafter\@gobble\string#1}%
12.1200   \expandafter\ifx\csname\bbl@tempa\space\endcsname\relax
12.1201     \expandafter\let\csname org@\bbl@tempa\endcsname#1
12.1202     \expandafter\edef\csname\bbl@tempa\endcsname{\noexpand\protect
12.1203       \expandafter\noexpand\csname\bbl@tempa\space\endcsname}%
12.1204   \else
12.1205     \expandafter\let\csname org@\bbl@tempa\expandafter\endcsname
12.1206       \csname\bbl@tempa\space\endcsname
12.1207   \fi
```

The result of the code above is that the command that is being redefined is always robust afterwards. Therefore all we need to do now is define `\foo_`.

```
12.1208 \expandafter\def\csname\bbl@tempa\space\endcsname}
```

This command should only be used in the preamble of the document.

```
12.1209 \@onlypreamble\bbl@redefineroobust
```

`\newlabel` The macro `\label` writes a line with a `\newlabel` command into the `.aux` file to define labels.

```

12.1210 %\bbl@redefine\newlabel#1#2{%
12.1211 % \@safe@activestruel\org@newlabel{#1}{#2}\@safe@activesfalse}

```

`\@newl@bel` We need to change the definition of the L^AT_EX-internal macro `\@newl@bel`. This is needed because we need to make sure that shorthand characters expand to their non-active version.

```

12.1212 \def\@newl@bel#1#2#3{%

```

First we open a new group to keep the changed setting of `\protect` local and then we set the `@safe@actives` switch to true to make sure that any shorthand that appears in any of the arguments immediately expands to its non-active self.

```

12.1213 {%
12.1214 \@safe@activestruel
12.1215 \@ifundefined{#1@#2}%
12.1216 \relax
12.1217 {%
12.1218 \gdef \@multiplelabels {%
12.1219 \@latex@warning@no@line{There were multiply-defined labels}}%
12.1220 \@latex@warning@no@line{Label ‘#2’ multiply defined}%
12.1221 }%
12.1222 \global\@namedef{#1@#2}{#3}%
12.1223 }%
12.1224 }

```

`\@testdef` An internal L^AT_EX macro used to test if the labels that have been written on the `.aux` file have changed. It is called by the `\enddocument` macro. This macro needs to be completely rewritten, using `\meaning`. The reason for this is that in some cases the expansion of `\#1@#2` contains the same characters as the `#3`; but the character codes differ. Therefor L^AT_EX keeps reporting that the labels may have changed.

```

12.1225 \CheckCommand*\@testdef[3]{%
12.1226 \def\reserved@a{#3}%
12.1227 \expandafter\ifx\csname #1@#2\endcsname\reserved@a
12.1228 \else
12.1229 \@tempswatrue
12.1230 \fi}

```

Now that we made sure that `\@testdef` still has the same definition we can rewrite it. First we make the shorthands ‘safe’.

```

12.1231 \def\@testdef #1#2#3{%
12.1232 \@safe@activestruel

```

Then we use `\bbl@tempa` as an ‘alias’ for the macro that contains the label which is being checked.

```

12.1233 \expandafter\let\expandafter\bbl@tempa\csname #1@#2\endcsname

```

Then we define `\bbl@tempb` just as `\@newl@bel` does it.

```

12.1234 \def\bbl@tempb{#3}%
12.1235 \@safe@activesfalse

```

When the label is defined we replace the definition of `\bbl@tempa` by its meaning.

```

12.1236 \ifx\bbl@tempa\relax
12.1237 \else
12.1238 \edef\bbl@tempa{\expandafter\strip@prefix\meaning\bbl@tempa}%
12.1239 \fi

```

We do the same for `\bbl@tempb`.

```

12.1240 \edef\bbl@tempb{\expandafter\strip@prefix\meaning\bbl@tempb}%

```


If the label didn't change, `\bbl@tempa` and `\bbl@tempb` should be identical macros.

```
12.1241 \ifx \bbl@tempa \bbl@tempb
12.1242 \else
12.1243 \@tempswatru
12.1244 \fi}
```

`\ref` The same holds for the macro `\ref` that references a label and `\pageref` to reference a page. So we redefine `\ref` and `\pageref`. While we change these macros, we make them robust as well (if they weren't already) to prevent problems if they should become expanded at the wrong moment.

```
12.1245 \bbl@redefineroobust\ref#1{%
12.1246 \@safe@activestrue\org@ref{#1}\@safe@activessalse}
12.1247 \bbl@redefineroobust\pageref#1{%
12.1248 \@safe@activestrue\org@pageref{#1}\@safe@activessalse}
```

`\@citex` The macro used to cite from a bibliography, `\cite`, uses an internal macro, `\@citex`. It is this internal macro that picks up the argument(s), so we redefine this internal macro and leave `\cite` alone. The first argument is used for typesetting, so the shorthands need only be deactivated in the second argument.

```
12.1249 \bbl@redefine\@citex[#1]#2{%
12.1250 \@safe@activestrue\edef\@tempa{#2}\@safe@activessalse
12.1251 \org@@citex[#1]{\@tempa}}
```

Unfortunately, the packages `natbib` and `cite` need a different definition of `\@citex`... To begin with, `natbib` has a definition for `\@citex` with *three* arguments... We only know that a package is loaded when `\begin{document}` is executed, so we need to postpone the different redefinition.

```
12.1252 \AtBeginDocument{%
12.1253 \@ifpackageloaded{natbib}{%
```

Notice that we use `\def` here instead of `\bbl@redefine` because `\org@@citex` is already defined and we don't want to overwrite that definition (it would result in parameter stack overflow because of a circular definition).

```
12.1254 \def\@citex[#1][#2]#3{%
12.1255 \@safe@activestrue\edef\@tempa{#3}\@safe@activessalse
12.1256 \org@@citex[#1][#2]{\@tempa}}%
12.1257 }{}}
```

The package `cite` has a definition of `\@citex` where the shorthands need to be turned off in both arguments.

```
12.1258 \AtBeginDocument{%
12.1259 \@ifpackageloaded{cite}{%
12.1260 \def\@citex[#1]#2{%
12.1261 \@safe@activestrue\org@@citex[#1]{#2}\@safe@activessalse}%
12.1262 }{}}
```

`\nocite` The macro `\nocite` which is used to instruct BiBTeX to extract uncited references from the database.

```
12.1263 \bbl@redefine\nocite#1{%
12.1264 \@safe@activestrue\org@nocite{#1}\@safe@activessalse}
```

`\bibcite` The macro that is used in the `.aux` file to define citation labels. When packages such as `natbib` or `cite` are not loaded its second argument is used to typeset the citation label. In that case, this second argument can contain active characters but is used in an environment where `\@safe@activestrue` is in effect. This switch needs to be reset inside the `\hbox` which contains the citation label. In order to determine during `.aux` file processing which definition of `\bibcite` is needed we define `\bibcite` in such a way that it redefines itself with the proper definition.

```
12.1265 \bbl@redefine\bibcite{%
```

We call `\bbl@cite@choice` to select the proper definition for `\bibcite`. This new definition is then activated.

```
12.1266 \bbl@cite@choice
12.1267 \bibcite}
```

`\bbl@bibcite` The macro `\bbl@bibcite` holds the definition of `\bibcite` needed when neither `natbib` nor `cite` is loaded.

```
12.1268 \def\bbl@bibcite#1#2{%
12.1269 \org@bibcite{#1}\@safe@activesfalse#2}}
```

`\bbl@cite@choice` The macro `\bbl@cite@choice` determines which definition of `\bibcite` is needed.

```
12.1270 \def\bbl@cite@choice{%
    First we give \bibcite its default definition.
12.1271 \global\let\bibcite\bbl@bibcite
    Then, when natbib is loaded we restore the original definition of \bibcite .
12.1272 \@ifpackageloaded{natbib}{\global\let\bibcite\org@bibcite}{}%
    For cite we do the same.
12.1273 \@ifpackageloaded{cite}{\global\let\bibcite\org@bibcite}{}%
    Make sure this only happens once.
12.1274 \global\let\bbl@cite@choice\relax
12.1275 }
```

When a document is run for the first time, no `.aux` file is available, and `\bibcite` will not yet be properly defined. In this case, this has to happen before the document starts.

```
12.1276 \AtBeginDocument{\bbl@cite@choice}
```

`\@bibitem` One of the two internal L^AT_EX macros called by `\bibitem` that write the citation label on the `.aux` file.

```
12.1277 \bbl@redefine\@bibitem#1{%
12.1278 \safe@activetrue\org@@bibitem{#1}\@safe@activesfalse}
```

12.16 marks

`\markright` Because the output routine is asynchronous, we must pass the current language attribute to the head lines, together with the text that is put into them. To achieve this we need to adapt the definition of `\markright` and `\markboth` somewhat.

```
12.1279 \bbl@redefine\markright#1{%
    First of all we temporarily store the language switching command, using an ex-
    panded definition in order to get the current value of \language.
12.1280 \edef\bbl@tempb{\noexpand\protect
12.1281 \noexpand\foreignlanguage{\language}}%
    Then, we check whether the argument is empty; if it is, we just make sure the
    scratch token register is empty.
12.1282 \def\bbl@arg{#1}%
12.1283 \ifx\bbl@arg\empty
12.1284 \toks@{}%
12.1285 \else
```

Next, we store the argument to `\markright` in the scratch token register, together with the expansion of `\bbl@tempb` (containing the language switching command) as defined before. This way these commands will not be expanded by using `\edef` later on, and we make sure that the text is typeset using the correct language settings. While doing so, we make sure that active characters that may end up in the mark are not disabled by the output routine kicking in while `\safe@activetrue` is in effect.

```
12.1286 \expandafter\toks@\expandafter{%
12.1287 \bbl@tempb{\protect\bbl@restore@actives#1}}%
12.1288 \fi
```

Then we define a temporary control sequence using `\edef`.

```
12.1289 \edef\bbl@tempa{%
```

When `\bbl@tempa` is executed, only `\language` will be expanded, because of the way the token register was filled.

```
12.1290 \noexpand\org@markright{\the\toks@}}%
```

```
12.1291 \bbl@tempa
```

```
12.1292 }
```

`\markboth` The definition of `\markboth` is equivalent to that of `\markright`, except that we need two token registers. The documentclasses `report` and `book` define and set the headings for the page. While doing so they also store a copy of `\markboth` in `\@mkboth`. Therefore we need to check whether `\@mkboth` has already been set. If so we need to do that again with the new definition of `\makrboth`.

```
12.1293 \ifx\@mkboth\markboth
```

```
12.1294 \def\bbl@tempc{\let\@mkboth\markboth}
```

```
12.1295 \else
```

```
12.1296 \def\bbl@tempc{}
```

```
12.1297 \fi
```

Now we can start the new definition of `\markboth`

```
12.1298 \bbl@redefine\markboth#1#2{%
```

```
12.1299 \edef\bbl@tempb{\noexpand\protect
```

```
12.1300 \noexpand\foreignlanguage{\language}}%
```

```
12.1301 \def\bbl@arg{#1}%
```

```
12.1302 \ifx\bbl@arg\@empty
```

```
12.1303 \toks@{}%
```

```
12.1304 \else
```

```
12.1305 \expandafter\toks@\expandafter{%
```

```
12.1306 \bbl@tempb{\protect\bbl@restore@actives#1}}%
```

```
12.1307 \fi
```

```
12.1308 \def\bbl@arg{#2}%
```

```
12.1309 \ifx\bbl@arg\@empty
```

```
12.1310 \toks8{}%
```

```
12.1311 \else
```

```
12.1312 \expandafter\toks8\expandafter{%
```

```
12.1313 \bbl@tempb{\protect\bbl@restore@actives#2}}%
```

```
12.1314 \fi
```

```
12.1315 \edef\bbl@tempa{%
```

```
12.1316 \noexpand\org@markboth{\the\toks@}{\the\toks8}}%
```

```
12.1317 \bbl@tempa
```

```
12.1318 }
```

and copy it to `\@mkboth` if necessary.

```
12.1319 \bbl@tempc
```

```
12.1320 </core | shorthands>
```

12.17 Encoding issues (part 2)

`\TeX` Because documents may use font encodings other than one of the latin encodings,
`\LaTeX` we make sure that the logos of `TeX` and `LaTeX` always come out in the right encoding.

```
12.1321 <*core>
```

```
12.1322 \bbl@redefine\TeX{\textlatin{\org@TeX}}
```

```
12.1323 \bbl@redefine\LaTeX{\textlatin{\org@LaTeX}}
```

```
12.1324 </core>
```

12.18 Preventing clashes with other packages

12.18.1 `ifthen`

`\ifthenelse` Sometimes a document writer wants to create a special effect depending on the page a certain fragment of text appears on. This can be achieved by the following

piece of code:

```
\ifthenelse{\isodd{\pageref{some:label}}}{%
  {code for odd pages}%
  {code for even pages}%
}
```

In order for this to work the argument of `\isodd` needs to be fully expandable. With the above redefinition of `\pageref` it is not in the case of this example. To overcome that, we add some code to the definition of `\ifthenelse` to make things work.

The first thing we need to do is check if the package `ifthen` is loaded. This should be done at `\begin{document}` time.

```
12.1325 (*package)
12.1326 \AtBeginDocument{%
12.1327   \@ifpackageloaded{ifthen}{%
```

Then we can redefine `\ifthenelse`:

```
12.1328   \bbl@redefine@long\ifthenelse#1#2#3{%
```

We want to revert the definition of `\pageref` to its original definition for the duration of `\ifthenelse`, so we first need to store its current meaning.

```
12.1329     \let\bbl@tempa\pageref
12.1330     \let\pageref\org@pageref
```

Then we can set the `\@safe@actives` switch and call the original `\ifthenelse`. In order to be able to use shorthands in the second and third arguments of `\ifthenelse` the resetting of the switch *and* the definition of `\pageref` happens inside those arguments.

```
12.1331     \@safe@activestrue
12.1332     \org@ifthenelse{#1}{%
12.1333       \let\pageref\bbl@tempa
12.1334       \@safe@activesfalse
12.1335       #2}{%
12.1336       \let\pageref\bbl@tempa
12.1337       \@safe@activesfalse
12.1338       #3}%
12.1339   }%
```

When the package wasn't loaded we do nothing.

```
12.1340   }{}%
12.1341 }
```

12.18.2 varioref

`\@@vpageref` When the package `varioref` is in use we need to modify its internal command `\vrefpagemum` `\@@vpageref` in order to prevent problems when an active character ends up in the argument of `\vref`.

```
12.1342 \AtBeginDocument{%
12.1343   \@ifpackageloaded{varioref}{%
12.1344     \bbl@redefine\@@vpageref#1[#2]#3{%
12.1345       \@safe@activestrue
12.1346       \org@@vpageref{#1}[#2]{#3}%
12.1347       \@safe@activesfalse}%
12.1348   }
```

The same needs to happen for `\vrefpagemum`.

```
12.1348     \bbl@redefine\vrefpagemum#1#2{%
12.1349       \@safe@activestrue
12.1350       \org\vrefpagemum{#1}#2}%
12.1351     \@safe@activesfalse}%
12.1352 }
```

The package `varioref` defines `\Ref` to be a robust command with uppercases the first character of the reference text. In order to be able to do that it needs to access the expandable form of `\ref`. So we employ a little trick here. We

redefine the (internal) command `\Ref` to call `\org@ref` instead of `\ref`. The disadvantage of this solution is that whenever the definition of `\Ref` changes, this definition needs to be updated as well.

```
12.1352 \expandafter\def\csname Ref \endcsname#1{%
12.1353 \protected@edef\@tempa{\org@ref{#1}}\expandafter\MakeUppercase\@tempa}
12.1354 }{}%
12.1355 }
```

12.18.3 hhline

`\hhline` Delaying the activation of the shorthand characters has introduced a problem with the `hhline` package. The reason is that it uses the ‘:’ character which is made active by the french support in `babel`. Therefore we need to *reload* the package when the ‘:’ is an active character.

So at `\begin{document}` we check whether `hhline` is loaded.

```
12.1356 \AtBeginDocument{%
12.1357 \ifpackageloaded{hhline}%
```

Then we check whether the expansion of `\normal@char:` is not equal to `\relax`.

```
12.1358 {\expandafter\ifx\csname normal@char:string:\endcsname\relax
12.1359 \else
```

In that case we simply reload the package. Note that this happens *after* the category code of the @-sign has been changed to other, so we need to temporarily change it to letter again.

```
12.1360 \makeatletter
12.1361 \def\@currname{hhline}\input{hhline.sty}\makeatother
12.1362 \fi}%
12.1363 {}}
```

12.18.4 hyperref

`\pdfstringdefDisableCommands` Although a number of interworking problems between `babel` and `hyperref` are tackled by `hyperref` itself we need to take care of correctly handling the shorthand characters. When they get expanded inside a bookmark a warning will appear in the log file which can be prevented. This is done by informing `hyperref` that it should use the shorthands as defined on the system level rather than at the user level.

```
12.1364 \AtBeginDocument{%
12.1365 \ifundefined{pdfstringdefDisableCommands}%
12.1366 {}%
12.1367 {\pdfstringdefDisableCommands{%
12.1368 \languageshorthands{system}}}%
12.1369 }%
12.1370 }
```

12.18.5 General

`\FOREIGNLANGUAGE` The package `fancyhdr` treats the running head and foot lines somewhat differently as the standard classes. A symptom of this is that the command `\foreignlanguage` which `babel` adds to the marks can end up inside the argument of `\MakeUppercase`. To prevent unexpected results we need to define `\FOREIGNLANGUAGE` here.

```
12.1371 \DeclareRobustCommand{\FOREIGNLANGUAGE}[1]{%
12.1372 \lowercase{\foreignlanguage{#1}}}%
12.1373 }
```

`\nfss@catcodes` L^AT_EX’s font selection scheme sometimes wants to read font definition files in the middle of processing the document. In order to guard against any characters having the wrong `\catcodes` it always calls `\nfss@catcodes` before loading a file.

Unfortunately, the characters " and ' are not dealt with. Therefore we have to add them until L^AT_EX does that herself.

```

12.1374 <*core | shorthands>
12.1375 \ifx\nfss@catcodes\@undefined
12.1376 \else
12.1377   \addto\nfss@catcodes{%
12.1378     \@makeother\'%
12.1379     \@makeother"%
12.1380   }
12.1381 \fi

12.1382 </core | shorthands>

```

13 Local Language Configuration

`\loadlocalcfg` At some sites it may be necessary to add site-specific actions to a language definition file. This can be done by creating a file with the same name as the language definition file, but with the extension `.cfg`. For instance the file `norsk.cfg` will be loaded when the language definition file `norsk.ldf` is loaded.

```

13.1 <*core>

For plain-based formats we don't want to override the definition of \loadlocalcfg
from plain.def.

13.2 \ifx\loadlocalcfg\@undefined
13.3   \def\loadlocalcfg#1{%
13.4     \InputIfFileExists{#1.cfg}
13.5       {\typeout{*****~J%
13.6         * Local config file #1.cfg used~J%
13.7         *}%
13.8       }
13.9     {}}
13.10 \fi

```

Just to be compatible with L^AT_EX 2.09 we add a few more lines of code:

```

13.11 \ifx\@unexpandable@protect\@undefined
13.12   \def\@unexpandable@protect{\noexpand\protect\noexpand}
13.13   \long\def \protected@write#1#2#3{%
13.14     \begingroup
13.15       \let\thepage\relax
13.16       #2%
13.17       \let\protect\@unexpandable@protect
13.18       \edef\reserved@a{\write#1{#3}}%
13.19       \reserved@a
13.20     \endgroup
13.21     \if@nobreak\ifvmode\nobreak\fi\fi
13.22   }
13.23 \fi
13.24 </core>

```

Since `babel` version 3.4 all source files that are part of the `babel` system can be typeset separately. But to typeset them all in one document, the file `babel.drv` can be used. If you only want the information on how to use the `babel` system and what goodies are provided by the language-specific files, you can run the file `user.drv` through \LaTeX to get a user guide.

Here \d1qq is defined so that an example of " can be given.

The code lines are numbered within sections.

63

which should also be visible in the index; hence this redefinition of a macro from `doc.sty`.

```
14.51 \renewcommand\codeline@wrindex[1]{\if@filesw
14.52     \immediate\write\@indexfile
14.53     {\string\indexentry{#1}%
14.54     {\number\c@section.\number\c@CodelineNo}}\fi}
```

The glossary environment is used or the change log, but its definition needs changing for this document.

```
14.55 \renewenvironment{theglossary}{%
14.56     \glossary@prologue%
14.57     \GlossaryParms \let\item\@idxitem \ignorespaces}%
14.58 {}
14.59 \<!user>
14.60 \makeatother
```

A few shorthands used in the documentation

```
14.61 \font\manual=logo10 % font used for the METAFONT logo, etc.
14.62 \newcommand*{\MF}{\manual META}\-{\manual FONT}}
14.63 \newcommand*{\TeXhax}{\TeX hax}
14.64 \newcommand*{\babel}{\textsf{babel}}
14.65 \newcommand*{\Babel}{\textsf{Babel}}
14.66 \newcommand*{\m[1]{\mbox{$\langle\it#1\rangle$}}
14.67 \newcommand*{\langvar{\m{lang}}}
```

Some more definitions needed in the documentation.

```
14.68 %\newcommand*\note[1]{\textbf{#1}}
14.69 \newcommand*\note[1]{}
14.70 \newcommand*\bsl{\protect\bslash}
14.71 \newcommand*\Lopt[1]{\textsf{#1}}
14.72 \newcommand*\Lenv[1]{\textsf{#1}}
14.73 \newcommand*\file[1]{\texttt{#1}}
14.74 \newcommand*\cls[1]{\texttt{#1}}
14.75 \newcommand*\pkg[1]{\texttt{#1}}
14.76 \newcommand*\langdeffile[1]{%
14.77 \<-user> \clearpage
14.78 \DocInput{#1}}
```

When a full index should be generated uncomment the line with `\EnableCrossrefs`.

Beware, processing may take some time. Use `\DisableCrossrefs` when the index is ready.

```
14.79 % \EnableCrossrefs
14.80 \DisableCrossrefs
```

Include the change log.

```
14.81 \<-user>\RecordChanges
```

The index should use the linenumbers of the code.

```
14.82 \<-user>\CodelineIndex
```

Set everything in `\MacroFont` instead of `\AltMacroFont`

```
14.83 \setcounter{StandardModuleDepth}{1}
```

For the user guide we only want the description parts of all the files.

```
14.84 \<+user>\OnlyDescription
```

Here starts the document

```
14.85 \begin{document}
14.86 \DocInput{babel.dtx}
```

All the language definition files.

```
14.87 \<+user>\clearpage
14.88 \langdeffile{esperanto.dtx}
14.89 \langdeffile{interlingua.dtx}
14.90 %
14.91 \langdeffile{dutch.dtx}
```



```

14.92 \langdeffile{english.dtx}
14.93 \langdeffile{germanb.dtx}
14.94 \langdeffile{ngermanb.dtx}
14.95 %
14.96 \langdeffile{breton.dtx}
14.97 \langdeffile{welsh.dtx}
14.98 \langdeffile{irish.dtx}
14.99 \langdeffile{scottish.dtx}
14.100 %
14.101 \langdeffile{greek.dtx}
14.102 %
14.103 \langdeffile{frenchb.dtx}
14.104 \langdeffile{italian.dtx}
14.105 \langdeffile{latin.dtx}
14.106 \langdeffile{portuges.dtx}
14.107 \langdeffile{spanish.dtx}
14.108 \langdeffile{catalan.dtx}
14.109 \langdeffile{galician.dtx}
14.110 \langdeffile{basque.dtx}
14.111 \langdeffile{romanian.dtx}
14.112 %
14.113 \langdeffile{danish.dtx}
14.114 \langdeffile{icelandic.dtx}
14.115 \langdeffile{norsk.dtx}
14.116 \langdeffile{swedish.dtx}
14.117 \langdeffile{samin.dtx}
14.118 %
14.119 \langdeffile{finnish.dtx}
14.120 \langdeffile{magyar.dtx}
14.121 \langdeffile{estonian.dtx}
14.122 %
14.123 \langdeffile{albanian.dtx}
14.124 \langdeffile{croatian.dtx}
14.125 \langdeffile{czech.dtx}
14.126 \langdeffile{polish.dtx}
14.127 \langdeffile{serbian.dtx}
14.128 \langdeffile{slovak.dtx}
14.129 \langdeffile{slovene.dtx}
14.130 \langdeffile{russianb.dtx}
14.131 \langdeffile{bulgarian.dtx}
14.132 \langdeffile{ukraineb.dtx}
14.133 %
14.134 \langdeffile{lsorbian.dtx}
14.135 \langdeffile{usorbian.dtx}
14.136 \langdeffile{turkish.dtx}
14.137 %
14.138 \langdeffile{hebrew.dtx}
14.139 \DocInput{hebinp.dtx}
14.140 \DocInput{hebrew.fdd}
14.141 \DocInput{heb209.dtx}
14.142 \langdeffile{bahasa.dtx}
14.143 \langdeffile{bahasam.dtx}
14.144 %\langdeffile{sanskrit.dtx}
14.145 %\langdeffile{kannada.dtx}
14.146 %\langdeffile{nagari.dtx}
14.147 %\langdeffile{tamil.dtx}
14.148 \clearpage
14.149 \DocInput{bbplain.dtx}

Finally print the index and change log (not for the user guide).

14.150 (*!user)
14.151 \clearpage
14.152 \def\filename{index}

```

```
14.153 \PrintIndex
14.154 \clearpage
14.155 \def\filename{changes}
14.156 \PrintChanges
14.157 \!user}
14.158 \end{document}
14.159 \!driver}
```

15 Conclusion

A system of document options has been presented that enable the user of \LaTeX to adapt the standard document classes of \LaTeX to the language he or she prefers to use. These options offer the possibility of switching between languages in one document. The basic interface consists of using one option, which is the same for *all* standard document classes.

In some cases the language definition files provide macros that can be useful to plain \TeX users as well as to \LaTeX users. The `babel` system has been implemented so that it can be used by both groups of users.

16 Acknowledgements

I would like to thank all who volunteered as β -testers for their time. I would like to mention Julio Sanchez who supplied the option file for the Spanish language and Maurizio Codogno who supplied the option file for the Italian language. Michel Goossens supplied contributions for most of the other languages. Nico Poppelier helped polish the text of the documentation and supplied parts of the macros for the Dutch language. Paul Wackers and Werenfried Spit helped find and repair bugs.

During the further development of the `babel` system I received much help from Bernd Raichle, for which I am grateful.

17 References

- [1] Donald E. Knuth, *The \TeX book*, Addison-Wesley, 1986.
- [2] Leslie Lamport, *\LaTeX , A document preparation System*, Addison-Wesley, 1986.
- [3] K.F. Treebus. *Tekstwijzer, een gids voor het grafisch verwerken van tekst*. SDU Uitgeverij ('s-Gravenhage, 1988). A Dutch book on layout design and typography.
- [4] Hubert Partl, *German \TeX* , *TUGboat* 9 (1988) #1, p. 70–72.
- [5] Leslie Lamport, in: \TeX hax Digest, Volume 89, #13, 17 February 1989.
- [6] Johannes Braams, Victor Eijkhout and Nico Poppelier, *The development of national \LaTeX styles*, *TUGboat* 10 (1989) #3, p. 401–406.
- [7] Joachim Schrod, *International \LaTeX is ready to use*, *TUGboat* 11 (1990) #1, p. 87–90.

18 The Esperanto language

The file `esperanto.dtx`¹⁰ defines all the language-specific macros for the Esperanto language.

For this language the character `^` is made active. In table 2 an overview is given of its purpose.

<code>^c</code>	gives <code>ĉ</code> with hyphenation in the rest of the word allowed, this works for <code>c</code> , <code>C</code> , <code>g</code> , <code>G</code> , <code>H</code> , <code>J</code> , <code>s</code> , <code>S</code> , <code>z</code> , <code>Z</code>
<code>^h</code>	prevents <code>ĥ</code> from becoming too tall
<code>^j</code>	gives <code>ĵ</code>
<code>^u</code>	gives <code>ŭ</code> , with hyphenation in the rest of the word allowed
<code>^U</code>	gives <code>Ŭ</code> , with hyphenation in the rest of the word allowed
<code>^ </code>	inserts a <code>\discretionary{-}{-}{-}</code>

Table 2: The functions of the active character for Esperanto.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
18.1 (*code)
18.2 \LdfInit{esperanto}\captionesperanto
```

When this file is read as an option, i.e. by the `\usepackage` command, `esperanto` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@esperanto` to see whether we have to do something here.

```
18.3 \ifx\l@esperanto\@undefined
18.4 \@nopatterns{Esperanto}
18.5 \adddialect\l@esperanto0\fi
```

The next step consists of defining commands to switch to the Esperanto language. The reason for this is that a user might want to switch back and forth between languages.

`\captionesperanto` The macro `\captionesperanto` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
18.6 \addto\captionesperanto{%
18.7   \def\prefacename{Anta\^u{u}parolo}%
18.8   \def\refname{Cita\^j{j}oj}%
18.9   \def\abstractname{Resumo}%
18.10  \def\bibname{Bibliografio}%
18.11  \def\chaptername{{\^C}apitro}%
18.12  \def\appendixname{Apendico}%
18.13  \def\contentsname{Enhavo}%
18.14  \def\listfigurename{Listo de figuroj}%
18.15  \def\listtablename{Listo de tabeloj}%
18.16  \def\indexname{Indekso}%
18.17  \def\figurename{Figuro}%
18.18  \def\tablename{Tabelo}%
18.19  \def\partname{Parto}%
18.20  \def\enclname{Aldono(j)}%
18.21  \def\ccname{Kopie al}%
18.22  \def\headtoname{Al}%
18.23  \def\pagename{Pa\^go}%
18.24  \def\subjectname{Temo}%
18.25  \def\seename{vidu}% a^u: vd.
```

¹⁰The file described in this section has version number ? and was last revised on ?. A contribution was made by Ruiz-Altba Marti (ruizaltb@cernvm.cern.ch). Code from the file `esperant.sty` by Jörg Knappen (knappen@vkpmzd.kph.uni-mainz.de) was included.

```

18.26 \def\alsoname{vidu anka\u{u}}% a~u vd. anka\u{u}
18.27 \def\proofname{Pruvo}%
18.28 \def\glossaryname{Glosaro}%
18.29 }

```

`\dateesperanto` The macro `\dateesperanto` redefines the command `\today` to produce Esperanto dates.

```

18.30 \def\dateesperanto{%
18.31   \def\today{\number\day{--a}~de~\ifcase\month\or
18.32     januaro\or februaro\or marto\or aprilo\or majo\or junio\or
18.33     julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
18.34     decembro\fi,\space \number\year}}

```

`\extrasesperanto` The macro `\extrasesperanto` performs all the extra definitions needed for the Esperanto language. The macro `\noextrasesperanto` is used to cancel the actions of `\extrasesperanto`.

For Esperanto the `~` character is made active. This is done once, later on its definition may vary.

```

18.35 \initiate@active@char{~}

```

Because the character `~` is used in math mode with quite a different purpose we need to add an extra level of evaluation to the definition of the active `~`. It checks whether math mode is active; if so the shorthand mechanism is bypassed by a direct call of `\normal@char~`.

```

18.36 \addto\extrasesperanto{\languageshorthands{esperanto}}
18.37 \addto\extrasesperanto{\bbl@activate{~}}
18.38 \addto\noextrasesperanto{\bbl@deactivate{~}}

```

In order to prevent problems with the active `~` we add a shorthand on system level which expands to a `'normal ~`.

```

18.39 \declare@shorthand{system}{~}{\csname normal@char\string~\endcsname}

```

And here are the uses of the active `~`:

```

18.40 \declare@shorthand{esperanto}{^c}{\^c\allowhyphens}
18.41 \declare@shorthand{esperanto}{^C}{\^C\allowhyphens}
18.42 \declare@shorthand{esperanto}{^g}{\^g\allowhyphens}
18.43 \declare@shorthand{esperanto}{^G}{\^G\allowhyphens}
18.44 \declare@shorthand{esperanto}{^h}{h\llap{\^h}\allowhyphens}
18.45 \declare@shorthand{esperanto}{^H}{\^H\allowhyphens}
18.46 \declare@shorthand{esperanto}{^j}{\^j\allowhyphens}
18.47 \declare@shorthand{esperanto}{^J}{\^J\allowhyphens}
18.48 \declare@shorthand{esperanto}{^s}{\^s\allowhyphens}
18.49 \declare@shorthand{esperanto}{^S}{\^S\allowhyphens}
18.50 \declare@shorthand{esperanto}{^u}{\u u\allowhyphens}
18.51 \declare@shorthand{esperanto}{^U}{\u U\allowhyphens}
18.52 \declare@shorthand{esperanto}{^|}{\discretionary{-}{-}{-}\allowhyphens}

```

`\Esper` In `esperant.sty` Jörg Knappen provides the macros `\esper` and `\Esper` that can be used instead of `\alph` and `\Alph`. These macros are available in this file as well.

Their definition takes place in two steps. First the toplevel.

```

18.53 \def\esper#1{\@esper{\@nameuse{c#1}}}
18.54 \def\Eesper#1{\@Esper{\@nameuse{c#1}}}

```

Then the second level.

```

18.55 \def\@esper#1{%
18.56   \ifcase#1\or a\or b\or c\or \^c\or d\or e\or f\or g\or \^g\or
18.57   h\or h\llap{\^h}\or i\or j\or \^j\or k\or l\or m\or n\or o\or
18.58   p\or r\or s\or \^s\or t\or u\or \u{u}\or v\or z\else\@ctrerr\fi}
18.59 \def\@Esper#1{%
18.60   \ifcase#1\or A\or B\or C\or \^C\or D\or E\or F\or G\or \^G\or
18.61   H\or \^H\or I\or J\or \^J\or K\or L\or M\or N\or O\or
18.62   P\or R\or S\or \^S\or T\or U\or \u{U}\or V\or Z\else\@ctrerr\fi}

```

`\hodiaun` In `esperant.sty` Jörg Knappen provides two alternative macros for `\today`, `\hodiaun` and `\hodiaun`. The second macro produces an accusative version of the date in Esperanto.

```

18.63 \addto\dateesperanto{\def\hodiaun{la \today}}
18.64 \def\hodiaun{la \number\day --an~de~\ifcase\month\or
18.65   januaro\or februaro\or marto\or aprilo\or majo\or junio\or
18.66   julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
18.67   decembro\fi, \space \number\year}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

18.68 \ldf@finish{esperanto}
18.69 \code

```

19 The Interlingua language

The file `interlingua.dtx`¹¹ defines all the language definition macros for the Interlingua language. This file was contributed by Peter Kleiweg, kleiweg at let.rug.nl.

Interlingua is an auxiliary language, built from the common vocabulary of Spanish/Portuguese, English, Italian and French, with some normalisation of spelling. The grammar is very easy, more similar to English's than to neolatin languages. The site <http://www.interlingua.com> is mostly written in interlingua (as is <http://interlingua.altervista.org>), in case you want to read some sample of it.

You can have a look at the grammar at <http://www.geocities.com/linguablau>

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
19.1 (*code)
19.2 \LdfInit{interlingua}{captionsinterlingua}
```

When this file is read as an option, i.e. by the `\usepackage` command, `interlingua` could be an 'unknown' language in which case we have to make it known. So we check for the existence of `\l@interlingua` to see whether we have to do something here.

```
19.3 \ifx\undefined\l@interlingua
19.4 \@nopatterns{Interlingua}
19.5 \adddialect\l@interlingua0\fi
```

The next step consists of defining commands to switch to (and from) the Interlingua language.

`\interlinguahyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
19.6 \providehyphenmins{interlingua}{\tw@\tw@}
```

`\captionsinterlingua` The macro `\captionsinterlingua` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
19.7 \def\captionsinterlingua{%
19.8   \def\prefacename{Prefacio}%
19.9   \def\refname{Referentias}%
19.10  \def\abstractname{Summario}%
19.11  \def\bibname{Bibliographia}%
19.12  \def\chaptername{Capitulo}%
19.13  \def\appendixname{Appendice}%
19.14  \def\contentsname{Contento}%
19.15  \def\listfigurename{Lista de figuras}%
19.16  \def\listtablename{Lista de tabellas}%
19.17  \def\indexname{Indice}%
19.18  \def\figurename{Figura}%
19.19  \def\tablename{Tabella}%
19.20  \def\partname{Parte}%
19.21  \def\enclname{Incluso}%
19.22  \def\ccname{Copia}%
19.23  \def\headtoname{A}%
19.24  \def\pagename{Pagina}%
19.25  \def\seename{vide}%
19.26  \def\alsename{vide etiam}%
19.27  \def\proofname{Prova}%
19.28  \def\glossaryname{Glossario}%
19.29 }
```

`\dateinterlingua` The macro `\dateinterlingua` redefines the command `\today` to produce Interlingua dates.

¹¹The file described in this section has version number v1.6 and was last revised on 2005/03/30.

```

19.30 \def\dateinterlingua{%
19.31   \def\today{le~\number\day\space de \ifcase\month\or
19.32     janeiro\or fevereiro\or martio\or april\or maio\or junio\or
19.33     julio\or agosto\or setembro\or outubro\or novembro\or
19.34     dezembro\fi
19.35     \space \number\year}}

```

`\extrasinterlingua` The macro `\extrasinterlingua` will perform all the extra definitions needed for the Interlingua language. The macro `\noextrasinterlingua` is used to cancel the actions of `\extrasinterlingua`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```

19.36 \addto\extrasinterlingua{}
19.37 \addto\noextrasinterlingua{}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

19.38 \ldf@finish{interlingua}
19.39 </code>

```


20 The Dutch language

The file `dutch.dtx`¹² defines all the language-specific macros for the Dutch language and the ‘Afrikaans’ version¹³ of it.

For this language the character " is made active. In table 3 an overview is given of its purpose. One of the reasons for this is that in the Dutch language a word with a dieresis can be hyphenated just before the letter with the umlaut, but the dieresis has to disappear if the word is broken between the previous letter and the accented letter.

In [3] the quoting conventions for the Dutch language are discussed. The preferred convention is the single-quote Anglo-American convention, i.e. ‘This is a quote’. An alternative is the slightly old-fashioned Dutch method with initial double quotes lowered to the baseline, „This is a quote”, which should be typed as "‘This is a quote’”.

"a	\ "a which hyphenates as -a; also implemented for the other letters.
"y	puts a negative kern between i and j
"Y	puts a negative kern between I and J
"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
"~	to produce a hyphencharacter without the following \discretionary{}{}{}.
" "	to produce an invisible ‘breakpoint’.
"‘	lowered double left quotes (see example below).
"’	normal double right quotes.
\-	like the old \-, but allowing hyphenation in the rest of the word.

Table 3: The extra definitions made by `dutch.ldf`

```

20.1 % \changes{dutch-3.8a}{1996/10/04}{made check dependant on
20.2 %   \cs{CurrentOption}}
20.3 %
20.4 %   The macro |\LdfInit| takes care of preventing that this file is
20.5 %   loaded more than once, checking the category code of the
20.6 %   \texttt{@} sign, etc.
20.7 % \changes{dutch-3.8a}{1996/10/30}{Now use \cs{LdfInit} to perform
20.8 %   initial checks}
20.9 %   \begin{macrocode}
20.10 (*code)
20.11 \LdfInit\CurrentOption{captions\CurrentOption}

```

When this file is read as an option, i.e. by the `\usepackage` command, `dutch` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@dutch` or `\l@afrikaans` to see whether we have to do something here.

First we try to establish with which option we are being processed.

```

20.12 \def\bbl@tempa{dutch}
20.13 \ifx\CurrentOption\bbl@tempa

```

If it is `dutch` then we first check if the Dutch hyphenation patterns were loaded,

```

20.14 \ifx\l@dutch\undefined

```

if no we issue a warning and make `dutch` a ‘dialect’ of either the hyphenation patterns that were loaded in slot 0 or of ‘afrikaans’ when it is available.

¹²The file described in this section has version number v3.8i, and was last revised on 2005/03/30.

¹³contributed by Stoffel Lombard (lombc@b31pc87.up.ac.za)

```

20.15 \nopatterns{Dutch}
20.16 \ifx\l@afrikaans\undefined
20.17 \adddialect\l@dutch0
20.18 \else
20.19 \adddialect\l@dutch\l@afrikaans
20.20 \fi
20.21 \fi

```

The next step consists of defining commands to switch to (and from) the Dutch language.

`\captionsdutch` The macro `\captionsdutch` defines all strings used in the four standard document classes provided with L^AT_EX.

```

20.22 \begingroup
20.23 \catcode'\active
20.24 \def\x{\endgroup
20.25 \def\captionsdutch{%
20.26 \def\prefacename{Voorwoord}%
20.27 \def\refname{Referenties}%
20.28 \def\abstractname{Samenvatting}%
20.29 \def\bibname{Bibliografie}%
20.30 \def\chaptername{Hoofdstuk}%
20.31 \def\appendixname{B"ylage}%
20.32 \def\contentsname{Inhoudsopgave}%
20.33 \def\listfigurename{L"yst van figuren}%
20.34 \def\listtablename{L"yst van tabellen}%
20.35 \def\indexname{Index}%
20.36 \def\figurename{Figuur}%
20.37 \def\tablename{Tabel}%
20.38 \def\partname{Deel}%
20.39 \def\enclname{B"ylage(n)}%
20.40 \def\ccname{cc}%
20.41 \def\headtoname{Aan}%
20.42 \def\pagename{Pagina}%
20.43 \def\seename{zie}%
20.44 \def\alsoname{zie ook}%
20.45 \def\proofname{Bew"ys}%
20.46 \def\glossaryname{Verklarende Woordenl"yst}%
20.47 }
20.48 }\x

```

`\datedutch` The macro `\datedutch` redefines the command `\today` to produce Dutch dates.

```

20.49 \def\datedutch{%
20.50 \def\today{\number\day~\ifcase\month\or
20.51 januari\or februari\or maart\or april\or mei\or juni\or
20.52 juli\or augustus\or september\or oktober\or november\or
20.53 december\fi
20.54 \space \number\year}}

```

When the option with which this file is being process was not `dutch` we assume it was `afrikaans`. We perform a similar check on the availability of the hyphenation patterns.

```

20.55 \else
20.56 \ifx\l@afrikaans\undefined
20.57 \nopatterns{Afrikaans}
20.58 \ifx\l@dutch\undefined
20.59 \adddialect\l@afrikaans0
20.60 \else
20.61 \adddialect\l@afrikaans\l@dutch
20.62 \fi
20.63 \fi

```

`\caption safrikaans` Now is the time to define the words for ‘Afrikaans’.

```
20.64 \def\caption safrikaans{%
20.65   \def\prefacename{Voorwoord}%
20.66   \def\refname{Verwysings}%
20.67   \def\abstractname{Samevatting}%
20.68   \def\bibname{Bibliografie}%
20.69   \def\chaptername{Hoofstuk}%
20.70   \def\appendixname{Bylae}%
20.71   \def\contentsname{Inhoudsopgawe}%
20.72   \def\listfigurename{Lys van figure}%
20.73   \def\listtablename{Lys van tabelle}%
20.74   \def\indexname{Inhoud}%
20.75   \def\figurename{Figuur}%
20.76   \def\tablename{Tabel}%
20.77   \def\partname{Deel}%
20.78   \def\enclname{Bylae(n)}%
20.79   \def\ccname{a.a.}%
20.80   \def\headtoname{Aan}%
20.81   \def\pagename{Bladsy}%
20.82   \def\seename{sien}%
20.83   \def\alsoname{sien ook}%
20.84   \def\proofname{Bewys}%
20.85 }
```

`\date afrikaans` Here is the ‘Afrikaans’ version of the date macro.

```
20.86 \def\date afrikaans{%
20.87   \def\today{\number\day~\ifcase\month\or
20.88     Januarie\or Februarie\or Maart\or April\or Mei\or Junie\or
20.89     Julie\or Augustus\or September\or Oktober\or November\or
20.90     Desember\fi
20.91     \space \number\year}}
20.92 \fi
```

`\extras dutch` The macros `\extras dutch` and `\caption safrikaans` will perform all the extra definitions needed for the Dutch language. The macros `\noextras dutch` and `\noextras afrikaans` is used to cancel the actions of `\extras dutch` and `\caption safrikaans`.

For Dutch the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the dutch group of shorthands should be used.

```
20.93 \initiate@active@char{"}
```

Both version of the language use the same set of shorthand definitions although the ‘ij’ is not used in Afrikaans.

```
20.94 \@namedef{extras\CurrentOption}{\languageshorthands{dutch}}
20.95 \expandafter\addto\csname extras\CurrentOption\endcsname{%
20.96   \bbl@activate{"}}
```

The ‘umlaut’ character should be positioned lower on *all* vowels in Dutch texts.

```
20.97 \expandafter\addto\csname extras\CurrentOption\endcsname{%
20.98   \umlautlow\umlautelow}
20.99 \@namedef{noextras\CurrentOption}{%
20.100   \umlauthigh}
```

`\dutch hyphenmins` The dutch hyphenation patterns can be used with `\lefthyphenmin` set to 2 and `\afrikaans hyphenmins` `\righthyphenmin` set to 3.

```
20.101 \providehyphenmins{\CurrentOption}{\tw@\thr@@}
```

`\@trema` In the Dutch language vowels with a trema are treated specially. If a hyphenation occurs before a vowel-plus-trema, the trema should disappear. To be able to do

this we could first define the hyphenation break behaviour for the five vowels, both lowercase and uppercase, in terms of `\discretionary`. But this results in a large `\if`-construct in the definition of the active ". Because we think a user should not use " when he really means something like ' ' we chose not to distinguish between vowels and consonants. Therefore we have one macro `\@trema` which specifies the hyphenation break behaviour for all letters.

```
20.102 \def\@trema#1{\allowhyphens\discretionary{-}{#1}{\{"#1}\allowhyphens}
```

Now we can define the doublequote macros: the tremas,

```
20.103 \declare@shorthand{dutch}{a}{\textormath{\@trema a}{\ddot a}}
20.104 \declare@shorthand{dutch}{e}{\textormath{\@trema e}{\ddot e}}
20.105 \declare@shorthand{dutch}{i}{\textormath
20.106   {\allowhyphens\discretionary{-}{i}{\{"i}\allowhyphens}%
20.107   {\ddot \imath}}
20.108 \declare@shorthand{dutch}{o}{\textormath{\@trema o}{\ddot o}}
20.109 \declare@shorthand{dutch}{u}{\textormath{\@trema u}{\ddot u}}
```

dutch quotes,

```
20.110 \declare@shorthand{dutch}{"'}{
20.111   \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
20.112 \declare@shorthand{dutch}{"'}{
20.113   \textormath{\textquotedblright}{\mbox{\textquotedblright}}}
```

and some additional commands:

```
20.114 \declare@shorthand{dutch}{-}{\nobreak-\bbl@allowhyphens}
20.115 \declare@shorthand{dutch}{~}{\textormath{\leavevmode\hbox{-}}{-}}
20.116 \declare@shorthand{dutch}{|}{
20.117   \textormath{\discretionary{-}{|}{\kern.03em}}{}}
20.118 \declare@shorthand{dutch}{"}{\hskip\z@skip}
20.119 \declare@shorthand{dutch}{y}{\textormath{\ij}}{\ddot y}}
20.120 \declare@shorthand{dutch}{Y}{\textormath{\IJ}}{\ddot Y}}
```

To enable hyphenation in two words, written together but separated by a slash, as in 'uitdrukking/opmerking' we define the command "/.

```
20.121 \declare@shorthand{dutch}{"/}{\textormath
20.122   {\bbl@allowhyphens\discretionary{/}{/}{\bbl@allowhyphens}}}
```

- \- All that is left now is the redefinition of \-. The new version of \- should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of T_EX in this respect is very unfortunate for languages such as Dutch and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that T_EX can generate from the hyphenation patterns.

```
20.123 \expandafter\addto\csname extras\CurrentOption\endcsname{%
20.124   \babel@save\-}
20.125 \expandafter\addto\csname extras\CurrentOption\endcsname{%
20.126   \def\-\{\bbl@allowhyphens\discretionary{-}{-}{\bbl@allowhyphens}}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of @ to its original value.

```
20.127 \ldf@finish\CurrentOption
20.128 \code>
```

21 The English language

The file `english.dtx`¹⁴ defines all the language definition macros for the English language as well as for the American and Australian version of this language. For the Australian version the British hyphenation patterns will be used, if available, for the Canadian variant the American patterns are selected.

For this language currently no special definitions are needed or available.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

21.1 `(*code)`

21.2 `\LdfInit\CurrentOption{date\CurrentOption}`

When this file is read as an option, i.e. by the `\usepackage` command, `english` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@english` to see whether we have to do something here.

We allow for the british english patterns to be loaded as either ‘british’, or ‘UKenglish’. When neither of those is known we try to define `\l@english` as an alias for `\l@american` or `\l@USenglish`.

21.3 `\ifx\l@english\@undefined`

21.4 `\ifx\l@UKenglish\@undefined`

21.5 `\ifx\l@british\@undefined`

21.6 `\ifx\l@american\@undefined`

21.7 `\ifx\l@USenglish\@undefined`

21.8 `\ifx\l@canadian\@undefined`

21.9 `\ifx\l@australian\@undefined`

21.10 `\ifx\l@newzealand\@undefined`

21.11 `\@nopatterns{English}`

21.12 `\addialect\l@english0`

21.13 `\else`

21.14 `\let\l@english\l@newzealand`

21.15 `\fi`

21.16 `\else`

21.17 `\let\l@english\l@australian`

21.18 `\fi`

21.19 `\else`

21.20 `\let\l@english\l@canadian`

21.21 `\fi`

21.22 `\else`

21.23 `\let\l@english\l@USenglish`

21.24 `\fi`

21.25 `\else`

21.26 `\let\l@english\l@american`

21.27 `\fi`

21.28 `\else`

21.29 `\let\l@english\l@british`

21.30 `\fi`

21.31 `\else`

21.32 `\let\l@english\l@UKenglish`

21.33 `\fi`

21.34 `\fi`

Because we allow ‘british’ to be used as the babel option we need to make sure that it will be recognised by `\selectlanguage`. In the code above we have made sure that `\l@english` was defined. Now we want to make sure that `\l@british` and `\l@UKenglish` are defined as well. When either of them is we make them equal to each other, when neither is we fall back to the default, `\l@english`.

21.35 `\ifx\l@british\@undefined`

21.36 `\ifx\l@UKenglish\@undefined`

¹⁴The file described in this section has version number v3.3o and was last revised on 2005/03/30.

```

21.37 \adddialect\l@british\l@english
21.38 \adddialect\l@UKenglish\l@english
21.39 \else
21.40 \let\l@british\l@UKenglish
21.41 \fi
21.42 \else
21.43 \let\l@UKenglish\l@british
21.44 \fi

```

‘American’ is a version of ‘English’ which can have its own hyphenation patterns. The default english patterns are in fact for american english. We allow for the patterns to be loaded as ‘english’ ‘american’ or ‘USenglish’.

```

21.45 \ifx\l@american\@undefined
21.46 \ifx\l@USenglish\@undefined

```

When the patterns are not known as ‘american’ or ‘USenglish’ we add a “dialect”.

```

21.47 \adddialect\l@american\l@english
21.48 \else
21.49 \let\l@american\l@USenglish
21.50 \fi
21.51 \else

```

Make sure that USenglish is known, even if the patterns were loaded as ‘american’.

```

21.52 \ifx\l@USenglish\@undefined
21.53 \let\l@USenglish\l@american
21.54 \fi
21.55 \fi

```

‘Canadian’ english spelling is a hybrid of British and American spelling. Although so far no special ‘translations’ have been reported we allow this file to be loaded by the option `candian` as well.

```

21.56 \ifx\l@canadian\@undefined
21.57 \adddialect\l@canadian\l@american
21.58 \fi

```

‘Australian’ and ‘New Zealand’ english spelling seem to be the same as British spelling. Although so far no special ‘translations’ have been reported we allow this file to be loaded by the options `australian` and `newzealand` as well.

```

21.59 \ifx\l@australian\@undefined
21.60 \adddialect\l@australian\l@british
21.61 \fi
21.62 \ifx\l@newzealand\@undefined
21.63 \adddialect\l@newzealand\l@british
21.64 \fi

```

`\englishhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```

21.65 \providehyphenmins{\CurrentOption}{\tw@\thr@@}

```

The next step consists of defining commands to switch to (and from) the English language.

`\captionseenglish` The macro `\captionseenglish` defines all strings used in the four standard document classes provided with L^AT_EX.

```

21.66 \@namedef{captions\CurrentOption}{%
21.67 \def\prefacename{Preface}%
21.68 \def\refname{References}%
21.69 \def\abstractname{Abstract}%
21.70 \def\bibname{Bibliography}%
21.71 \def\chaptername{Chapter}%
21.72 \def\appendixname{Appendix}%
21.73 \def\contentsname{Contents}%
21.74 \def\listfigurename{List of Figures}%

```

```

21.75 \def\listtablename{List of Tables}%
21.76 \def\indexname{Index}%
21.77 \def\figurename{Figure}%
21.78 \def\tablename{Table}%
21.79 \def\partname{Part}%
21.80 \def\enclname{encl}%
21.81 \def\ccname{cc}%
21.82 \def\headtoname{To}%
21.83 \def\pagename{Page}%
21.84 \def\seename{see}%
21.85 \def\alsoname{see also}%
21.86 \def\proofname{Proof}%
21.87 \def\glossaryname{Glossary}%
21.88 }

```

`\dateenglish` In order to define `\today` correctly we need to know whether it should be ‘english’, ‘australian’, or ‘american’. We can find this out by checking the value of `\CurrentOption`.

```

21.89 \def\bbl@tempa{british}
21.90 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{UK}\fi
21.91 \def\bbl@tempa{UKenglish}
21.92 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{UK}\fi
21.93 \def\bbl@tempa{american}
21.94 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{US}\fi
21.95 \def\bbl@tempa{USenglish}
21.96 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{US}\fi
21.97 \def\bbl@tempa{canadian}
21.98 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{US}\fi
21.99 \def\bbl@tempa{australian}
21.100 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{AU}\fi
21.101 \def\bbl@tempa{newzealand}
21.102 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{AU}\fi

```

The macro `\dateenglish` redefines the command `\today` to produce English dates.

```

21.103 \def\bbl@tempa{UK}
21.104 \ifx\bbl@tempa\bbl@tempb
21.105   \@namedef{date\CurrentOption}{%
21.106     \def\today{\ifcase\day\or
21.107       1st\or 2nd\or 3rd\or 4th\or 5th\or
21.108       6th\or 7th\or 8th\or 9th\or 10th\or
21.109       11th\or 12th\or 13th\or 14th\or 15th\or
21.110       16th\or 17th\or 18th\or 19th\or 20th\or
21.111       21st\or 22nd\or 23rd\or 24th\or 25th\or
21.112       26th\or 27th\or 28th\or 29th\or 30th\or
21.113       31st\fi~\ifcase\month\or
21.114       January\or February\or March\or April\or May\or June\or
21.115       July\or August\or September\or October\or November\or
21.116       December\fi\space \number\year}}

```

`\dateaustralian` Now, test for ‘australian’ or ‘american’.

```

21.117 \else

```

The macro `\dateaustralian` redefines the command `\today` to produce Australian resp. New Zealand dates.

```

21.118 \def\bbl@tempa{AU}
21.119 \ifx\bbl@tempa\bbl@tempb
21.120   \@namedef{date\CurrentOption}{%
21.121     \def\today{\number\day~\ifcase\month\or
21.122       January\or February\or March\or April\or May\or June\or
21.123       July\or August\or September\or October\or November\or
21.124       December\fi\space \number\year}}

```

`\dateamerican` The macro `\dateamerican` redefines the command `\today` to produce American dates.

```
21.125 \else
21.126   \@namedef{date\CurrentOption}{%
21.127     \def\today{\ifcase\month\or
21.128       January\or February\or March\or April\or May\or June\or
21.129       July\or August\or September\or October\or November\or
21.130       December\fi \space\number\day, \number\year}}
21.131 \fi
21.132 \fi
```

`\extrasenglish` The macro `\extrasenglish` will perform all the extra definitions needed for the English language. The macro `\noextrasenglish` is used to cancel the actions of `\extrasenglish`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
21.133 \@namedef{extras\CurrentOption}{}
21.134 \@namedef{noextras\CurrentOption}{}%
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
21.135 \ldf@finish\CurrentOption
21.136 </code>
```


22 The German language

The file `germanb.dtx`¹⁵ defines all the language definition macros for the German language as well as for the Austrian dialect of this language¹⁶.

For this language the character " is made active. In table 4 an overview is given of its purpose. One of the reasons for this is that in the German language some character combinations change when a word is broken between the combination. Also the vertical placement of the umlaut can be controlled this way. The quotes

"a	\ "a, also implemented for the other lowercase and uppercase vowels.
"s	to produce the German ſ (like \ss{ }).
"z	to produce the German ſ (like \ss{ }).
"ck	for ck to be hyphenated as k-k.
"ff	for ff to be hyphenated as ff-f, this is also implemented for l, m, n, p, r and t
"S	for SS to be \uppercase{"s}.
"Z	for SZ to be \uppercase{"z}.
"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
""	like "-", but producing no hyphen sign (for compound words with hyphen, e.g. x-""y).
"~	for a compound word mark without a breakpoint.
"=	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
"‘	for German left double quotes (looks like „).
"’	for German right double quotes.
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).

Table 4: The extra definitions made by `german.ldf`

in table 4 can also be typeset by using the commands in table 5.

\glqq	for German left double quotes (looks like „).
\grqq	for German right double quotes (looks like “).
\glq	for German left single quotes (looks like ,).
\grq	for German right single quotes (looks like ‘).
\flqq	for French left double quotes (similar to <<).
\frqq	for French right double quotes (similar to >>).
\flq	for (French) left single quotes (similar to <).
\frq	for (French) right single quotes (similar to >).
\dq	the original quotes character (").

Table 5: More commands which produce quotes, defined by `german.ldf`

When this file was read through the option `germanb` we make it behave as if `german` was specified.

```

22.1 \def\bbl@tempa{germanb}
22.2 \ifx\CurrentOption\bbl@tempa
22.3   \def\CurrentOption{german}
22.4   \ifx\l@german\@undefined
22.5     \nopatterns{German}

```

¹⁵The file described in this section has version number v2.6m and was last revised on 2008/06/01.

¹⁶This file is a re-implementation of Hubert Partl's `german.sty` version 2.5b, see [4].

```

22.6   \adddialect\l@german0
22.7   \fi
22.8   \let\l@germanb\l@german
22.9   \AtBeginDocument{%
22.10   \let\captionsgermanb\captionsgerman
22.11   \let\dategermanb\dategerman
22.12   \let\extragermanb\extragerman
22.13   \let\noextragermanb\noextragerman
22.14   }
22.15 \fi

```

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```

22.16 (*code)
22.17 \LdfInit\CurrentOption{captions\CurrentOption}

```

When this file is read as an option, i.e., by the `\usepackage` command, `german` will be an ‘unknown’ language, so we have to make it known. So we check for the existence of `\l@german` to see whether we have to do something here.

```

22.18 \ifx\l@german\@undefined
22.19   \@nopatterns{German}
22.20   \adddialect\l@german0
22.21 \fi

```

For the Austrian version of these definitions we just add another language.

```

22.22 \adddialect\l@austrian\l@german

```

The next step consists of defining commands to switch to (and from) the German language.

`\captionsgerman` Either the macro `\captionsgerman` or the macro `\captionsaustrian` will define all strings used in the four standard document classes provided with L^AT_EX.

```

22.23 \@namedef{captions\CurrentOption}{%
22.24   \def\prefacename{Vorwort}%
22.25   \def\refname{Literatur}%
22.26   \def\abstractname{Zusammenfassung}%
22.27   \def\bibname{Literaturverzeichnis}%
22.28   \def\chaptername{Kapitel}%
22.29   \def\appendixname{Anhang}%
22.30   \def\contentsname{Inhaltsverzeichnis}%    % oder nur: Inhalt
22.31   \def\listfigurename{Abbildungsverzeichnis}%
22.32   \def\listtablename{Tabellenverzeichnis}%
22.33   \def\indexname{Index}%
22.34   \def\figurename{Abbildung}%
22.35   \def\tablename{Tabelle}%                % oder: Tafel
22.36   \def\partname{Teil}%
22.37   \def\enclname{Anlage(n)}%              % oder: Beilage(n)
22.38   \def\ccname{Verteiler}%                % oder: Kopien an
22.39   \def\headtoname{An}%
22.40   \def\pagename{Seite}%
22.41   \def\seename{siehe}%
22.42   \def\alsoname{siehe auch}%
22.43   \def\proofname{Beweis}%
22.44   \def\glossaryname{Glossar}%
22.45   }

```

`\dategerman` The macro `\dategerman` redefines the command `\today` to produce German dates.

```

22.46 \def\month@german{\ifcase\month\or
22.47   Januar\or Februar\or M\"arz\or April\or Mai\or Juni\or
22.48   Juli\or August\or September\or Oktober\or November\or Dezember\fi}
22.49 \def\dategerman{\def\today{\number\day.\~\month@german
22.50   \space\number\year}}

```

`\dateaustrian` The macro `\dateaustrian` redefines the command `\today` to produce Austrian version of the German dates.

```
22.51 \def\dateaustrian{\def\today{\number\day.\ifnum1=\month
```

```
22.52 J\"anner\else \month@german\fi \space\number\year}}
```

`\extrasgerman` Either the macro `\extrasgerman` or the macros `\extrasaustrian` will perform all the extra definitions needed for the German language. The macro `\noextrasgerman` is used to cancel the actions of `\extrasgerman`.

`\noextrasaustrian` For German (as well as for Dutch) the " character is made active. This is done once, later on its definition may vary.

```
22.53 \initiate@active@char{"}
```

```
22.54 \@namedef{extras\CurrentOption}{%
```

```
22.55 \languageshorthands{german}}
```

```
22.56 \expandafter\addto\csname extras\CurrentOption\endcsname{%
```

```
22.57 \bbl@activate{"}}
```

Don't forget to turn the shorthands off again.

```
22.58 \addto\noextrasgerman{\bbl@deactivate{"}}
```

In order for T_EX to be able to hyphenate German words which contain ‘ß’ (in the OT1 position \sim Y) we have to give the character a nonzero `\lccode` (see Appendix H, the T_EXbook).

```
22.59 \expandafter\addto\csname extras\CurrentOption\endcsname{%
```

```
22.60 \babel@savevariable{\lccode25}%
```

```
22.61 \lccode25=25}
```

The umlaut accent macro `\` is changed to lower the umlaut dots. The redefinition is done with the help of `\umlautlow`.

```
22.62 \expandafter\addto\csname extras\CurrentOption\endcsname{%
```

```
22.63 \babel@save{"\umlautlow}
```

```
22.64 \@namedef{noextras\CurrentOption}{\umlauthigh}
```

The german hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
22.65 \providehyphenmins{\CurrentOption}{\tw@\tw@}
```

For German texts we need to make sure that `\frenchspacing` is turned on.

```
22.66 \expandafter\addto\csname extras\CurrentOption\endcsname{%
```

```
22.67 \bbl@frenchspacing}
```

```
22.68 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
```

```
22.69 \bbl@nonfrenchspacing}
```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 4.

To be able to define the function of `\`, we first define a couple of ‘support’ macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\` can now be typed as `\`.

```
22.70 \begingroup \catcode'\12
```

```
22.71 \def\x{\endgroup
```

```
22.72 \def\SS{\mathchar"7019 }
```

```
22.73 \def\dq{"}}
```

```
22.74 \x
```

Now we can define the doublequote macros: the umlauts,

```
22.75 \declare@shorthand{german}{a}{\textormath{"{a}\allowhyphens}{\ddot a}}
```

```
22.76 \declare@shorthand{german}{o}{\textormath{"{o}\allowhyphens}{\ddot o}}
```

```
22.77 \declare@shorthand{german}{u}{\textormath{"{u}\allowhyphens}{\ddot u}}
```

```
22.78 \declare@shorthand{german}{A}{\textormath{"{A}\allowhyphens}{\ddot A}}
```

```
22.79 \declare@shorthand{german}{O}{\textormath{"{O}\allowhyphens}{\ddot O}}
```

```
22.80 \declare@shorthand{german}{U}{\textormath{"{U}\allowhyphens}{\ddot U}}
```

tremas,

```

22.81 \declare@shorthand{german}{\e}{\textormath{\{e\}{\ddot e}}
22.82 \declare@shorthand{german}{\E}{\textormath{\{E\}{\ddot E}}
22.83 \declare@shorthand{german}{\i}{\textormath{\{\i\}%
22.84         {\ddot\imath}}}
22.85 \declare@shorthand{german}{\I}{\textormath{\{I\}{\ddot I}}

```

german es-zet (sharp s),

```

22.86 \declare@shorthand{german}{\s}{\textormath{\ss}{\@SS{}}}
22.87 \declare@shorthand{german}{\S}{\SS}
22.88 \declare@shorthand{german}{\z}{\textormath{\ss}{\@SS{}}}
22.89 \declare@shorthand{german}{\Z}{\SZ}

```

german and french quotes,

```

22.90 \declare@shorthand{german}{\`}{\glqq}
22.91 \declare@shorthand{german}{\'}{\grqq}
22.92 \declare@shorthand{german}{\<}{\flqq}
22.93 \declare@shorthand{german}{\>}{\frqq}

```

discretionary commands

```

22.94 \declare@shorthand{german}{\c}{\textormath{\bbl@disc ck}{c}}
22.95 \declare@shorthand{german}{\C}{\textormath{\bbl@disc CK}{C}}
22.96 \declare@shorthand{german}{\F}{\textormath{\bbl@disc F{FF}}{F}}
22.97 \declare@shorthand{german}{\l}{\textormath{\bbl@disc l{ll}}{l}}
22.98 \declare@shorthand{german}{\L}{\textormath{\bbl@disc L{LL}}{L}}
22.99 \declare@shorthand{german}{\m}{\textormath{\bbl@disc m{mm}}{m}}
22.100 \declare@shorthand{german}{\M}{\textormath{\bbl@disc M{MM}}{M}}
22.101 \declare@shorthand{german}{\n}{\textormath{\bbl@disc n{nn}}{n}}
22.102 \declare@shorthand{german}{\N}{\textormath{\bbl@disc N{NN}}{N}}
22.103 \declare@shorthand{german}{\p}{\textormath{\bbl@disc p{pp}}{p}}
22.104 \declare@shorthand{german}{\P}{\textormath{\bbl@disc P{PP}}{P}}
22.105 \declare@shorthand{german}{\r}{\textormath{\bbl@disc r{rr}}{r}}
22.106 \declare@shorthand{german}{\R}{\textormath{\bbl@disc R{RR}}{R}}
22.107 \declare@shorthand{german}{\t}{\textormath{\bbl@disc t{tt}}{t}}
22.108 \declare@shorthand{german}{\T}{\textormath{\bbl@disc T{TT}}{T}}

```

We need to treat "f a bit differently in order to preserve the ff-ligature.

```

22.109 \declare@shorthand{german}{\f}{\textormath{\bbl@discff}{f}}
22.110 \def\bbl@discff{\penalty\@M
22.111 \afterassignment\bbl@insertff \let\bbl@nextff= }
22.112 \def\bbl@insertff{%
22.113 \if f\bbl@nextff
22.114 \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi
22.115 {\relax\discretionary{ff-}{f}{ff}\allowhyphens}{f\bbl@nextff}}
22.116 \let\bbl@nextff=f

```

and some additional commands:

```

22.117 \declare@shorthand{german}{\~}{\nobreak\-\bbl@allowhyphens}
22.118 \declare@shorthand{german}{\|}{\%
22.119 \textormath{\penalty\@M\discretionary{-}{\}{\kern.03em}%
22.120 \allowhyphens}{}}
22.121 \declare@shorthand{german}{\"}{\hskip\z@skip}
22.122 \declare@shorthand{german}{\~}{\textormath{\leavevmode\hbox{-}}{-}}
22.123 \declare@shorthand{german}{\=}{\penalty\@M-\hskip\z@skip}

```

`\mdqon` All that's left to do now is to define a couple of commands for reasons of compatibility with `german.sty`.

```

\ck22.124 \def\mdqon{\shorthandon{}}
22.125 \def\mdqoff{\shorthandoff{}}
22.126 \def\ck{\allowhyphens\discretionary{k-}{k}{ck}\allowhyphens}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
22.127 \ldf@finish\CurrentOption
22.128 </code>
```

23 The German language – new orthography

The file `ngermanb.dtx`¹⁷ defines all the language definition macros for the German language with the ‘new orthography’ introduced in August 1998. This includes also the Austrian dialect of this language.

As with the ‘traditional’ German orthography, the character " is made active, and the commands in table 4 can be used, except for "ck and "ff etc., which are no longer required.

The internal language names are `ngerman` and `naustrian`.

When this file was read through the option `ngermanb` we make it behave as if `ngerman` was specified.

```
23.1 \def\bbl@tempa{ngermanb}
23.2 \ifx\CurrentOption\bbl@tempa
23.3   \def\CurrentOption{ngerman}
23.4 \fi
```

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
23.5 (*code)
23.6 \LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e., by the `\usepackage` command, `ngerman` will be an ‘unknown’ language, so we have to make it known. So we check for the existence of `\l@ngerman` to see whether we have to do something here.

```
23.7 \ifx\l@ngerman\@undefined
23.8   \@nopatterns{ngerman}
23.9   \adddialect\l@ngerman0
23.10 \fi
```

For the Austrian version of these definitions we just add another language.

```
23.11 \adddialect\l@naustrian\l@ngerman
```

The next step consists of defining commands to switch to (and from) the German language.

<code>\captionsngerman</code> <code>\captionsnaustrian</code>	Either the macro <code>\captionsngerman</code> or the macro <code>\captionsnaustrian</code> will define all strings used in the four standard document classes provided with L ^A T _E X.
------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```
23.12 \@namedef{captions\CurrentOption}{%
23.13   \def\prefacename{Vorwort}%
23.14   \def\refname{Literatur}%
23.15   \def\abstractname{Zusammenfassung}%
23.16   \def\bibname{Literaturverzeichnis}%
23.17   \def\chaptername{Kapitel}%
23.18   \def\appendixname{Anhang}%
23.19   \def\contentsname{Inhaltsverzeichnis}%    % oder nur: Inhalt
23.20   \def\listfigurename{Abbildungsverzeichnis}%
23.21   \def\listtablename{Tabellenverzeichnis}%
23.22   \def\indexname{Index}%
23.23   \def\figurename{Abbildung}%
23.24   \def\tablename{Tabelle}%                  % oder: Tafel
23.25   \def\partname{Teil}%
23.26   \def\enclname{Anlage(n)}%                 % oder: Beilage(n)
23.27   \def\ccname{Verteiler}%                   % oder: Kopien an
23.28   \def\headtoname{An}%
23.29   \def\pagename{Seite}%
23.30   \def\seename{siehe}%
23.31   \def\alsaname{siehe auch}%
23.32   \def\proofname{Beweis}%
23.33   \def\glossaryname{Glossar}%
23.34 }
```

¹⁷The file described in this section has version number v2.6n and was last revised on 2008/07/06.

`\datengerman` The macro `\datengerman` redefines the command `\today` to produce German dates.

```
23.35 \def\month@ngerman{\ifcase\month\or
23.36 Januar\or Februar\or M"arz\or April\or Mai\or Juni\or
23.37 Juli\or August\or September\or Oktober\or November\or Dezember\fi}
23.38 \def\datengerman{\def\today{\number\day.\~\month@ngerman
23.39 \space\number\year}}
```

`\dateanaustrian` The macro `\datenaustrian` redefines the command `\today` to produce Austrian version of the German dates.

```
23.40 \def\datenaustrian{\def\today{\number\day.\~\ifnum1=\month
23.41 J"anner\else \month@ngerman\fi \space\number\year}}
```

`\extrasngerman` Either the macro `\extrasngerman` or the macros `\extrasnaustrian` will perform all the extra definitions needed for the German language. The macro `\noextrasngerman` is used to cancel the actions of `\extrasngerman`.
`\noextrasnaustrian` For German (as well as for Dutch) the " character is made active. This is done once, later on its definition may vary.

```
23.42 \initiate@active@char{"}
23.43 \@namedef{extras\CurrentOption}{%
23.44 \languageshorthands{ngerman}}
23.45 \expandafter\addto\csname extras\CurrentOption\endcsname{%
23.46 \bbl@activate{"}}
```

Don't forget to turn the shorthands off again.

```
23.47 \addto\noextrasngerman{\bbl@deactivate{"}}
```

In order for $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ to be able to hyphenate German words which contain ‘ß’ (in the OT1 position $\sim\mathrm{Y}$) we have to give the character a nonzero `\lccode` (see Appendix H, the $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ book).

```
23.48 \expandafter\addto\csname extras\CurrentOption\endcsname{%
23.49 \babel@savevariable{\lccode25}%
23.50 \lccode25=25}
```

The umlaut accent macro `\` is changed to lower the umlaut dots. The redefinition is done with the help of `\umlautlow`.

```
23.51 \expandafter\addto\csname extras\CurrentOption\endcsname{%
23.52 \babel@save{\}\umlautlow}
23.53 \@namedef{noextras\CurrentOption}{\umlauthigh}
```

The current version of the ‘new’ German hyphenation patterns (`dehyphn.tex` is to be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
23.54 \providehyphenmins{\CurrentOption}{\tw@\tw@}
```

For German texts we need to make sure that `\frenchspacing` is turned on.

```
23.55 \expandafter\addto\csname extras\CurrentOption\endcsname{%
23.56 \bbl@frenchspacing}
23.57 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
23.58 \bbl@nonfrenchspacing}
```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 4.

To be able to define the function of `\`, we first define a couple of ‘support’ macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\` can now be typed as `\`.

```
23.59 \begingroup \catcode'\12
23.60 \def\x{\endgroup
23.61 \def\@SS{\mathchar"7019 }
23.62 \def\dq{"}}
23.63 \x
```

Now we can define the doublequote macros: the umlauts,

```
23.64 \declare@shorthand{ngerman}{\a}{\textormath{\{a\}\allowhyphens}{\ddot a}}
23.65 \declare@shorthand{ngerman}{\o}{\textormath{\{o\}\allowhyphens}{\ddot o}}
23.66 \declare@shorthand{ngerman}{\u}{\textormath{\{u\}\allowhyphens}{\ddot u}}
23.67 \declare@shorthand{ngerman}{\A}{\textormath{\{A\}\allowhyphens}{\ddot A}}
23.68 \declare@shorthand{ngerman}{\O}{\textormath{\{O\}\allowhyphens}{\ddot O}}
23.69 \declare@shorthand{ngerman}{\U}{\textormath{\{U\}\allowhyphens}{\ddot U}}
```

tremas,

```
23.70 \declare@shorthand{ngerman}{\e}{\textormath{\{e\}{\ddot e}}}
23.71 \declare@shorthand{ngerman}{\E}{\textormath{\{E\}{\ddot E}}}
23.72 \declare@shorthand{ngerman}{\i}{\textormath{\{i\}}}%
23.73 \ddot{\imath}}
23.74 \declare@shorthand{ngerman}{\I}{\textormath{\{I\}{\ddot I}}}
```

german es-zet (sharp s),

```
23.75 \declare@shorthand{ngerman}{\s}{\textormath{\ss}{\@SS{}}}
23.76 \declare@shorthand{ngerman}{\S}{\SS}
23.77 \declare@shorthand{ngerman}{\z}{\textormath{\ss}{\@SS{}}}
23.78 \declare@shorthand{ngerman}{\Z}{\SZ}
```

german and french quotes,

```
23.79 \declare@shorthand{ngerman}{\'}{\glqq}
23.80 \declare@shorthand{ngerman}{\'}{\grqq}
23.81 \declare@shorthand{ngerman}{\<}{\flqq}
23.82 \declare@shorthand{ngerman}{\>}{\frqq}
```

and some additional commands:

```
23.83 \declare@shorthand{ngerman}{\nobreak}{\allowhyphens}
23.84 \declare@shorthand{ngerman}{\|}{\%}
23.85 \textormath{\penalty\@M\discretionary{-}{-}{\kern.03em}%
23.86 \allowhyphens}{}}
23.87 \declare@shorthand{ngerman}{\hspace}{\hspace}
23.88 \declare@shorthand{ngerman}{\leavevmode}{\leavevmode}
23.89 \declare@shorthand{ngerman}{\hspace}{\hspace}
```

`\mdqon` All that's left to do now is to define a couple of commands for reasons of compatibility with `german.sty`.

```
23.90 \def\mdqon{\shorthandon{}}
23.91 \def\mdqoff{\shorthandoff{}}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
23.92 \ldf@finish\CurrentOption
23.93 \code}
```


24 The Breton language

The file `breton.dtx`¹⁸ defines all the language-specific macros for the Breton language.

There are not really typographic rules for the Breton language. It is a local language (it's one of the celtic languages) which is spoken in Brittany (West of France). So we have a synthesis between french typographic rules and english typographic rules. The characters `:`, `;`, `!` and `?` are made active in order to get a whitespace automatically before these characters.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
24.1 (*code)
24.2 \LdfInit{breton}\captionsbreton
```

When this file is read as an option, i.e. by the `\usepackage` command, `breton` will be an 'unknown' language in which case we have to make it known. So we check for the existence of `\l@breton` to see whether we have to do something here.

```
24.3 \ifx\l@breton\@undefined
24.4     \@nopatterns{Breton}
24.5     \adddialect\l@breton0\fi
```

The next step consists of defining commands to switch to the English language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsbreton` The macro `\captionsbreton` defines all strings used in the four standard document classes provided with L^AT_EX.

```
24.6 \addto\captionsbreton{%
24.7   \def\prefacename{Rakskrid}%
24.8   \def\refname{Daveenno\'u}%
24.9   \def\abstractname{Dvierra\~n}%
24.10  \def\bibname{Lennadurezh}%
24.11  \def\chaptername{Pennad}%
24.12  \def\appendixname{Stagadenn}%
24.13  \def\contentsname{Taolenn}%
24.14  \def\listfigurename{Listenn ar Figurenno\'u}%
24.15  \def\listtablename{Listenn an taolenno\'u}%
24.16  \def\indexname{Meneger}%
24.17  \def\figurename{Figurenn}%
24.18  \def\tablename{Taolenn}%
24.19  \def\partname{Lodenn}%
24.20  \def\enclname{Diello\'u kevret}%
24.21  \def\ccname{Eilskrid da}%
24.22  \def\headtoname{evit}
24.23  \def\pagename{Pajenn}%
24.24  \def\seename{Gwelout}%
24.25  \def\alsoname{Gwelout ivez}%
24.26  \def\proofname{Proof}% <-- needs translation
24.27  \def\glossaryname{Glossary}% <-- Needs translation
24.28 }
```

`\datebreton` The macro `\datebreton` redefines the command `\today` to produce Breton dates.

```
24.29 \def\datebreton{%
24.30   \def\today{\ifnum\day=1\relax 1\/\${\rm a\tilde{n}}\}$\else
24.31     \number\day\fi \space a\space viz\space ifcase\month\or
24.32     Genver\or C'hwevrer\or Meurzh\or Ebrel\or Mae\or Mezheven\or
24.33     Gouere\or Eost\or Gwengolo\or Here\or Du\or Kerzu\fi
24.34     \space\number\year}}
```

¹⁸The file described in this section has version number v1.0h and was last revised on 2005/03/29.

`\extrasbreton` The macro `\extrasbreton` will perform all the extra definitions needed for the Breton language. The macro `\noextrasbreton` is used to cancel the actions of `\extrasbreton`.

The category code of the characters `:`, `;`, `!` and `?` is made `\active` to insert a little white space.

```
24.35 \initiate@active@char{:}
24.36 \initiate@active@char{;}
24.37 \initiate@active@char{!}
24.38 \initiate@active@char{?}
```

We specify that the breton group of shorthands should be used.

```
24.39 \addto\extrasbreton{\languageshorthands{breton}}
```

These characters are ‘turned on’ once, later their definition may vary.

```
24.40 \addto\extrasbreton{%
24.41   \bbl@activate{:}\bbl@activate{;}%
24.42   \bbl@activate{!}\bbl@activate{?}}
```

Don’t forget to turn the shorthands off again.

```
24.43 \addto\noextrasbreton{%
24.44   \bbl@deactivate{:}\bbl@deactivate{;}%
24.45   \bbl@deactivate{!}\bbl@deactivate{?}}
```

The last thing `\extrasbreton` needs to do is to make sure that `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasbreton` will switch it of again.

```
24.46 \addto\extrasbreton{\bbl@frenchspacing}
24.47 \addto\noextrasbreton{\bbl@nonfrenchspacing}
```

`\breton@sh@;` We have to reduce the amount of white space before `;`, `:` and `!` when the user types a space in front of these characters. This should only happen outside mathmode, hence the test with `\ifmmode`.

```
24.48 \declare@shorthand{breton}{:}{;%
24.49   \ifmmode
24.50     \string;\space
24.51   \else\relax
```

In horizontal mode we check for the presence of a ‘space’ and replace it by a `\thinspace`.

```
24.52   \ifhmode
24.53     \ifdim\lastskip>\z@
24.54       \unskip\penalty\@M\thinspace
24.55     \fi
24.56   \fi
24.57   \string;\space
24.58 \fi}%
```

`\breton@sh@:` Because these definitions are very similar only one is displayed in a way that the definition can be easily checked.

`\breton@sh@!:`

```
24.59 \declare@shorthand{breton}{:}{;%
24.60   \ifmmode\string;\space
24.61   \else\relax
24.62   \ifhmode
24.63     \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
24.64   \fi
24.65   \string;\space
24.66 \fi}
24.67 \declare@shorthand{breton}{!}{;%
24.68   \ifmmode\string!\space
24.69   \else\relax
24.70   \ifhmode
24.71     \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
24.72   \fi
```

```

24.73 \string!\space
24.74 \fi}

```

\breton@sh@?@ For the question mark something different has to be done. In this case the amount of white space that replaces the space character depends on the dimensions of the font.

```

24.75 \declare@shorthand{breton}{?}{%
24.76 \ifmmode
24.77 \string?\space
24.78 \else\relax
24.79 \ifhmode
24.80 \ifdim\lastskip>\z@
24.81 \unskip
24.82 \kern\fontdimen2\font
24.83 \kern-1.4\fontdimen3\font
24.84 \fi
24.85 \fi
24.86 \string?\space
24.87 \fi}

```

All that is left to do now is provide the breton user with some extra utilities. Some definitions for special characters.

```

24.88 \DeclareTextSymbol{\at}{OT1}{64}
24.89 \DeclareTextSymbol{\at}{T1}{64}
24.90 \DeclareTextSymbolDefault{\at}{OT1}
24.91 \DeclareTextSymbol{\boi}{OT1}{92}
24.92 \DeclareTextSymbol{\boi}{T1}{16}
24.93 \DeclareTextSymbolDefault{\boi}{OT1}
24.94 \DeclareTextSymbol{\circonflexe}{OT1}{94}
24.95 \DeclareTextSymbol{\circonflexe}{T1}{2}
24.96 \DeclareTextSymbolDefault{\circonflexe}{OT1}
24.97 \DeclareTextSymbol{\tild}{OT1}{126}
24.98 \DeclareTextSymbol{\tild}{T1}{3}
24.99 \DeclareTextSymbolDefault{\tild}{OT1}
24.100 \DeclareTextSymbol{\degre}{OT1}{23}
24.101 \DeclareTextSymbol{\degre}{T1}{6}
24.102 \DeclareTextSymbolDefault{\degre}{OT1}

```

The following macros are used in the redefinition of `\^` and `\"` to handle the letter i.

```

24.103 \AtBeginDocument{%
24.104 \DeclareTextCompositeCommand{\^}{OT1}{i}{\^i}
24.105 \DeclareTextCompositeCommand{\"}{OT1}{i}{\"i}}

```

And some more macros for numbering.

```

24.106 \def\kentan{1/\${}\^{\rm a\tilde{n}}}\$}
24.107 \def\eil{2/\${}\^{\rm l}}\$}
24.108 \def\re{\/\${}\^{\rm re}}\$}
24.109 \def\tredef{3\re}
24.110 \def\pevare{4\re}
24.111 \def\vet{\/\${}\^{\rm vet}}\$}
24.112 \def\pempvet{5\vet}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

24.113 \ldf@finish{breton}
24.114 \code>

```

25 The Welsh language

The file `welsh.dtx`¹⁹ defines all the language definition macros for the Welsh language.

For this language currently no special definitions are needed or available.

The macro `\ldf@init` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
25.1 (*code)
25.2 \LdfInit{welsh}{captionswelsh}
```

When this file is read as an option, i.e. by the `\usepackage` command, `welsh` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@welsh` to see whether we have to do something here.

```
25.3 \ifx\undefined\l@welsh
25.4 \nopatterns{welsh}
25.5 \adddialect\l@welsh0\fi
```

The next step consists of defining commands to switch to (and from) the Welsh language.

`\welshhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
25.6 \providehyphenmins{\CurrentOption}{\tw@\thr@@}
```

`\captionswelsh` The macro `\captionswelsh` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
25.7 \def\captionswelsh{%
25.8   \def\prefacename{Rhagair}%
25.9   \def\refname{Cyfeiriadau}%
25.10  \def\abstractname{Crynodeb}%
25.11  \def\bibname{Llyfryddiaeth}%
25.12  \def\chaptername{Pennod}%
25.13  \def\appendixname{Atodiad}%
25.14  \def\contentsname{Cynnwys}%
25.15  \def\listfigurename{Rhestr Ddarluniau}%
25.16  \def\listtablename{Rhestr Dablau}%
25.17  \def\indexname{Mynegai}%
25.18  \def\figurename{Darlun}%
25.19  \def\tablename{Taflen}%
25.20  \def\partname{Rhan}%
25.21  \def\enclname{amgae"edig}%
25.22  \def\ccname{cop}"i au}%
25.23  \def\headtoname{At}% % ‘at’ on letters meaning ‘to ( a person)’
25.24                      % ‘to (a place)’ is ‘i’ in Welsh
25.25  \def\pagename{tudalen}%
25.26  \def\seename{gweler}%
25.27  \def\alsoname{gweler hefyd}%
25.28  \def\proofname{Prawf}%
25.29  \def\glossaryname{Rhestr termau}%
25.30 }
```

`\datewelsh` The macro `\datewelsh` redefines the command `\today` to produce welsh dates.

```
25.31 \def\datewelsh{%
25.32   \def\today{\ifnum\day=1\relax 1/\${\mathrm{a\tilde{n}}}}$\else
25.33     \number\day\fi\space\ifcase\month\or
25.34     Ionawr\or Chwefror\or Mawrth\or Ebrill\or
25.35     Mai\or Mehefin\or Gorffennaf\or Awst\or
25.36     Medi\or Hydref\or Tachwedd\or Rhagfyr\fi
25.37   \space\number\year}}
```

¹⁹The file described in this section has version number v1.0d and was last revised on 2005/03/31.

`\extraswelsh` The macro `\extraswelsh` will perform all the extra definitions needed for the welsh language. The macro `\noextraswelsh` is used to cancel the actions of `\extraswelsh`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

25.38 `\addto\extraswelsh{}`

25.39 `\addto\noextraswelsh{}`

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

25.40 `\ldf@finish{welsh}`

25.41 `\code{}`

26 The Irish language

The file `irish.dtx`²⁰ defines all the language definition macros for the Irish language.

For this language currently no special definitions are needed or available.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
26.1 (*code)
26.2 \LdfInit{irish}\captionsirish
```

When this file is read as an option, i.e. by the `\usepackage` command, `irish` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@irish` to see whether we have to do something here.

```
26.3 \ifx\l@irish\@undefined
26.4   \@nopatterns{irish}
26.5   \adddialect\l@irish0\fi
```

The next step consists of defining commands to switch to (and from) the Irish language.

`\irishhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
26.6 \providehyphenmins{\CurrentOption}{\tw@\thr@@}
```

`\captionsirish` The macro `\captionsirish` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
26.7 \addto\captionsirish{%
26.8   \def\prefacename{R\'eamhr\'a}%      <-- also "Brollach"
26.9   \def\refname{Tagairt\'{\i}}%
26.10  \def\abstractname{Achoimre}%
26.11  \def\bibname{Leabharliosta}%
26.12  \def\chaptername{Caibidil}%
26.13  \def\appendixname{Aguis\'{\i}n}%
26.14  \def\contentsname{Cl\'ar \'Abhair}%
26.15  \def\listfigurename{L\'ear\'aid\'{\i}}%
26.16  \def\listtablename{T\'abla\'{\i}}%
26.17  \def\indexname{Inn\'eacs}%
26.18  \def\figurename{L\'ear\'aid}%
26.19  \def\tablename{T\'abla}%
26.20  \def\partname{Cuid}%
26.21  \def\enclname{faoi iamh}%
26.22  \def\ccname{cc}%                    abrv. 'c\'oip chuig'
26.23  \def\headtoname{Go}%
26.24  \def\pagename{Leathanach}%
26.25  \def\seename{f\'each}%
26.26  \def\alsoname{f\'each freisin}%
26.27  \def\proofname{Cruth\'unas}%
26.28  \def\glossaryname{Glossary}% <-- Needs translation
26.29 }
```

`\dateirish` The macro `\dateirish` redefines the command `\today` to produce Irish dates.

```
26.30 \def\dateirish{%
26.31   \def\today{%
26.32     \number\day\space \ifcase\month\or
26.33     Ean\'air\or Feabhra\or M\'arta\or Aibre\'an\or
26.34     Bealtaine\or Meitheamh\or I\'uil\or L\'unasa\or
26.35     Me\'an F\'omhair\or Deireadh F\'omhair\or
26.36     M\'{\i} na Samhna\or M\'{\i} na Nollag\fi
26.37     \space \number\year}}
```

²⁰The file described in this section has version number v1.0h and was last revised on 2005/03/30. A contribution was made by Marion Gunn.

`\extrasirish` The macro `\extrasirish` will perform all the extra definitions needed for the Irish language. The macro `\noextrasirish` is used to cancel the actions of `\extrasirish`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

26.38 `\addto\extrasirish{}`

26.39 `\addto\noextrasirish{}`

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

26.40 `\ldf@finish{irish}`

26.41 `\code`

27 The Scottish language

The file `scottish.dtx`²¹ defines all the language definition macros for the Scottish language.

For this language currently no special definitions are needed or available.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
27.1 (*code)
27.2 \LdfInit{scottish}\captionsscottish
```

When this file is read as an option, i.e. by the `\usepackage` command, `scottish` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@scottish` to see whether we have to do something here.

```
27.3 \ifx\l@scottish@undefined
27.4 \nopatterns{scottish}
27.5 \adddialect\l@scottish0\fi
```

The next step consists of defining commands to switch to (and from) the Scottish language.

`\captionsscottish` The macro `\captionsscottish` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
27.6 \addto\captionsscottish{%
27.7 \def\prefacename{Preface}%    <-- needs translation
27.8 \def\refname{Iomraidh}%
27.9 \def\abstractname{Br‘{i}gh}%
27.10 \def\bibname{Leabhraichean}%
27.11 \def\chaptername{Caibideil}%
27.12 \def\appendixname{Ath-sgr‘{i}obhadh}%
27.13 \def\contentsname{Cl‘ar-obrach}%
27.14 \def\listfigurename{Liosta Dhealbh}%
27.15 \def\listtablename{Liosta Chl‘ar}%
27.16 \def\indexname{Cl‘ar-innse}%
27.17 \def\figurename{Dealbh}%
27.18 \def\tablename{Cl‘ar}%
27.19 \def\partname{Cuid}%
27.20 \def\enclname{a-staigh}%
27.21 \def\ccname{lethbhreac gu}%
27.22 \def\headtoname{gu}%
27.23 \def\pagename{t.d.}%          abrv. ‘taobh duilleag’
27.24 \def\seename{see}%    <-- needs translation
27.25 \def\alsename{see also}%    <-- needs translation
27.26 \def\proofname{Proof}%    <-- needs translation
27.27 \def\glossaryname{Glossary}% <-- Needs translation
27.28 }
```

`\datescottish` The macro `\datescottish` redefines the command `\today` to produce Scottish dates.

```
27.29 \def\datescottish{%
27.30 \def\today{%
27.31 \number\day\space \ifcase\month\or
27.32 am Faoilteach\or an Gearran\or am M‘art\or an Giblean\or
27.33 an C‘eitean\or an t-‘Og mhios\or an t-Iuchar\or
27.34 L‘unasdal\or an Sultuine\or an D‘amhar\or
27.35 an t-Samhainn\or an Dubhlachd\fi
27.36 \space \number\year}}
```

`\extrasscottish` The macro `\extrasscottish` will perform all the extra definitions needed for the Scottish language. The macro `\noextrasscottish` is used to cancel the actions of

²¹The file described in this section has version number v1.0g and was last revised on 2005/03/31. A contribution was made by Fraser Grant (FRASER@CERNVM).

`\extrasscottish`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
27.37 \addto\extrasscottish{}  
27.38 \addto\noextrasscottish{}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
27.39 \ldf@finish{scottish}  
27.40 \code}
```

28 The Greek language

The file `greek.dtx`²² defines all the language definition macros for the Greek language, i.e., as it is used today with only one accent, and the attribute *πολυτονικό* (“Polytónico”) for typesetting greek text with all accents. This separation arose out of the need to simplify things, for only very few people will be really interested to typeset polytonic Greek text.

`\greektext` The commands `\greektext` and `\latintext` can be used to switch to greek or latin fonts. These are declarations.
`\latintext`
`\textgreek` The commands `\textgreek` and `\textlatin` both take one argument which is then typeset using the requested font encoding. The command `\greekol` switches to the greek outline font family, while the command `\textol` typesets a short text in outline font. A number of extra greek characters are made available through the added text commands `\stigma`, `\qoppa`, `\sampi`, `\ddigamma`, `\Digamma`, `\euro`, `\permill`, and `\vardigamma`.
`\textlatin`
`\textol`

28.1 Typing conventions

Entering greek text can be quite difficult because of the many diacritical signs that need to be added for various purposes. The fonts that are used to typeset Greek make this a lot easier by offering a lot of ligatures. But in order for this to work, some characters need to be considered as letters. These characters are `<`, `>`, `~`, `‘`, `’`, `"` and `|`. Therefore their `\lccode` is changed when Greek is in effect. In order to let `\uppercase` give correct results, the `\uccode` of these characters is set to a non-existing character to make them disappear. Of course not all characters are needed when typesetting “modern” *μονοτονικό*. In that case we only need the `’` and `"` symbols which are treated in the proper way.

28.2 Greek numbering

The Greek alphabetical numbering system, like the Roman one, is still used in everyday life for short enumerations. Unfortunately most Greeks don’t know how to write Greek numbers bigger than 20 or 30. Nevertheless, in official editions of the last century and beginning of this century this numbering system was also used for dates and numbers in the range of several thousands. Nowadays this numbering system is primarily used by the Eastern Orthodox Church and by certain scholars. It is hence necessary to be able to typeset any Greek numeral up to 999 999. Here are the conventions:

- There is no Greek numeral for any number less than or equal to 0.
- Numbers from 1 to 9 are denoted by letters alpha, beta, gamma, delta, epsilon, stigma, zeta, eta, theta, followed by a mark similar to the mathematical symbol “prime”. (Nowadays instead of letter stigma the digraph sigma tau is used for number 6. Mainly because the letter stigma is not always available, so people opt to write down the first two letters of its name as an alternative. In our implementation we produce the letter stigma, not the digraph sigma tau.)
- Decades from 10 to 90 are denoted by letters iota, kappa, lambda, mu, nu, xi, omikron, pi, qoppa, again followed by the numeric mark. The qoppa used for this purpose has a special zig-zag form, which doesn’t resemble at all the original ‘q’-like qoppa.
- Hundreds from 100 to 900 are denoted by letters rho, sigma, tau, upsilon, phi, chi, psi, omega, sampi, followed by the numeric mark.

²²The file described in this section has version number v1.3l and was last revised on 2005/03/30. The original author is Apostolos Syropoulos (apostolo@platon.ee.duth.gr), code from `kdgreek.sty` by David Kastrup dak@neuroinformatik.ruhr-uni-bochum.de was used to enhance the support for typesetting greek texts.

- Any number between 1 and 999 is obtained by a group of letters denoting the hundreds decades and units, followed by a numeric mark.
- To denote thousands one uses the same method, but this time the mark is placed in front of the letter, and under the baseline (it is inverted by 180 degrees). When a group of letters denoting thousands is followed by a group of letters denoting a number under 1000, then both marks are used.

`\greeknumeral` Using these conventions one obtains numbers up to 999999. The command `\greeknumeral` makes it possible to typeset Greek numerals. There is also an “uppercase” version of this macro: `\Greeknumeral`.

Another system which was in wide use only in Athens, could express any positive number. This system is implemented in package `athnum`.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

28.1 `{*code}`

28.2 `\LdfInit\CurrentOption{captions\CurrentOption}`

When the option `polutonikogreek` was used, redefine `\CurrentOption` to prevent problems later on.

28.3 `\gdef\CurrentOption{greek}%`

When this file is read as an option, i.e. by the `\usepackage` command, `greek` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@greek` to see whether we have to do something here.

28.4 `\ifx\l@greek\@undefined`

28.5 `\@nopatterns{greek}`

28.6 `\adddialect\l@greek0\fi`

Now we declare the `polutoniko` language attribute.

28.7 `\bbl@declare@ttribute{greek}{polutoniko}{%`

This code adds the expansion of `\extrapolutonikogreek` to `\extragreek` and changes the definition of `\today` for Greek to produce polytonic month names.

28.8 `\expandafter\addto\expandafter\extragreek`

28.9 `\expandafter{\extrapolutonikogreek}%`

28.10 `\let\captionsgreek\captionspolutonikogreek`

28.11 `\let\gr@month\gr@c@month`

We need to take some extra precautions in order not to break older documents which still use the old `polutonikogreek` option.

28.12 `\let\l@polutonikogreek\l@greek`

28.13 `\let\datepolutonikogreek\dategreek`

28.14 `\let\extrapolutonikogreek\extragreek`

28.15 `\let\noextrapolutonikogreek\noextragreek`

28.16 `}`

Typesetting Greek texts implies that a special set of fonts needs to be used. The current support for greek uses the `cb` fonts created by Claudio Beccari²³. The `cb` fonts provide all sorts of *font combinations*. In order to use these fonts we define the Local GReek encoding (LGR, see the file `greek.fdd`). We make sure that this encoding is known to L^AT_EX, and if it isn’t we abort.

28.17 `\InputIfFileExists{lgrenc.def}{%`

28.18 `\message{Loading the definitions for the Greek font encoding}}{%`

28.19 `\errhelp{I can’t find the lgrenc.def file for the Greek fonts}}%`

28.20 `\errmessage{Since I do not know what the LGR encoding means^^J`

28.21 `I can’t typeset Greek.^^J`

28.22 `I stop here, while you get a suitable lgrenc.def file}\@end`

28.23 `}`

Now we define two commands that offer the possibility to switch between Greek and Roman encodings.

²³Apostolos Syropoulos wishes to thank him for his patience, collaboration, cooments and suggestions.

`\greektext` The command `\greektext` will switch from Latin font encoding to the Greek font encoding. This assumes that the ‘normal’ font encoding is a Latin one. This command is a *declaration*, for shorter pieces of text the command `\textgreek` should be used.

```
28.24 \DeclareRobustCommand{\greektext}{%
28.25   \fontencoding{LGR}\selectfont
28.26   \def\encodingdefault{LGR}}
```

`\textgreek` This command takes an argument which is then typeset using the requested font encoding. In order to avoid many encoding switches it operates in a local scope.

```
28.27 \DeclareRobustCommand{\textgreek}[1]{\leavevmode{\greektext #1}}
```

`\textol` A last aspect of the set of fonts provided with this version of support for typesetting Greek texts is that it contains an outline family. In order to make it available we define the command `\textol`.

```
28.28 \def\outlfamily{\usefont{LGR}{cmro}{m}{n}}
28.29 \DeclareTextFontCommand{\textol}{\outlfamily}
```

The next step consists in defining commands to switch to (and from) the Greek language.

`\greekhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
28.30 % Yannis Haralambous has suggested this value
28.31 \providehyphenmins{\CurrentOption}{\@ne\@ne}
```

`\captionsgreek` The macro `\captionsgreek` defines all strings used in the four standard document classes provided with L^AT_EX.

```
28.32 \addto\captionsgreek{%
28.33   \def\prefacename{Pr'ologos}%
28.34   \def\refname{Anafor'es}%
28.35   \def\abstractname{Per'ilhyh}%
28.36   \def\bibname{Bibliograf'ia}%
28.37   \def\chaptername{Kef'alaio}%
28.38   \def\appendixname{Par'arthma}%
28.39   \def\contentsname{Perieq'omena}%
28.40   \def\listfigurename{Kat'alogos Sqhm'atwn}%
28.41   \def\listtablename{Kat'alogos Pin'akwn}%
28.42   \def\indexname{Euret'hrio}%
28.43   \def\figurename{Sq'hma}%
28.44   \def\tablename{P'inakas}%
28.45   \def\partname{M'eros}%
28.46   \def\enclname{Sunhmm'ena}%
28.47   \def\ccname{Koinopo'ihsh}%
28.48   \def\headtoname{Pros}%
28.49   \def\pagename{Sel'ida}%
28.50   \def\seename{bl'epe}%
28.51   \def\alsiname{bl'epe ep'ishs}%
28.52   \def\proofname{Ap'odeixh}%
28.53   \def\glossaryname{Glwss'ari}%
28.54 }
```

`\captionspolutonikogreek` For texts written in the *πολυτονικό* (polytonic greek) the translations are the same as above, but some words are spelled differently. For now we just add extra definitions to `\captionsgreek` in order to override the earlier definitions.

```
28.55 \let\captionspolutonikogreek\captionsgreek
28.56 \addto\captionspolutonikogreek{%
28.57   \def\refname{>Anafor'es}%
28.58   \def\indexname{E<uret'hrio}%

```

```

28.59 \def\figurename{Sq~hma}%
28.60 \def\headtoname{Pr'os}%
28.61 \def\alsoname{bl'epe >ep'ishs}%
28.62 \def\proofname{>Ap'odeixh}%
28.63 }

\gr@month The macro \dategreek redefines the command \today to produce greek dates.
\dategreek The name of the month is now produced by the macro \gr@month since it is needed
in the definition of the macro \Grtoday.
28.64 \def\gr@month{%
28.65 \ifcase\month\or
28.66 Ianouar'iou\or Febrouar'iou\or Mart'iou\or April'iou\or
28.67 Ma'"iou\or Ioun'iou\or Ioul'iou\or Augo'ustou\or
28.68 Septembr'iou\or Oktwbr'iou\or Noembr'iou\or Dekembr'iou\fi}
28.69 \def\dategreek{%
28.70 \def\today{\number\day \space \gr@month\space \number\year}}

\gr@c@greek
28.71 \def\gr@c@month{%
28.72 \ifcase\month\or >Ianouar'iou\or
28.73 Febrouar'iou\or Mart'iou\or >April'iou\or Ma'"'iou\or
28.74 >Ioun'iou\or >Ioul'iou\or A>ugo'ustou\or Septembr'iou\or
28.75 >Oktwbr'iou\or Noembr'iou\or Dekembr'iou\fi}

\Grtoday The macro \Grtoday produces the current date, only that the month and the day
are shown as greek numerals instead of arabic as it is usually the case.
28.76 \def\Grtoday{%
28.77 \expandafter\Greeknatural\expandafter{\the\day}\space
28.78 \gr@c@month \space
28.79 \expandafter\Greeknatural\expandafter{\the\year}}

\extrasgreek The macro \extrasgreek will perform all the extra definitions needed for the
\noextrasgreek Greek language. The macro \noextrasgreek is used to cancel the actions of
\extrasgreek. For the moment these macros switch the fontencoding used and
the definition of the internal macros \@alph and \@Alph because in Greek we do
use the Greek numerals.
28.80 \addto\extrasgreek{\greektext}
28.81 \addto\noextrasgreek{\latintext}

\gr@ill@value When the argument of \greeknumeral has a value outside of the acceptable
bounds ( $0 < x < 999999$ ) a warning will be issued (and nothing will be printed).
28.82 \def\gr@ill@value#1{%
28.83 \PackageWarning{babel}{Illegal value (#1) for greeknumeral}}

\anw@true When a a large number with three trailing zero's is to be printed those zeros and
\anw@false the numeric mark need to be discarded. As each 'digit' is processed by a separate
\anw@print macro and because the processing needs to be expandable we need some helper
macros that help remember to not print the numeric mark (\anwtonos).
The command \anw@false switches the printing of the numeric mark off by
making \anw@print expand to nothing. The command \anw@true (re)enables the
printing of the numeric marc. These macro's need to be robust in order to prevent
improper expansion during writing to files or during \uppercase.
28.84 \DeclareRobustCommand\anw@false{%
28.85 \DeclareRobustCommand\anw@print{}}
28.86 \DeclareRobustCommand\anw@true{%
28.87 \DeclareRobustCommand\anw@print{\anwtonos}}
28.88 \anw@true

\greeknumeral The command \greeknumeral needs to be fully expandable in order to get the
right information in auxiliary files. Therefore we use a big \if-construction to
check the value of the argument and start the parsing at the right level.
28.89 \def\greeknumeral#1{%

```

If the value is negative or zero nothing is printed and a warning is issued.

```

28.90 \ifnum#1<\@ne\space\gr@ill@value{#1}%
28.91 \else
28.92 \ifnum#1<10\expandafter\gr@num@i\number#1%
28.93 \else
28.94 \ifnum#1<100\expandafter\gr@num@ii\number#1%
28.95 \else

```

We use the available shorthands for 1.000 ($\@m$) and 10.000 ($\@M$) to save a few tokens.

```

28.96 \ifnum#1<\@m\expandafter\gr@num@iii\number#1%
28.97 \else
28.98 \ifnum#1<\@M\expandafter\gr@num@iv\number#1%
28.99 \else
28.100 \ifnum#1<100000\expandafter\gr@num@v\number#1%
28.101 \else
28.102 \ifnum#1<1000000\expandafter\gr@num@vi\number#1%
28.103 \else

```

If the value is too large, nothing is printed and a warning is issued.

```

28.104 \space\gr@ill@value{#1}%
28.105 \fi
28.106 \fi
28.107 \fi
28.108 \fi
28.109 \fi
28.110 \fi
28.111 \fi
28.112 }

```

\Greeknatural The command `\Greeknatural` prints uppercase greek numerals. The parsing is performed by the macro `\greeknatural`.

```

28.113 \def\Greeknatural#1{%
28.114 \expandafter\MakeUppercase\expandafter{\greeknatural{#1}}

```

\greek@alph In the previous release of this language definition the commands `\greek@aplh` and `\greek@Alph` were kept just for reasons of compatibility. Here again they become meaningful macros. They are defined in a way that even page numbering with greek numerals is possible. Since the macros `\@alph` and `\@Alph` will lose their original meaning while the Greek option is active, we must save their original value. macros `\@alph`

```

28.115 \let\latin@alph\@alph
28.116 \let\latin@Alph\@Alph

```

Then we define the Greek versions; the additional `\expandafters` are needed in order to make sure the table of contents will be correct, e.g., when we have appendixes.

```

28.117 \def\greek@alph#1{\expandafter\greeknatural\expandafter{\the#1}}
28.118 \def\greek@Alph#1{\expandafter\Greeknatural\expandafter{\the#1}}

```

Now we can set up the switching.

```

28.119 \addto\extrasgreek{%
28.120 \let\@alph\greek@alph
28.121 \let\@Alph\greek@Alph}
28.122 \addto\noextrasgreek{%
28.123 \let\@alph\latin@alph
28.124 \let\@Alph\latin@Alph}

```

\greek@roman To prevent roman numerals being typeset in greek letters we need to adopt the internal L^AT_EX commands `\@roman` and `\@Roman`. **Note that this may cause errors where roman ends up in a situation where it needs to be expanded; problems are known to exist with the AMS document classes.**

```

28.125 \let\latin@roman\@roman
28.126 \let\latin@Roman\@Roman
28.127 \def\greek@roman#1{\textlatin{\latin@roman{#1}}}
28.128 \def\greek@Roman#1{\textlatin{\latin@Roman{#1}}}
28.129 \addto\extrasgreek{%
28.130   \let\@roman\greek@roman
28.131   \let\@Roman\greek@Roman}
28.132 \addto\noextrasgreek{%
28.133   \let\@roman\latin@roman
28.134   \let\@Roman\latin@Roman}

```

`\greek&` The greek fonts do not contain an ampersand, so the L^AT_EX command `\&` doesn't
`\ltx&` give the expected result if we do not do something about it.

```

28.135 \let\ltx&\&
28.136 \def\greek&{\textlatin{\ltx&}}
28.137 \addto\extrasgreek{\let\&\greek&}
28.138 \addto\noextrasgreek{\let\&\ltx&}}

```

What is left now is the definition of a set of macros to produce the various digits.

`\gr@num@i` As there is no representation for 0 in this system the zeros are simply discarded.
`\gr@num@ii` When we have a large number with three *trailing* zero's also the numeric mark
`\gr@num@iii` is discarded. Therefore these macros need to pass the information to each other
about the (non-)translation of a zero.

```

28.139 \def\gr@num@i#1{%
28.140   \ifcase#1\or a\or b\or g\or d\or e\or \stigma\or z\or h\or j\fi
28.141   \ifnum#1=\z@\else\anw@true\fi\anw@print}
28.142 \def\gr@num@ii#1{%
28.143   \ifcase#1\or i\or k\or l\or m\or n\or x\or o\or p\or \qoppa\fi
28.144   \ifnum#1=\z@\else\anw@true\fi\gr@num@i}
28.145 \def\gr@num@iii#1{%
28.146   \ifcase#1\or r\or sv\or t\or u\or f\or q\or y\or w\or \sampi\fi
28.147   \ifnum#1=\z@\anw@false\else\anw@true\fi\gr@num@ii}

```

`\gr@num@iv` The first three 'digits' always have the numeric mark, except when one is discarded
`\gr@num@v` because it's value is zero.

```

28.148 \def\gr@num@iv#1{%
28.149   \ifnum#1=\z@\else\katwtonos\fi
28.150   \ifcase#1\or a\or b\or g\or d\or e\or \stigma\or z\or h\or j\fi
28.151   \gr@num@iii}
28.152 \def\gr@num@v#1{%
28.153   \ifnum#1=\z@\else\katwtonos\fi
28.154   \ifcase#1\or i\or k\or l\or m\or n\or x\or o\or p\or \qoppa\fi
28.155   \gr@num@iv}
28.156 \def\gr@num@vi#1{%
28.157   \katwtonos
28.158   \ifcase#1\or r\or sv\or t\or u\or f\or q\or y\or w\or \sampi\fi
28.159   \gr@num@v}

```

`\greek@tilde` In greek typesetting we need a number of characters with more than one accent. In the underlying family of fonts (the cb fonts) this is solved using Knuth's ligature mechanism. Characters we need to have ligatures with are the tilde, the acute and grave accent characters, the rough and smooth breathings, the subscript, and the double quote character. In text input the ~ is normally used to produce an unbreakable space. The command `\~` normally produces a tilde accent. For polytonic Greek we change the definition of `\~` to produce the tilde character itself, making sure it has category code 12.

```

28.160 \begingroup
28.161   \@ifundefined{active@char\string!}{\catcode'\!=12\relax}
28.162   \catcode'\~=12

```

```

28.163 \lccode'\!='\~
28.164 \lowercase{\def\x{\endgroup
28.165 \def\greek@tilde{!}}\x}
28.166 \addto\extrapolutonikogreek{%
28.167 \babel@save\~\let\~\greek@tilde}

```

In order to get correct hyphenation we need to set the lower case code of a number of characters. The ‘v’ character has a special usage for the **cb** fonts: in fact this ligature mechanism detects the end of a word and assures that a final sigma is typeset with the proper sign which is different from that of an initial or medial sigma; the ‘v’ after an *isolated* sigma fools the ligature mechanism in order to typeset σ in place of ς . Because of this we make sure its lowercase code is not changed. For “modern” greek we have to deal only with ‘ and ’ and so things are easy.

```

28.168 \addto\extragreek{%
28.169 \babel@savevariable{\lccode‘v}\lccode‘v=‘v%
28.170 \babel@savevariable{\lccode‘\'}\lccode‘\’=‘\’%
28.171 \babel@savevariable{\lccode‘\'}\lccode‘\’=‘\’%
28.172 \addto\extrapolutonikogreek{%
28.173 \babel@savevariable{\lccode‘<}\lccode‘<=‘<%
28.174 \babel@savevariable{\lccode‘>}\lccode‘>=‘>%
28.175 \babel@savevariable{\lccode‘~}\lccode‘~=‘~%
28.176 \babel@savevariable{\lccode‘|}\lccode‘|=‘|%
28.177 \babel@savevariable{\lccode‘\'}\lccode‘\’=‘\’}

```

And in order to get rid of all accents and breathings when a string is `\uppercase`d we also change a number of uppercase codes.

```

28.178 \addto\extragreek{%
28.179 \babel@savevariable{\uccode‘\'}\uccode‘\’=‘\’%
28.180 \babel@savevariable{\uccode‘\'}\uccode‘\’=159} %% 159 == ~9f
28.181 \addto\extrapolutonikogreek{%
28.182 \babel@savevariable{\uccode‘~}\uccode‘~=159%
28.183 \babel@savevariable{\uccode‘>}\uccode‘>=159%
28.184 \babel@savevariable{\uccode‘<}\uccode‘<=159%
28.185 \babel@savevariable{\uccode‘|}\uccode‘|=‘|%
28.186 \babel@savevariable{\uccode‘\'}\uccode‘\’=159}

```

For this to work we make the character `~9f` a shorthand that expands to nothing. In order for this to work we need to make a character look like `~9f` in T_EX’s eyes. The trick is to have another character and assign it a different lowercase code. Then execute the macros needed in a `\lowercase` environment. Usually the tilde character is used for such purposes. Before we do this we save its original lowercase code to restore it once we’re done.

```

28.187 \@tempcnta=\lccode‘\~
28.188 \lccode‘~=159
28.189 \lowercase{%
28.190 \initiate@active@char{~}%
28.191 \declare@shorthand{greek}{~}{}}
28.192 \lccode‘~= \@tempcnta

```

We can also make the tilde character itself expand to a tilde with category code 12 to make the typing of texts easier.

```

28.193 \addto\extrapolutonikogreek{\languageshorthands{greek}}%
28.194 \declare@shorthand{greek}{~}{\greek@tilde}

```

We now define a few symbols which are used in the typesetting of greek numerals, as well as some other symbols which are useful, such as the $\epsilon\rho\omega$ symbol, etc.

```

28.195 \DeclareTextCommand{\anwtonos}{LGR}{\char"FE\relax}
28.196 \DeclareTextCommand{\katwtonos}{LGR}{\char"FF\relax}
28.197 \DeclareTextCommand{\qoppa}{LGR}{\char"12\relax}
28.198 \DeclareTextCommand{\stigma}{LGR}{\char"06\relax}

```



```

28.199 \DeclareTextCommand{\sampi}{LGR}{\char"1B\relax}
28.200 \DeclareTextCommand{\Digamma}{LGR}{\char"C3\relax}
28.201 \DeclareTextCommand{\ddigamma}{LGR}{\char"93\relax}
28.202 \DeclareTextCommand{\vardigamma}{LGR}{\char"07\relax}
28.203 \DeclareTextCommand{\euro}{LGR}{\char"18\relax}
28.204 \DeclareTextCommand{\permill}{LGR}{\char"19\relax}

```

Since the `~` cannot be used to produce an unbreakable white space we must redefine at least the commands `\fnum@figure` and `\fnum@table` so they do not produce a `~` instead of white space.

```

28.205 %\def\fnum@figure{\figurename\nobreakspace\thefigure}
28.206 %\def\fnum@table{\tablename\nobreakspace\thetable}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

28.207 \ldf@finish{\CurrentOption}
28.208 </code>

```

29 The French language

The file `frenchb.dtx`²⁴, defines all the language definition macros for the French language.

Customisation for the French language is achieved following the book “Lexique des règles typographiques en usage à l’Imprimerie nationale” troisième édition (1994), ISBN-2-11-081075-0.

First version released: 1.1 (1996/05/31) as part of `babel-3.6beta`.

`frenchb` has been improved using helpful suggestions from many people, mainly from Jacques André, Michel Bovani, Thierry Bouche, and Vincent Jalby. Thanks to all of them!

This new version (2.x) has been designed to be used with $\text{\LaTeX} 2_{\epsilon}$ and Plain \TeX formats only. \LaTeX -2.09 is no longer supported. Changes between version 1.6 and ? are listed in subsection 29.4 p. 110.

An extensive documentation is available in French here:

<http://daniel.flipo.free.fr/frenchb>

29.1 Basic interface

In a multilingual document, some typographic rules are language dependent, i.e. spaces before ‘double punctuation’ (: ; ! ?) in French, others concern the general layout (i.e. layout of lists, footnotes, indentation of first paragraphs of sections) and should apply to the whole document.

Starting with version 2.2, `frenchb` behaves differently according to `babel`’s *main language* defined as the *last option*²⁵ at `babel`’s loading. When French is not `babel`’s main language, `frenchb` no longer alters the global layout of the document (even in parts where French is the current language): the layout of lists, footnotes, indentation of first paragraphs of sections are not customised by `frenchb`.

When French is loaded as the last option of `babel`, `frenchb` makes the following changes to the global layout, *both in French and in all other languages*²⁶:

1. the first paragraph of each section is indented (\LaTeX only);
2. the default items in itemize environment are set to ‘—’ instead of ‘•’, and all vertical spacing and glue is deleted; it is possible to change ‘—’ to something else (‘—’ for instance) using `\frenchbsetup{}`;
3. vertical spacing in general \LaTeX lists is shortened;
4. footnotes are displayed “à la française”.

Regarding local typography, the command `\selectlanguage{french}` switches to the French language²⁷, with the following effects:

1. French hyphenation patterns are made active;
2. ‘double punctuation’ (: ; ! ?) is made active²⁸ for correct spacing in French;
3. `\today` prints the date in French;
4. the caption names are translated into French (\LaTeX only);
5. the space after `\dots` is removed in French.

Some commands are provided in `frenchb` to make typesetting easier:

²⁴The file described in this section has version number ? and was last revised on ?.

²⁵Its name is kept in `\bbl@main@language`.

²⁶For each item, hooks are provided to reset standard \LaTeX settings or to emulate the behavior of former versions of `frenchb` (see command `\frenchbsetup{}`, section 29.2).

²⁷`\selectlanguage{francais}` and `\selectlanguage{frenchb}` are kept for backward compatibility but should no longer be used.

²⁸Actually, they are active in the whole document, only their expansions differ in French and outside French

1. French quotation marks can be entered using the commands `\og` and `\fg` which work in $\text{\LaTeX 2}_{\epsilon}$ and PlainTeX , their appearance depending on what is available to draw them; even if you use $\text{\LaTeX 2}_{\epsilon}$ and T1-encoding, you should refrain from entering them as `<<~French quotation marks~>>`: `\og` and `\fg` provide better horizontal spacing. `\og` and `\fg` can be used outside French, they typeset then English quotes “ and ”.
2. A command `\up` is provided to typeset superscripts like `M\up{me}` (abbreviation for “Madame”), `1\up{er}` (for “premier”). Other commands are also provided for ordinals: `\ier`, `\iere`, `\iers`, `\ieres`, `\ieme`, `\iemes` (`3\iemes` prints 3^{es}).
3. Family names should be typeset in small capitals and never be hyphenated, the macro `\bsc` (boxed small caps) does this, e.g., `Leslie~\bsc{Lamport}` will produce Leslie LAMPORT. Note that composed names (such as Dupont-Durant) may now be hyphenated on explicit hyphens, this differs from `frenchb` v.1.x.
4. Commands `\primo`, `\secundo`, `\tertio` and `\quarto` print 1°, 2°, 3°, 4°. `\FrenchEnumerate{6}` prints 6°.
5. Abbreviations for “Numéro(s)” and “numéro(s)” (`N°` `Nos` `n°` and `nos`) are obtained via the commands `\No`, `\Nos`, `\no`, `\nos`.
6. Two commands are provided to typeset the symbol for “degré”: `\degre` prints the raw character and `\degres` should be used to typeset temperatures (e.g., “20~\degres C” with an unbreakable space), or for alcohols’ strengths (e.g., “45\degres” with *no* space in French).
7. In math mode the comma has to be surrounded with braces to avoid a spurious space being inserted after it, in decimal numbers for instance (see the \TeXbook p. 134). The command `\DecimalMathComma` makes the comma be an ordinary character *in French only* (no space added); as a counterpart, if `\DecimalMathComma` is active, an explicit space has to be added in lists and intervals: `$[0,\ 1]$, $(x,\ y)$`. `\StandardMathComma` switches back to the standard behaviour of the comma.
8. A command `\nombre` was provided in 1.x versions to easily format numbers in slices of three digits separated either by a comma in English or with a space in French; `\nombre` is now mapped to `\numprint` from `numprint.sty`, see `numprint.pdf` for more information.
9. `frenchb` has been designed to take advantage of the `xspace` package if present: adding `\usepackage{xspace}` in the preamble will force macros like `\fg`, `\ier`, `\ieme`, `\dots`, ..., to respect the spaces you type after them, for instance typing ‘1\ier juin’ will print ‘1^{er} juin’ (no need for a forced space after 1\ier).

29.2 Customisation

Up to version 1.6, customisation of `frenchb` was achieved by entering commands in `frenchb.cfg`. This possibility remains for compatibility, but *should not longer be used*. Version 2.0 introduces a new command `\frenchbsetup{}` using the `keyval` syntax which should make it easier to choose among the many options available. The command `\frenchbsetup{}` is to appear in the preamble only (after loading `babel`).

`\frenchbsetup{ShowOptions}` prints all available options to the `.log` file, it is just meant as a remainder of the list of offered options. As usual with `keyval` syntax, boolean options (as `ShowOptions`) can be entered as `ShowOptions=true` or just `ShowOptions`, the ‘`=true`’ part can be omitted.

The other options are listed below. Their default value is shown between brackets, sometimes followed by a ‘*’. The ‘*’ means that the default shown applies when **frenchb** is loaded as the *last* option of **babel** —**babel**’s *main language*—, and is toggled otherwise:

- **StandardLayout=true** [**false***] forces **frenchb** not to interfere with the layout: no action on any kind of lists, first paragraphs of sections are not indented (as in English), no action on footnotes. This option replaces the former command `\StandardLayout`. It can be used to avoid conflicts with classes or packages which customise lists or footnotes.
- **GlobalLayoutFrench=false** [**true***] can be used, when French is the main language, to emulate what prior versions of **frenchb** (pre-2.2) did: lists, and first paragraphs of sections will be displayed the standard way in other languages than French, and “à la française” in French. Note that the layout of footnotes is language independent anyway (see below **FrenchFootnotes** and **AutoSpaceFootnotes**). This option replaces the former command `\FrenchLayout`.
- **ReduceListSpacing=false** [**true***]; **frenchb** normally reduces the values of the vertical spaces used in the environment `list` in French; setting this option to **false** reverts to the standard settings of `list`. This option replaces the former command `\FrenchListSpacingfalse`.
- **CompactItemize=false** [**true***]; **frenchb** normally suppresses any vertical space between items of `itemize` lists in French; setting this option to **false** reverts to the standard settings of `itemize` lists. This option replaces the former command `\FrenchItemizeSpacingfalse`.
- **StandardItemLabels=true** [**false***] when set to **true** this option stops **frenchb** from changing the labels in `itemize` lists in French.
- **ItemLabels=\textendash, \textbullet, \ding{43}, ..., [\textendash*]**; when **StandardItemLabels=false** (the default), this option enables to choose the label used in `itemize` lists for all levels. The next three options do the same but each one for one level only. Note that the example `\ding{43}` requires `\usepackage{pifont}`.
- **ItemLabeli=\textendash, \textbullet, \ding{43}, ..., [\textendash*]**
- **ItemLabelii=\textendash, \textbullet, \ding{43}, ..., [\textendash*]**
- **ItemLabeliii=\textendash, \textbullet, \ding{43}, ..., [\textendash*]**
- **ItemLabeliv=\textendash, \textbullet, \ding{43}, ..., [\textendash*]**
- **StandardLists=true** [**false***] forbids **frenchb** to customise any kind of list. Do activate the option **StandardLists** when using classes or packages that customise lists too (`enumitem`, `paralist`, ...) to avoid conflicts. This option is just a shorthand for **ReduceListSpacing=false** and **CompactItemize=false** and **StandardItemLabels=true**.
- **IndentFirst=false** [**true***]; **frenchb** normally forces indentation of the first paragraph of sections. When this option is set to **false**, the first paragraph of will look the same in French and in English (not indented).
- **FrenchFootnotes=false** [**true***] reverts to the standard layout of footnotes. By default **frenchb** typesets leading numbers as ‘1.’ instead of ‘1’, but has no effect on footnotes numbered with symbols (as in the `\thanks` command). The former commands `\StandardFootnotes` and `\FrenchFootnotes` are still there, `\StandardFootnotes` can be useful when some footnotes are numbered with letters (inside minipages for instance).

- `AutoSpaceFootnotes=false [true*]` ; by default `frenchb` adds a thin space in the running text before the number or symbol calling the footnote. Making this option `false` reverts to the standard setting (no space added).
- `FrenchSuperscripts=false [true]` ; then `\up=\textsuperscript` (option added in version 2.1). Should only be made `false` to recompile older documents. By default `\up` now relies on `\fup` designed to produce better looking superscripts.
- `AutoSpacePunctuation=false [true]`; in French, the user *should* input a space before the four characters ‘:;!?’ but as many people forget about it (even among native French writers!), the default behaviour of `frenchb` is to automatically add a `\thinspace` before ‘:’ ‘!’ ‘?’ and a normal (unbreakable) space before ‘:’ (this is recommended by the French Imprimerie nationale). This is convenient in most cases but can lead to addition of spurious spaces in URLs or in MS-DOS paths but only if they are not typed using `\texttt` or verbatim mode. When the current font is a monospaced (typewriter) font, `AutoSpacePunctuation` is locally switched to `false`, no spurious space is added in that case, so the default behaviour of `frenchb` in that area should be fine in most circumstances.
Choosing `AutoSpacePunctuation=false` will ensure that a proper space will be added before ‘:;!?’ *if and only if* a (normal) space has been typed in. Those who are unsure about their typing in this area should stick to the default option and type `\string; \string: \string! \string?` instead of `; : ! ?` in case an unwanted space is added by `frenchb`.
- `ThinColonSpace=true [false]` changes the normal (unbreakable) space added before the colon ‘:’ to a thin space, so that the same amount of space is added before any of the four double punctuation characters. The default setting is supported by the French Imprimerie nationale.
- `LowercaseSuperscripts=false [true]` ; by default `frenchb` inhibits the uppercasing of superscripts (for instance when they are moved to page headers). Making this option `false` will disable this behaviour (not recommended).
- `PartNameFull=false [true]`; when true, `frenchb` numbers the title of `\part{}` commands as “Première partie”, “Deuxième partie” and so on. With some classes which change the `\part{}` command (AMS and SMF classes do so), you will get “Première partie I”, “Deuxième partie II” instead; when this occurs, this option should be set to `false`, part titles will then be printed as “Partie I”, “Partie II”.
- `og=«, fg=»`; when guillemets characters are available on the keyboard (through a compose key for instance), it is nice to use them instead of typing `\og` and `\fg`. This option tells `frenchb` which characters are opening and closing French guillemets (they depend on the input encoding), then you can type either `« guillemets »`, or `«guillemets»` (with or without spaces), to get properly typeset French quotes. This option requires `inputenc` to be loaded with the proper encoding, it works with 8-bits encodings (latin1, latin9, ansinew, applemac, ...) and multi-byte encodings (utf8 and utf8x).

29.3 Hyphenation checks

Once you have built your format, a good precaution would be to perform some basic tests about hyphenation in French. For $\text{\LaTeX}2_{\epsilon}$ I suggest this:

- run the following file, with the encoding suitable for your machine (*my-encoding* will be `latin1` for UNIX machines, `ansinew` for PCs running Windows, `applemac` or `latin1` for Macintoshes, or `utf8`...

```

%% Test file for French hyphenation.
\documentclass{article}
\usepackage[my-encoding]{inputenc}
\usepackage[T1]{fontenc} % Use LM fonts
\usepackage{lmodern}      % for French
\usepackage[frenchb]{babel}
\begin{document}
\showhyphens{signal container \’ev\’enement alg\’ebre}
\showhyphens{signal container événement algèbre}
\end{document}

```

- check the hyphenations proposed by T_EX in your log-file; in French you should get with both 7-bit and 8-bit encodings
`si-gnal contai-ner évé-ne-ment al-gèbre`.
 Do not care about how accented characters are displayed in the log-file, what matters is the position of the ‘-’ hyphen signs *only*.

If they are all correct, your installation (probably) works fine, if one (or more) is (are) wrong, ask a local wizard to see what’s going wrong and perform the test again (or e-mail me about what happens).

Frequent mismatches:

- you get `sig-nal con-tainer`, this probably means that the hyphenation patterns you are using are for US-English, not for French;
- you get no hyphen at all in `évé-ne-ment`, this probably means that you are using CM fonts and the macro `\accent` to produce accented characters. Using 8-bits fonts with built-in accented characters avoids this kind of mismatch.

Options’ order – Please remember that options are read in the order they appear inside the `\frenchbsetup` command. Someone wishing that `frenchb` leaves the layout of lists and footnotes untouched but caring for indentation of first paragraph of sections could choose `\frenchbsetup{StandardLayout,IndentFirst}` and get the expected layout. Choosing `\frenchbsetup{IndentFirst,StandardLayout}` would not lead to the expected result: option `IndentFirst` would be overwritten by `StandardLayout`.

29.4 Changes

What’s new in version 2.0?

Here is the list of all changes:

- Support for L^AT_EX-2.09 and for L^AT_EX 2_ε in compatibility mode has been dropped. This version is meant for L^AT_EX 2_ε and Plain based formats (like `bplain`). L^AT_EX 2_ε formats based on m^LT_EX are no longer supported either (plenty of good 8-bits fonts are available now, so T1 encoding should be preferred for typesetting in French). A warning is issued when OT1 encoding is in use at the `\begin{document}`.
- Customisation should now be handled by command `\frenchbsetup{}`, `frenchb.cfg` (kept for compatibility) should no longer be used. See section 29.2 for the list of available options.
- Captions in figures and table have changed in French: former abbreviations “Fig.” and “Tab.” have been replaced by full names “Figure” and “Table”. If this leads to formatting problems in captions, you can add the following two commands to your preamble (after loading `babel`) to get the former captions
`\addto\captionsfrench{\def\figurename{{\scshape Fig.}}}`
`\addto\captionsfrench{\def\tablename{{\scshape Tab.}}}`.

- The `\nombre` command is now provided by the `numprint` package which has to be loaded *after* `babel` with the option `autolanguage` if number formatting should depend on the current language.
- The `\bsc` command no longer uses an `\hbox` to stop hyphenation of names but a `\kern0pt` instead. This change enables `microtype` to fine tune the length of the argument of `\bsc`; as a side-effect, compound names like Dupont-Durand can now be hyphenated on explicit hyphens. You can get back to the former behaviour of `\bsc` by adding `\renewcommand*{\bsc}[1]{\leavevmode\hbox{\scshape #1}}` to the preamble of your document.
- Footnotes are now displayed “à la française” for the whole document, except with an explicit `\frenchbsetup{AutoSpaceFootnotes=false,FrenchFootnotes=false}`. Add this command if you want standard footnotes. It is still possible to revert locally to the standard layout of footnotes by adding `\StandardFootnotes` (inside a `minipage` environment for instance).

What’s new in version 2.1?

New command `\fup` to typeset better looking superscripts. Former command `\up` is now defined as `\fup`, but an option `\frenchbsetup{FrenchSuperscripts=false}` is provided for backward compatibility. `\fup` was designed using ideas from Jacques André, Thierry Bouche and René Fritz, thanks to them!

What’s new in version 2.2?

Starting with version 2.2a, `frenchb` alters the layout of lists, footnotes, and the indentation of first paragraphs of sections) *only if* French is the “main language” (i.e. `babel`’s last language option). The layout is global for the whole document: lists, etc. look the same in French and in other languages, everything is typeset “à la française” if French is the “main language”, otherwise `frenchb` doesn’t change anything regarding lists, footnotes, and indentation of paragraphs.

What’s new in version 2.3?

Starting with version 2.3a, `frenchb` no longer inserts spaces automatically before ‘:;!?’ when a typewriter font is in use; this was suggested by Yannis Haralambous to prevent spurious spaces in computer source code or expressions like `C:/foo`, `http://foo.bar`, etc. An option (`OriginalTypewriter`) is provided to get back to the former behaviour of `frenchb`.

Another probably invisible change: lowercase conversion in `\up{}` is now achieved by the \LaTeX command `\MakeLowercase` instead of \TeX ’s `\lowercase` command. This prevents error messages when diacritics are used inside `\up{}` (diacritics should *never* be used in superscripts though!).

29.5 File `frenchb.cfg`

`frenchb.cfg` is now a dummy file just kept for compatibility with previous versions.

```

29.1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29.2 %%%%%%%%% WARNING: THIS FILE SHOULD NO LONGER BE USED %%%%%%%%%
29.3 %% If you want to customise frenchb, please DO NOT hack into the code!
29.4 %% Do no put any code in this file either, please use the new command
29.5 %% \frenchbsetup{} with the proper options to customise frenchb.
29.6 %%
29.7 %% Add \frenchbsetup{ShowOptions} to your preamble to see the list of
29.8 %% available options and/or read the documentation.
29.9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

30 T_EXnical details

30.1 Initial setup

While this file was read through the option `frenchb` we make it behave as if `french` was specified.

```
30.1 \def\CurrentOption{french}
```

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
30.2 \LdfInit\CurrentOption\datefrench
```

```
\ifLaTeXe No support is provided for late LATEX-2.09: issue a warning and exit if LATEX-2.09
is in use. Plain is still supported.
```

```
30.3 \newif\ifLaTeXe
```

```
30.4 \let\bbl@tempa\relax
```

```
30.5 \ifx\magnification\@undefined
```

```
30.6 \ifx\@compatibilitytrue\@undefined
```

```
30.7 \PackageError{frenchb.ldf}
```

```
30.8 {LaTeX-2.09 format is no longer supported.\MessageBreak
```

```
30.9 Aborting here}
```

```
30.10 {Please upgrade to LaTeX2e!}
```

```
30.11 \let\bbl@tempa\endinput
```

```
30.12 \else
```

```
30.13 \LaTeXettrue
```

```
30.14 \fi
```

```
30.15 \fi
```

```
30.16 \bbl@tempa
```

Check if hyphenation patterns for the French language have been loaded in `language.dat`; we allow for the names ‘`french`’, ‘`français`’, ‘`canadien`’ or ‘`acadian`’. The latter two are both names used in Canada for variants of French that are in use in that country.

```
30.17 \ifx\l@french\@undefined
```

```
30.18 \ifx\l@français\@undefined
```

```
30.19 \ifx\l@canadien\@undefined
```

```
30.20 \ifx\l@acadian\@undefined
```

```
30.21 \@nopatterns{French}
```

```
30.22 \addialect\l@french0
```

```
30.23 \else
```

```
30.24 \let\l@french\l@acadian
```

```
30.25 \fi
```

```
30.26 \else
```

```
30.27 \let\l@french\l@canadien
```

```
30.28 \fi
```

```
30.29 \else
```

```
30.30 \let\l@french\l@français
```

```
30.31 \fi
```

```
30.32 \fi
```

Now `\l@french` is always defined.

The internal name for the French language is `french`; `français` and `frenchb` are synonymous for `french`: first let both names use the same hyphenation patterns. Later we will have to set aliases for `\captionfrench`, `\datefrench`, `\extrasfrench` and `\noextrasfrench`. As French uses the standard values of `\lefthyphenmin` (2) and `\righthyphenmin` (3), no special setting is required here.

```
30.33 \ifx\l@français\@undefined
```

```
30.34 \let\l@français\l@french
```

```
30.35 \fi
```

```
30.36 \ifx\l@frenchb\@undefined
```

```
30.37 \let\l@frenchb\l@french
```

```
30.38 \fi
```


When this language definition file was loaded for one of the Canadian versions of French we need to make sure that a suitable hyphenation pattern register will be found by \TeX .

```
30.39 \ifx\l@canadien\@undefined
30.40   \let\l@canadien\l@french
30.41 \fi
30.42 \ifx\l@acadian\@undefined
30.43   \let\l@acadian\l@french
30.44 \fi
```

This language definition can be loaded for different variants of the French language. The ‘key’ babel macros are only defined once, using ‘french’ as the language name, but `frenchb` and `français` are synonymous.

```
30.45 \def\datefrancais{\datefrench}
30.46 \def\datefrenchb{\datefrench}
30.47 \def\extrasfrancais{\extrasfrench}
30.48 \def\extrasfrenchb{\extrasfrench}
30.49 \def\noextrasfrancais{\noextrasfrench}
30.50 \def\noextrasfrenchb{\noextrasfrench}
```

`\extrasfrench` The macro `\extrasfrench` will perform all the extra definitions needed for the French language. The macro `\noextrasfrench` is used to cancel the actions of `\extrasfrench`.

In French, character “apostrophe” is a letter in expressions like `l’ambulance` (French hyphenation patterns provide entries for this kind of words). This means that the `\lccode` of “apostrophe” has to be non null in French for proper hyphenation of those expressions, and has to be reset to null when exiting French.

```
30.51 \@namedef{extras\CurrentOption}{\lccode‘\’=‘\’}
30.52 \@namedef{noextras\CurrentOption}{\lccode‘\’=0}
```

One more thing `\extrasfrench` needs to do is to make sure that `\frenchspacing` is in effect. `\noextrasfrench` will switch `\frenchspacing` off again.

```
30.53 \expandafter\addto\csname extras\CurrentOption\endcsname{%
30.54   \bbl@frenchspacing}
30.55 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
30.56   \bbl@nonfrenchspacing}
```

30.2 Punctuation

As long as no better solution is available ²⁹, the ‘double punctuation’ characters (`;` `!` `?` and `:`) have to be made `\active` for an automatic control of the amount of space to insert before them. Before doing so, we have to save the standard definition of `\@makecaption` (which includes two ‘:’) to compare it later to its definition at the `\begin{document}`.

```
30.57 \long\def\STD@makecaption#1#2{%
30.58   \vskip\abovecaptionskip
30.59   \sbox\@tempboxa{#1: #2}%
30.60   \ifdim \wd\@tempboxa >\hsize
30.61     #1: #2\par
30.62   \else
30.63     \global \@minipagefalse
30.64     \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
30.65   \fi
30.66   \vskip\belowcaptionskip}%

```

We define a new ‘if’ `\FBpunct@active` which will be made false whenever a better alternative will be available. Another ‘if’ `\FBAutoSpacePunctuation` needs to be defined now.

```
30.67 \newif\ifFBpunct@active          \FBpunct@activetrue
30.68 \newif\ifFBAutoSpacePunctuation \FBAutoSpacePunctuationtrue
```

²⁹Lua \TeX , or pdf \TeX might provide alternatives in the future...

The following code makes the four characters ; ! ? and : ‘active’ and provides their definitions.

```
30.69 \iffBpunct@active
30.70 \initiate@active@char{:}
30.71 \initiate@active@char{;}
30.72 \initiate@active@char{!}
30.73 \initiate@active@char{?}
```

We first tune the amount of space before ; ! ? and :. This should only happen in horizontal mode, hence the test `\ifhmode`.

In horizontal mode, if a space has been typed before ‘;’ we remove it and put an unbreakable `\thinspace` instead. If no space has been typed, we add `\FDP@thinspace` which will be defined, up to the user’s wishes, as an automatic added thin space, or as `\@empty`.

```
30.74 \declare@shorthand{french}{;}{}%
30.75 \ifhmode
30.76 \ifdim\lastskip>\z@
30.77 \unskip\penalty\@M\thinspace
30.78 \else
30.79 \FDP@thinspace
30.80 \fi
30.81 \fi
```

Now we can insert a ; character.

```
30.82 \string;}
```

The next three definitions are very similar.

```
30.83 \declare@shorthand{french}{!}{}%
30.84 \ifhmode
30.85 \ifdim\lastskip>\z@
30.86 \unskip\penalty\@M\thinspace
30.87 \else
30.88 \FDP@thinspace
30.89 \fi
30.90 \fi
30.91 \string!}
30.92 \declare@shorthand{french}{?}{}%
30.93 \ifhmode
30.94 \ifdim\lastskip>\z@
30.95 \unskip\penalty\@M\thinspace
30.96 \else
30.97 \FDP@thinspace
30.98 \fi
30.99 \fi
30.100 \string?}
```

According to the I.N. specifications, the ‘:’ requires a normal space before it, but some people prefer a `\thinspace` (just like the other three). We define `\Fcolonspace` to hold the required amount of space (user customisable).

```
30.101 \newcommand*{\Fcolonspace}{\space}
30.102 \declare@shorthand{french}{:}{}%
30.103 \ifhmode
30.104 \ifdim\lastskip>\z@
30.105 \unskip\penalty\@M\Fcolonspace
30.106 \else
30.107 \FDP@colonspace
30.108 \fi
30.109 \fi
30.110 \string:}
```

`\AutoSpaceBeforeFDP` `\FDP@thinspace` and `\FDP@space` are defined as unbreakable spaces by `\autospace@beforeFDP`
`\NoAutoSpaceBeforeFDP` or as `\@empty` by `\noautospace@beforeFDP` (internal commands), user commands
`\AutoSpaceBeforeFDP` and `\NoAutoSpaceBeforeFDP` do the same and take care

of the flag `\ifFBAutoSpacePunctuation` in \LaTeX . Set the default now for Plain (done later for \LaTeX).

```

30.111 \def\autospace@beforeFDP{%
30.112     \def\FDP@thinspace{\penalty\@M\thinspace}%
30.113     \def\FDP@colonspace{\penalty\@M\Fcolonspace}}
30.114 \def\noautospace@beforeFDP{\let\FDP@thinspace\@empty
30.115     \let\FDP@colonspace\@empty}
30.116 \ifLaTeXe
30.117     \def\AutoSpaceBeforeFDP{\autospace@beforeFDP
30.118         \FBAutoSpacePunctuationtrue}
30.119     \def\NoAutoSpaceBeforeFDP{\noautospace@beforeFDP
30.120         \FBAutoSpacePunctuationfalse}
30.121 \else
30.122     \let\AutoSpaceBeforeFDP\autospace@beforeFDP
30.123     \let\NoAutoSpaceBeforeFDP\noautospace@beforeFDP
30.124     \AutoSpaceBeforeFDP
30.125 \fi

```

In \LaTeX 2_ϵ `\ttfamily` (and hence `\texttt`) will be redefined ‘AtBeginDocument’ as `\ttfamilyFB` so that no space is added before the four `; : ! ?` characters, even if `AutoSpacePunctuation` is true. `\rmfamily` and `\sffamily` need to be redefined also (`\ttfamily` is not always used inside a group, its effect can be cancelled by `\rmfamily` or `\sffamily`).

These redefinitions can be canceled if necessary, for instance to recompile older documents, see option `OriginalTypewriter` below.

```

30.126 \ifLaTeXe
30.127     \let\ttfamilyORI\ttfamily
30.128     \let\rmfamilyORI\rmfamily
30.129     \let\sffamilyORI\sffamily
30.130     \DeclareRobustCommand\ttfamilyFB{%
30.131         \noautospace@beforeFDP\ttfamilyORI}%
30.132     \DeclareRobustCommand\rmfamilyFB{%
30.133         \ifFBAutoSpacePunctuation
30.134             \autospace@beforeFDP\rmfamilyORI
30.135         \else
30.136             \noautospace@beforeFDP\rmfamilyORI
30.137         \fi}%
30.138     \DeclareRobustCommand\sffamilyFB{%
30.139         \ifFBAutoSpacePunctuation
30.140             \autospace@beforeFDP\sffamilyORI
30.141         \else
30.142             \noautospace@beforeFDP\sffamilyORI
30.143         \fi}%
30.144 \fi

```

When the active characters appear in an environment where their French behaviour is not wanted they should give an ‘expected’ result. Therefore we define shorthands at system level as well.

```

30.145 \declare@shorthand{system}{:}{\string:}
30.146 \declare@shorthand{system}{!}{\string!}
30.147 \declare@shorthand{system}{?}{\string?}
30.148 \declare@shorthand{system}{;}{\string;}

```

We specify that the French group of shorthands should be used.

```

30.149 \addto\extrasfrench{%
30.150     \languageshorthands{french}%

```

These characters are ‘turned on’ once, later their definition may vary. Don’t misunderstand the following code: they keep being active all along the document, even when leaving French.

```

30.151 \bbl@activate{:}\bbl@activate{;}%
30.152 \bbl@activate{!}\bbl@activate{?}%

```

```

30.153 }
30.154 \addto\noextrasfrench{%
30.155 \bbl@deactivate{:}\bbl@deactivate{;}%
30.156 \bbl@deactivate{!}\bbl@deactivate{?}}
30.157 \fi

```

30.3 French quotation marks

`\og` The top macros for quotation marks will be called `\og` (“ouvrez guillemets”) and `\fg` (“fermez guillemets”). Another option for typesetting quotes in multilingual texts is to use the package `csquotes.sty` and its command `\enquote`.

```

30.158 \newcommand*{\og}{\@empty}
30.159 \newcommand*{\fg}{\@empty}

```

`\guillemotleft` L^AT_EX users are supposed to use 8-bit output encodings (T1, LY1,...) to typeset French, those who still stick to OT1 should call `aeguill.sty` or a similar package. In both cases the commands `\guillemotleft` and `\guillemotright` will print the French opening and closing quote characters from the output font. For XeLaTeX, `\guillemotleft` and `\guillemotright` are defined by package `xunicode.sty`. We will check ‘AtBeginDocument’ that the proper output encodings are in use (see end of section 30.13).

We give the following definitions for Plain users only as a (poor) fall-back, they are welcome to change them for anything better.

```

30.160 \ifLaTeXe
30.161 \else
30.162 \ifx\guillemotleft\@undefined
30.163 \def\guillemotleft{\leavevmode\raise0.25ex
30.164 \hbox{$\scriptscriptstyle\l1$}}
30.165 \fi
30.166 \ifx\guillemotright\@undefined
30.167 \def\guillemotright{\raise0.25ex
30.168 \hbox{$\scriptscriptstyle\gg$}}
30.169 \fi
30.170 \let\xspace\relax
30.171 \fi

```

The next step is to provide correct spacing after `\guillemotleft` and before `\guillemotright`: a space precedes and follows quotation marks but no line break is allowed neither *after* the opening one, nor *before* the closing one. `\FBguill@spacing` which does the spacing, has been fine tuned by Thierry Bouche. French quotes (including spacing) are printed by `\FB@og` and `\FB@fg`, the expansion of the top level commands `\og` and `\fg` is different in and outside French. We’ll try to be smart to users of David Carlisle’s `xspace` package: if this package is loaded there will be no need for `{}` or `\` to get a space after `\fg`, otherwise `\xspace` will be defined as `\relax` (done at the end of this file).

```

30.172 \newcommand*{\FBguill@spacing}{\penalty\@M\hskip.8\fontdimen2\font
30.173 plus.3\fontdimen3\font
30.174 minus.8\fontdimen4\font}
30.175 \DeclareRobustCommand*{\FB@og}{\leavevmode
30.176 \guillemotleft\FBguill@spacing}
30.177 \DeclareRobustCommand*{\FB@fg}{\ifdim\lastskip>\z@ \unskip\fi
30.178 \FBguill@spacing\guillemotright\xspace}

```

The top level definitions for French quotation marks are switched on and off through the `\extrasfrench` `\noextrasfrench` mechanism. Outside French, `\og` and `\fg` will typeset standard English opening and closing double quotes.

```

30.179 \ifLaTeXe
30.180 \def\bbl@frenchguillemets{\renewcommand*{\og}{\FB@og}%
30.181 \renewcommand*{\fg}{\FB@fg}}
30.182 \def\bbl@nonfrenchguillemets{\renewcommand*{\og}{\textquotedblleft}%

```

```

30.183      \renewcommand*{\fg}{\ifdim\lastskip>\z@\unskip\fi
30.184      \textquotedblright}}
30.185 \else
30.186   \def\bbl@frenchguillemets{\let\og\FB@og
30.187   \let\fg\FB@fg}
30.188   \def\bbl@nonfrenchguillemets{\def\og{'}%
30.189   \def\fg{\ifdim\lastskip>\z@\unskip\fi ''}}
30.190 \fi
30.191 \expandafter\addto\csname extras\CurrentOption\endcsname{%
30.192   \bbl@frenchguillemets}
30.193 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
30.194   \bbl@nonfrenchguillemets}

```

30.4 Date in French

`\datefrench` The macro `\datefrench` redefines the command `\today` to produce French dates.

```

30.195 \@namedef{date\CurrentOption}{%
30.196   \def\today{{\number\day}\ifnum1=\day {\ier}\fi \space
30.197   \ifcase\month
30.198     \or janvier\or f'evrier\or mars\or avril\or mai\or juin\or
30.199     juillet\or ao\ut\or septembre\or octobre\or novembre\or
30.200     d'ecembre\fi
30.201   \space \number\year}}

```

30.5 Extra utilities

Let's provide the French user with some extra utilities.

`\up` `\up` eases the typesetting of superscripts like '1^{er}'. Up to version 2.0 of `frenchb` `\fup` was just a shortcut for `\textsuperscript` in L^AT_EX 2_ε, but several users complained that `\textsuperscript` typesets superscripts too high and too big, so we now define `\fup` as an attempt to produce better looking superscripts. `\up` is defined as `\fup` but can be redefined by `\frenchbsetup{FrenchSuperscripts=false}` as `\textsuperscript` for compatibility with previous versions.

When a font has built-in superscripts, the best thing to do is to just use them, otherwise `\fup` has to simulate superscripts by scaling and raising ordinary letters. Scaling is done using package `scalefnt` which will be loaded at the end of `babel`'s loading (`frenchb` being an option of `babel`, it cannot load a package while being read).

```

30.202 \newif\ifFB@poorman
30.203 \newdimen\FB@Mht
30.204 \ifLaTeXe
30.205   \AtEndOfPackage{\RequirePackage{scalefnt}}

```

`\FB@up@fake` holds the definition of fake superscripts. The scaling ratio is 0.65, raising is computed to put the top of lower case letters (like 'm') just under the top of upper case letters (like 'M'), precisely 12% down. The chosen settings look correct for most fonts, but can be tuned by the end-user if necessary by changing `\FBsupR` and `\FBsupS` commands.

`\FB@lc` is defined as `\MakeLowercase` to inhibit the uppcasing of superscripts (this may happen in page headers with the standard classes but is wrong); `\FB@lc` can be redefined to do nothing by option `LowercaseSuperscripts=false` of `\frenchbsetup{}`.

```

30.206   \newcommand*{\FBsupR}{-0.12}
30.207   \newcommand*{\FBsupS}{0.65}
30.208   \newcommand*{\FB@lc}[1]{\MakeLowercase{#1}}
30.209   \DeclareRobustCommand*{\FB@up@fake}[1]{%
30.210     \settoheight{\FB@Mht}{M}%
30.211     \addtolength{\FB@Mht}{\FBsupR \FB@Mht}%
30.212     \addtolength{\FB@Mht}{-\FBsupS ex}%

```

```

30.213 \raisebox{\FB@Mht}{\scalefont{\FBsupS}{\FB@lc{#1}}}%
30.214 }

```

The only packages I currently know to take advantage of real superscripts are a) `xltextra` used in conjunction with XeLaTeX and OpenType fonts having the font feature 'VerticalPosition=Superior' (`xltextra` defines `\realsuperscript` and `\fakesuperscript`) and b) `fourier` (from version 1.6) when Expert Utopia fonts are available.

`\FB@up` checks whether the current font is a Type1 'Expert' (or 'Pro') font with real superscripts or not (the code works currently only with `fourier-1.6` but could work with any Expert Type1 font with built-in superscripts, see below), and decides to use real or fake superscripts. It works as follows: the content of `\f@family` (family name of the current font) is split by `\FB@split` into two pieces, the first three characters ('fut' for Fourier, 'ppl' for Adobe's Palatino, ...) stored in `\FB@firstthree` and the rest stored in `\FB@suffix` which is expected to be 'x' or 'j' for expert fonts.

```

30.215 \def\FB@split#1#2#3#4@nil{\def\FB@firstthree{#1#2#3}%
30.216 \def\FB@suffix{#4}}
30.217 \def\FB@x{x}
30.218 \def\FB@j{j}
30.219 \DeclareRobustCommand*\FB@up}[1]{%
30.220 \bgroup \FB@poormantrue
30.221 \expandafter\FB@split\f@family@nil

```

Then `\FB@up` looks for a .fd file named `t1fut-sup.fd` (Fourier) or `t1ppl-sup.fd` (Palatino), etc. supposed to define the subfamily (fut-sup or ppl-sup, etc.) giving access to the built-in superscripts. If the .fd file is not found by `\IfFileExists`, `\FB@up` falls back on fake superscripts, otherwise `\FB@suffix` is checked to decide whether to use fake or real superscripts.

```

30.222 \edef\reserved@a{\lowercase{%
30.223 \noexpand\IfFileExists{\f@encoding\FB@firstthree -sup.fd}}}%
30.224 \reserved@a
30.225 {\ifx\FB@suffix\FB@x \FB@poormanfalse\fi
30.226 \ifx\FB@suffix\FB@j \FB@poormanfalse\fi
30.227 \if\FB@poorman \FB@up@fake{#1}%
30.228 \else \FB@up@real{#1}%
30.229 \fi}%
30.230 {\FB@up@fake{#1}}%
30.231 \egroup}

```

`\FB@up@real` just picks up the superscripts from the subfamily (and forces lower-case).

```

30.232 \newcommand*\FB@up@real}[1]{\bgroup
30.233 \fontfamily\FB@firstthree -sup}\selectfont \FB@lc{#1}\egroup}

```

`\fup` is now defined as `\FB@up` unless `\realsuperscript` is defined (occurs with XeLaTeX calling `xltextra.sty`).

```

30.234 \DeclareRobustCommand*\fup}[1]{%
30.235 \@ifundefined{realsuperscript}%
30.236 {\FB@up{#1}}%
30.237 {\bgroup\let\fakesuperscript\FB@up@fake
30.238 \realsuperscript\FB@lc{#1}\egroup}}

```

Temporary definition of `up` (redefined 'AtBeginDocument').

```

30.239 \newcommand*\up{\relax}

```

Poor man's definition of `\up` for Plain. In $\text{\LaTeX 2}_{\epsilon}$, `\up` will be defined as `\fup` or `\textsuperscript` later on while processing the options of `\frenchbsetup`.

```

30.240 \else
30.241 \newcommand*\up}[1]{\leavevmode\raise1ex\hbox{\sevenrm #1}}
30.242 \fi

```

```

\ieme Some handy macros for those who don't know how to abbreviate ordinals:
\ier
\iere
\iemes
\iers
\ieres

```

```

30.243 \def\ieme{\up{\lowercase{e}}\xspace}
30.244 \def\iemes{\up{\lowercase{es}}\xspace}
30.245 \def\ier{\up{\lowercase{er}}\xspace}
30.246 \def\iers{\up{\lowercase{ers}}\xspace}
30.247 \def\iere{\up{\lowercase{re}}\xspace}
30.248 \def\ieres{\up{\lowercase{res}}\xspace}

```

\No And some more macros relying on \up for numbering, first two support macros.

```

\newcommand*\FrenchEnumerate[1]{%
  \nos30.250 #1\up{\lowercase{o}}\kern+.3em}
\newcommand*\FrenchPopularEnumerate[1]{%
  \nos30.251 #1\up{\lowercase{o}}\kern+.3em}
\primo30.252
\primo) Typing \primo should result in ‘1°’,
30.253 \def\primo{\FrenchEnumerate1}
30.254 \def\secundo{\FrenchEnumerate2}
30.255 \def\tertio{\FrenchEnumerate3}
30.256 \def\quarto{\FrenchEnumerate4}
  while typing \fprimo) gives ‘1°’.
30.257 \def\fprimo{\FrenchPopularEnumerate1}
30.258 \def\fsecundo{\FrenchPopularEnumerate2}
30.259 \def\ftertio{\FrenchPopularEnumerate3}
30.260 \def\fquarto{\FrenchPopularEnumerate4}

```

Let’s provide four macros for the common abbreviations of “Numéro”.

```

30.261 \DeclareRobustCommand*\No{\N\up{\lowercase{o}}\kern+.2em}
30.262 \DeclareRobustCommand*\no{\n\up{\lowercase{o}}\kern+.2em}
30.263 \DeclareRobustCommand*\Nos{\N\up{\lowercase{os}}\kern+.2em}
30.264 \DeclareRobustCommand*\nos{\n\up{\lowercase{os}}\kern+.2em}

```

\bsc As family names should be written in small capitals and never be hyphenated, we provide a command (its name comes from Boxed Small Caps) to input them easily. Note that this command has changed with version 2 of frenchb: a \kern0pt is used instead of \hbox because \hbox would break microtype’s font expansion; as a (positive?) side effect, composed names (such as Dupont-Durand) can now be hyphenated on explicit hyphens. Usage: Jean~\bsc{Duchemin}.

```

30.265 \DeclareRobustCommand*\bsc[1]{\leavevmode\begin{group}\kern0pt
30.266 #1\endgroup\scshape}
30.267 \ifLaTeX\else\let\scshape\relax\fi

```

Some definitions for special characters. We won’t define \tilde as a Text Symbol not to conflict with the macro \tilde for math mode and use the name \tild instead. Note that \boi may *not* be used in math mode, its name in math mode is \backslash. \degre can be accessed by the command \r{} for ring accent.

```

30.268 \ifLaTeX
30.269 \DeclareTextSymbol{\at}{T1}{64}
30.270 \DeclareTextSymbol{\circonflexe}{T1}{94}
30.271 \DeclareTextSymbol{\tild}{T1}{126}
30.272 \DeclareTextSymbolDefault{\at}{T1}
30.273 \DeclareTextSymbolDefault{\circonflexe}{T1}
30.274 \DeclareTextSymbolDefault{\tild}{T1}
30.275 \DeclareRobustCommand*\boi{\textbackslash}
30.276 \DeclareRobustCommand*\degre{\r{}}
30.277 \else
30.278 \def\T@one{T1}
30.279 \ifx\f@encoding\T@one
30.280 \newcommand*\degre{\char6}
30.281 \else
30.282 \newcommand*\degre{\char23}

```

```

30.283 \fi
30.284 \newcommand*{\at}{\char64}
30.285 \newcommand*{\circonflexe}{\char94}
30.286 \newcommand*{\tild}{\char126}
30.287 \newcommand*{\boi}{\backslash$}
30.288 \fi

```

\degrees We now define a macro `\degrees` for typesetting the abbreviation for ‘degrees’ (as in ‘degrees Celsius’). As the bounding box of the character ‘degree’ has *very* different widths in CM/EC and PostScript fonts, we fix the width of the bounding box of `\degrees` to 0.3 em, this lets the symbol ‘degree’ stick to the preceding (e.g., `45\degrees`) or following character (e.g., `20~\degrees C`).

If the T_EX Companion fonts are available (`textcomp.sty`), we pick up `\textdegree` from them instead of using emulating ‘degrees’ from the `\r{}` accent. Otherwise we overwrite the (poor) definition of `\textdegree` given in `latin1.def`, `applemac.def` etc. (called by `inputenc.sty`) by our definition of `\degrees`. We also advice the user (once only) to use TS1-encoding.

```

30.289 \ifLaTeXe
30.290 \newcommand*{\degrees}{\degre}
30.291 \def\Warning@degree@TSone{%
30.292     \PackageWarning{frenchb.ldf}{%
30.293         Degrees would look better in TS1-encoding:
30.294         \MessageBreak add \protect
30.295         \usepackage{textcomp} to the preamble.
30.296         \MessageBreak Degrees used}}
30.297 \AtBeginDocument{\expandafter\ifx\csname M@TS1\endcsname\relax
30.298     \DeclareRobustCommand*{\degrees}{%
30.299         \leavevmode\hbox to 0.3em{\hss\degre\hss}%
30.300         \Warning@degree@TSone
30.301         \global\let\Warning@degree@TSone\relax}%
30.302     \let\textdegree\degrees
30.303 \else
30.304     \DeclareRobustCommand*{\degrees}{%
30.305         \hbox{\UseTextSymbol{TS1}{\textdegree}}}%
30.306 \fi}
30.307 \else
30.308 \newcommand*{\degrees}{%
30.309     \leavevmode\hbox to 0.3em{\hss\degre\hss}}
30.310 \fi

```

30.6 Formatting numbers

\DecimalMathComma As mentioned in the T_EXbook p. 134, the comma is of type `\mathpunct` in math mode: it is automatically followed by a space. This is convenient in lists and intervals but unpleasant when the comma is used as a decimal separator in French: it has to be entered as `{,}`. `\DecimalMathComma` makes the comma be an ordinary character (of type `\mathord`) in French *only* (no space added); `\StandardMathComma` switches back to the standard behaviour of the comma.

```

30.311 \newcount\std@mcc
30.312 \newcount\dec@mcc
30.313 \std@mcc=\mathcode'\,,
30.314 \dec@mcc=\std@mcc
30.315 \@tempcnta=\std@mcc
30.316 \divide\@tempcnta by "1000
30.317 \multiply\@tempcnta by "1000
30.318 \advance\dec@mcc by -\@tempcnta
30.319 \newcommand*{\DecimalMathComma}{\iflanguage{french}%
30.320     {\mathcode'\,,=\dec@mcc}{}}%
30.321     \addto\extrasfrench{\mathcode'\,,=\dec@mcc}}
30.322 \newcommand*{\StandardMathComma}{\mathcode'\,,=\std@mcc
30.323     \addto\extrasfrench{\mathcode'\,,=\std@mcc}}

```



```

30.324 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
30.325   \mathcode'\,=\std@mcc}

```

`\nombre` The command `\nombre` is now borrowed from `numprint.sty` for $\text{\LaTeX} 2_{\epsilon}$. There is no point to maintain the former tricky code when a package is dedicated to do the same job and more. For Plain based formats, `\nombre` no longer formats numbers, it prints them as is and issues a warning about the change.

Fake command `\nombre` for Plain based formats, warning users of `frenchb v.1.x.` of the change.

```

30.326 \newcommand*{\nombre}[1]{\#1}\message{%
30.327   *** \noexpand\nombre no longer formats numbers\string! ***}}%

```

The next definitions only make sense for $\text{\LaTeX} 2_{\epsilon}$. Let's cleanup and exit if the format in Plain based.

```

30.328 \let\FBstop@here\relax
30.329 \def\FBclean@on@exit{\let\ifLaTeXe\@undefined
30.330                      \let\LaTeXettrue\@undefined
30.331                      \let\LaTeXefalse\@undefined}
30.332 \ifx\magnification\@undefined
30.333 \else
30.334   \def\FBstop@here{\let\STD@makecaption\relax
30.335                  \FBclean@on@exit
30.336                  \ldf@quit\CurrentOption\endinput}
30.337 \fi
30.338 \FBstop@here

```

What follows now is for $\text{\LaTeX} 2_{\epsilon}$ *only*. We redefine `\nombre` for $\text{\LaTeX} 2_{\epsilon}$. A warning is issued at the first call of `\nombre` if `\numprint` is not defined, suggesting what to do. The package `numprint` is *not* loaded automatically by `frenchb` because of possible options conflict.

```

30.339 \renewcommand*{\nombre}[1]{\Warning@nombre\numprint{#1}}
30.340 \newcommand*{\Warning@nombre}{%
30.341   \@ifundefined{numprint}%
30.342     {\PackageWarning{frenchb.ldf}{%
30.343       \protect\nombre\space now relies on package numprint.sty,
30.344       \MessageBreak add \protect
30.345       \usepackage[autolanguage]{numprint}\MessageBreak
30.346       to your preamble *after* loading babel, \MessageBreak
30.347       see file numprint.pdf for other options.\MessageBreak
30.348       \protect\nombre\space called}%
30.349     \global\let\Warning@nombre\relax
30.350     \global\let\numprint\relax
30.351   }{}}%
30.352 }

30.353 \newcommand*{\ThinSpaceInFrenchNumbers}{%
30.354   \PackageWarning{frenchb.ldf}{%
30.355     Type \protect\frenchbsetup{ThinSpaceInFrenchNumbers}
30.356     \MessageBreak Command \protect\ThinSpaceInFrenchNumbers\space
30.357     is no longer\MessageBreak defined in frenchb v.2,}}

```

30.7 Caption names

The next step consists of defining the French equivalents for the \LaTeX caption names.

`\captionsfrench` Let's first define `\captionsfrench` which sets all strings used in the four standard document classes provided with \LaTeX .

```

30.358 \@namedef{captions\CurrentOption}{%
30.359   \def\refname{R\ref\erences}%
30.360   \def\abstractname{R\esum\es}}

```

```

30.361 \def\bibName{Bibliographie}%
30.362 \def\prefacename{Pr\'eface}%
30.363 \def\chaptername{Chapitre}%
30.364 \def\appendixname{Annexe}%
30.365 \def\contentsname{Table des mati\'eres}%
30.366 \def\listfigurename{Table des figures}%
30.367 \def\listtablename{Liste des tableaux}%
30.368 \def\indexname{Index}%
30.369 \def\figurename{{\scshape Figure}}%
30.370 \def\tablename{{\scshape Table}}%

“Première partie” instead of “Part I”.

30.371 \def\partname{\protect\@Fpt partie}%
30.372 \def\@Fpt{{\ifcase\value{part}\or Premi\'ere\or Deuxi\'eme\or
30.373 Troisi\'eme\or Quatri\'eme\or Cinqui\'eme\or Sixi\'eme\or
30.374 Septi\'eme\or Huiti\'eme\or Neuvi\'eme\or Dixi\'eme\or Onzi\'eme\or
30.375 Douzi\'eme\or Treizi\'eme\or Quatorzi\'eme\or Quinzi\'eme\or
30.376 Seizi\'eme\or Dix-septi\'eme\or Dix-huiti\'eme\or Dix-neuvi\'eme\or
30.377 Vingt\'eme\fi}\space\def\thepart{}}%
30.378 \def\pagename{page}%
30.379 \def\seenname{{\emph{voir}}}%
30.380 \def\alsoname{{\emph{voir aussi}}}%
30.381 \def\enclname{P.~J. }%
30.382 \def\ccname{Copie \\'a }%
30.383 \def\headtoname{}%
30.384 \def\proofname{D\'emonstration}%
30.385 \def\glossaryname{Glossaire}%
30.386 }

```

As some users who choose frenchb or francais as option of babel, might customise \captionfrenchb or \captionfrancais in the preamble, we merge their changes at the \begin{document} when they do so. The other variants of French (canadien, acadian) are defined by checking if the relevant option was used and then adding one extra level of expansion.

```

30.387 \AtBeginDocument{\let\captions@French\captionfrench
30.388 \ifundefined{captionfrenchb}%
30.389 {\let\captions@Frenchb\relax}%
30.390 {\let\captions@Frenchb\captionfrenchb}%
30.391 \ifundefined{captionfrancais}%
30.392 {\let\captions@Francais\relax}%
30.393 {\let\captions@Francais\captionfrancais}%
30.394 \def\captionfrench{\caption@French
30.395 \caption@Francais\caption@Frenchb}%
30.396 \def\captionfrancais{\captionfrench}%
30.397 \def\captionfrenchb{\captionfrench}%
30.398 \iflanguage{french}{\captionfrench}{}%
30.399 }
30.400 \ifpackagewith{babel}{canadien}{%
30.401 \def\captionscanadien{\captionfrench}%
30.402 \def\datecanadien{\datefrench}%
30.403 \def\extrascanadien{\extrasfrench}%
30.404 \def\noextrascanadien{\noextrasfrench}%
30.405 }{}
30.406 \ifpackagewith{babel}{acadian}{%
30.407 \def\captionacadian{\captionfrench}%
30.408 \def\dateacadian{\datefrench}%
30.409 \def\extrasacadian{\extrasfrench}%
30.410 \def\noextrasacadian{\noextrasfrench}%
30.411 }{}

```

\CaptionSeparator Let’s consider now captions in figures and tables. In French, captions in figures and tables should be printed with endash (‘–’) instead of the standard ‘:’.

The standard definition of `\@makecaption` (e.g., the one provided in `article.cls`, `report.cls`, `book.cls` which is frozen for L^AT_EX 2_ε according to Frank Mittelbach), has been saved in `\STD@makecaption` before making ‘:’ active (see section 30.2). ‘AtBeginDocument’ we compare it to its current definition (some classes like `koma-script` classes, `AMS` classes, `ua-thesis.cls`...change it). If they are identical, `frenchb` just adds a hook called `\CaptionSeparator` to `\@makecaption`, `\CaptionSeparator` defaults to ‘: ’ as in the standard `\@makecaption`, and will be changed to ‘ – ’ in French. If the definitions differ, `frenchb` doesn’t overwrite the changes, but prints a message in the .log file.

```

30.412 \def\CaptionSeparator{\string:\space}
30.413 \long\def\FB@makecaption#1#2{%
30.414   \vskip\abovecaptionskip
30.415   \sbox\@tempboxa{#1\CaptionSeparator #2}%
30.416   \ifdim \wd\@tempboxa >\hsize
30.417     #1\CaptionSeparator #2\par
30.418   \else
30.419     \global \@minipagefalse
30.420     \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
30.421   \fi
30.422   \vskip\belowcaptionskip}
30.423 \AtBeginDocument{%
30.424   \ifx\@makecaption\STD@makecaption
30.425     \global\let\@makecaption\FB@makecaption
30.426   \else
30.427     \@ifundefined{@makecaption}{}%
30.428     {\PackageWarning{frenchb.ldb}%
30.429       {The definition of \protect\@makecaption\space
30.430        has been changed,\MessageBreak
30.431        frenchb will NOT customise it;\MessageBreak reported}%
30.432     }%
30.433   \fi
30.434   \let\FB@makecaption\relax
30.435   \let\STD@makecaption\relax
30.436 }
30.437 \expandafter\addto\csname extras\CurrentOption\endcsname{%
30.438   \def\CaptionSeparator{\space\textendash\space}}
30.439 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
30.440   \def\CaptionSeparator{\string:\space}}

```

30.8 French lists

`\listFB` Vertical spacing in general lists should be shorter in French texts than the defaults provided by L^AT_EX. Note that the easy way, just changing values of vertical spacing parameters when entering French and restoring them to their defaults on exit would not work; as most lists are based on `\list` we will define a variant of `\list` (`\listFB`) to be used in French.

The amount of vertical space before and after a list is given by `\topsep` + `\parskip` (+ `\partopsep` if the list starts a new paragraph). IMHO, `\parskip` should be added *only* when the list starts a new paragraph, so I subtract `\parskip` from `\topsep` and add it back to `\partopsep`; this will normally make no difference because `\parskip`’s default value is 0pt, but will be noticeable when `\parskip` is *not* null.

`\endlist` is not redefined, but `\endlistORI` is provided for the users who prefer to define their own lists from the original command, they can code: `\begin{listORI}{}{} \end{listORI}`.

```

30.441 \let\listORI\list
30.442 \let\endlistORI\endlist
30.443 \def\FB@listsettings{%
30.444   \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
30.445   \setlength{\parsep}{0.4ex plus 0.2ex minus 0.2ex}%

```

```

30.446 \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
30.447 \setlength{\partopsep}{0.4ex plus 0.2ex minus 0.2ex}%
\parskip is of type ‘skip’, its mean value only (not the glue) should be subtracted
from \topsep and added to \partopsep, so convert \parskip to a ‘dimen’ using
\@tempdima.
30.448 \@tempdima=\parskip
30.449 \addtolength{\topsep}{-\@tempdima}%
30.450 \addtolength{\partopsep}{\@tempdima}%
30.451 \def\listFB#1#2{\listORI{#1}{\FB@listsettings #2}}%
30.452 \let\endlistFB\endlist

```

\itemizeFB Let’s now consider French itemize lists. They differ from those provided by the
\itemizeORI standard L^AT_EX 2_ε classes:

- \bbl@frenchlabelitems • vertical spacing between items, before and after the list, should be *null* with
\bbl@nonfrenchlabelitems *no glue* added;
- the item labels of a first level list should be vertically aligned on the para-
graph’s first character (i.e. at \parindent from the left margin);
- the ‘•’ is never used in French itemize-lists, a long dash ‘—’ is preferred for all
levels. The item label used in French is stored in \FrenchLabelItem, it de-
faults to ‘—’ and can be changed using \frenchbsetup{} (see section 30.13).

```

30.453 \newcommand*{\FrenchLabelItem}{\textendash}
30.454 \newcommand*{\Frlabelitemi}{\FrenchLabelItem}
30.455 \newcommand*{\Frlabelitemii}{\FrenchLabelItem}
30.456 \newcommand*{\Frlabelitemiii}{\FrenchLabelItem}
30.457 \newcommand*{\Frlabelitemiv}{\FrenchLabelItem}

```

\bbl@frenchlabelitems saves current itemize labels and changes them to their
value in French. This code should never be executed twice in a row, so we
need a new flag that will be set and reset by \bbl@nonfrenchlabelitems and
\bbl@frenchlabelitems.

```

30.458 \newif\iffB@enterFrench \FB@enterFrenchtrue
30.459 \def\bbl@frenchlabelitems{%
30.460 \iffB@enterFrench
30.461 \let\@ltiORI\labelitemi
30.462 \let\@ltiiORI\labelitemii
30.463 \let\@ltiiiORI\labelitemiii
30.464 \let\@ltivORI\labelitemiv
30.465 \let\labelitemi\Frlabelitemi
30.466 \let\labelitemii\Frlabelitemii
30.467 \let\labelitemiii\Frlabelitemiii
30.468 \let\labelitemiv\Frlabelitemiv
30.469 \FB@enterFrenchfalse
30.470 \fi
30.471 }
30.472 \let\itemizeORI\itemize
30.473 \let\enditemizeORI\enditemize
30.474 \let\enditemizeFB\enditemize
30.475 \def\itemizeFB{%
30.476 \ifnum \@itemdepth >\thr@@\toodeep\else
30.477 \advance\@itemdepth\@ne
30.478 \edef\@itemitem{\labelitem\romannumeral\the\@itemdepth}%
30.479 \expandafter
30.480 \listORI
30.481 \csname\@itemitem\endcsname
30.482 {\settowidth{\labelwidth}{\csname\@itemitem\endcsname}%
30.483 \setlength{\leftmargin}{\labelwidth}%
30.484 \addtolength{\leftmargin}{\labelsep}%
30.485 \ifnum\@listdepth=0

```

```

30.486      \setlength{\itemindent}{\parindent}%
30.487      \else
30.488      \addtolength{\leftmargin}{\parindent}%
30.489      \fi
30.490      \setlength{\itemsep}{\z@}%
30.491      \setlength{\parsep}{\z@}%
30.492      \setlength{\topsep}{\z@}%
30.493      \setlength{\partopsep}{\z@}%

\parskip is of type ‘skip’, its mean value only (not the glue) should be subtracted
from \topsep and added to \partopsep, so convert \parskip to a ‘dimen’ using
\@tempdima.

30.494      \@tempdima=\parskip
30.495      \addtolength{\topsep}{-\@tempdima}%
30.496      \addtolength{\partopsep}{\@tempdima}%
30.497      \fi}

The user’s changes in labelitems are saved when leaving French for further use
when switching back to French. This code should never be executed twice in a
row (toggle with \bbl@frenchlabelitems).

30.498 \def\bbl@nonfrenchlabelitems{%
30.499   \ifFB@enterFrench
30.500   \else
30.501     \let\Frlabelitemi\labelitemi
30.502     \let\Frlabelitemii\labelitemii
30.503     \let\Frlabelitemiii\labelitemiii
30.504     \let\Frlabelitemiv\labelitemiv
30.505     \let\labelitemi\@ltiORI
30.506     \let\labelitemii\@ltiiORI
30.507     \let\labelitemiii\@ltiiiORI
30.508     \let\labelitemiv\@ltivORI
30.509     \FB@enterFrenchtrue
30.510   \fi
30.511 }

```

30.9 French indentation of sections

`\bbl@frenchindent` In French the first paragraph of each section should be indented, this is another difference with US-English. This is controlled by the flag `\if@afterindent`.
`\bbl@nonfrenchindent` We need to save the value of the flag `\if@afterindent` ‘AtBeginDocument’ before eventually changing its value.

```

30.512 \AtBeginDocument{\ifx\@afterindentfalse\@afterindenttrue
30.513                   \let\@aifORI\@afterindenttrue
30.514                   \else \let\@aifORI\@afterindentfalse
30.515                   \fi
30.516 }
30.517 \def\bbl@frenchindent{\let\@afterindentfalse\@afterindenttrue
30.518                   \@afterindenttrue}
30.519 \def\bbl@nonfrenchindent{\let\@afterindentfalse\@aifORI
30.520                   \@afterindentfalse}

```

30.10 Formatting footnotes

The **bigfoot** package deeply changes the way footnotes are handled. When **bigfoot** is loaded, we just warn the user that **frenchb** will drop the customisation of footnotes.

The layout of footnotes is controlled by two flags `\ifFBAutoSpaceFootnotes` and `\ifFBFrenchFootnotes` which are set by options of `\frenchbsetup{}` (see section 30.13). Notice that the layout of footnotes *does not depend* on the current language (just think of two footnotes on the same page looking different because one was called in a French part, the other one in English!).

When `\ifFBAutoSpaceFootnotes` is true, `\@footnotemark` (whose definition is saved at the `\begin{document}` in order to include any customisation that packages might have done) is redefined to add a thin space before the number or symbol calling a footnote (any space typed in is removed first). This has no effect on the layout of the footnote itself.

```

30.521 \AtBeginDocument{\@ifpackageloaded{bigfoot}%
30.522     {\PackageWarning{frenchb.ldf}%
30.523       {bigfoot package in use.\MessageBreak
30.524         frenchb will NOT customise footnotes;\MessageBreak
30.525         reported}}}%
30.526     {\let\@footnotemarkORI\@footnotemark
30.527      \def\@footnotemarkFB{\leavevmode\unskip\unkern
30.528        \,\@footnotemarkORI}%
30.529      \ifFBAutoSpaceFootnotes
30.530        \let\@footnotemark\@footnotemarkFB
30.531      \fi}%
30.532   }

```

We then define `\@makefntextFB`, a variant of `\@makefntext` which is responsible for the layout of footnotes, to match the specifications of the French ‘Imprimerie Nationale’: footnotes will be indented by `\parindentFFN`, numbers (if any) typeset on the baseline (instead of superscripts) and followed by a dot and an half quad space. Whenever symbols are used to number footnotes (as in `\thanks` for instance), we switch back to the standard layout (the French layout of footnotes is meant for footnotes numbered by Arabic or Roman digits).

The value of `\parindentFFN` will be redefined at the `\begin{document}`, as the maximum of `\parindent` and 1.5em *unless* it has been set in the preamble (the weird value 10in is just for testing whether `\parindentFFN` has been set or not).

```

30.533 \newcommand*{\dotFFN}{.}
30.534 \newcommand*{\kernFFN}{\kern .5em}
30.535 \newdimen\parindentFFN
30.536 \parindentFFN=10in
30.537 \def\ftnISsymbol{\@fnsymbol\c@footnote}
30.538 \long\def\@makefntextFB#1{\ifx\thefootnote\ftnISsymbol
30.539     \@makefntextORI{#1}%
30.540   \else
30.541     \parindent=\parindentFFN
30.542     \rule{z@}{\footnotesep}
30.543     \setbox\@tempboxa\hbox{\@thefnmark}%
30.544     \ifdim\wd\@tempboxa>z@
30.545       \llap{\@thefnmark}\dotFFN\kernFFN
30.546     \fi #1
30.547   \fi}%

```

We save the standard definition of `\@makefntext` at the `\begin{document}`, and then redefine `\@makefntext` according to the value of flag `\ifFBFrenchFootnotes` (true or false).

```

30.548 \AtBeginDocument{\@ifpackageloaded{bigfoot}{}%
30.549     {\ifdim\parindentFFN<10in
30.550       \else
30.551         \parindentFFN=\parindent
30.552         \ifdim\parindentFFN<1.5em\parindentFFN=1.5em\fi
30.553       \fi
30.554       \let\@makefntextORI\@makefntext
30.555       \long\def\@makefntext#1{%
30.556         \ifFBFrenchFootnotes
30.557           \@makefntextFB{#1}%
30.558         \else
30.559           \@makefntextORI{#1}%
30.560         \fi}%

```

```

30.561         }%
30.562     }

```

For compatibility reasons, we provide definitions for the commands dealing with the layout of footnotes in **frenchb** version 1.6. `\frenchbsetup{}` (see in section 30.13) should be preferred for setting these options. `\StandardFootnotes` may still be used locally (in minipages for instance), that’s why the test `\ifFBFrenchFootnotes` is done inside `\@makefnctext`.

```

30.563 \newcommand*{\AddThinSpaceBeforeFootnotes}{\FBAutoSpaceFootnotestrue}
30.564 \newcommand*{\FrenchFootnotes}{\FBFrenchFootnotestrue}
30.565 \newcommand*{\StandardFootnotes}{\FBFrenchFootnotesfalse}

```

30.11 Global layout

In multilingual documents, some typographic rules must depend on the current language (e.g., hyphenation, typesetting of numbers, spacing before double punctuation. . .), others should, IMHO, be kept global to the document: especially the layout of lists (see 30.8) and footnotes (see 30.10), and the indentation of the first paragraph of sections (see 30.9).

From version 2.2 on, if **frenchb** is babel’s “main language” (i.e. last language option at babel’s loading), **frenchb** customises the layout (i.e. lists, indentation of the first paragraphs of sections and footnotes) in the whole document regardless the current language. On the other hand, if **frenchb** is *not* babel’s “main language”, it leaves the layout unchanged both in French and in other languages.

`\FrenchLayout` The former commands `\FrenchLayout` and `\StandardLayout` are kept for compatibility reasons but should no longer be used.

```

30.566 \newcommand*{\FrenchLayout}{%
30.567     \FBGlobalLayoutFrenchtrue
30.568     \PackageWarning{frenchb.1df}%
30.569     {\protect\FrenchLayout\space is obsolete. Please use\MessageBreak
30.570     \protect\frenchbsetup{GlobalLayoutFrench} instead.}%
30.571 }
30.572 \newcommand*{\StandardLayout}{%
30.573     \FBReduceListSpacingfalse
30.574     \FBCompactItemizefalse
30.575     \FBStandardItemLabelstrue
30.576     \FBIndentFirstfalse
30.577     \FBFrenchFootnotesfalse
30.578     \FBAutoSpaceFootnotesfalse
30.579     \PackageWarning{frenchb.1df}%
30.580     {\protect\StandardLayout\space is obsolete. Please use\MessageBreak
30.581     \protect\frenchbsetup{StandardLayout} instead.}%
30.582 }
30.583 \@onlypreamble\FrenchLayout
30.584 \@onlypreamble\StandardLayout

```

30.12 Dots...

`\FBtextellipsis` L^AT_EX 2_ε’s standard definition of `\dots` in text-mode is `\textellipsis` which includes a `\kern` at the end; this space is not wanted in some cases (before a closing brace for instance) and `\kern` breaks hyphenation of the next word. We define `\FBtextellipsis` for French (in L^AT_EX 2_ε only).

The `\if` construction in the L^AT_EX 2_ε definition of `\dots` doesn’t allow the use of `xspace` (`xspace` is always followed by a `\fi`), so we use the AMS-L^AT_EX construction of `\dots`; this has to be done ‘AtBeginDocument’ not to be overwritten when `amsmath.sty` is loaded after `babel`.

LY1 has a ready made character for `\textellipsis`, it should be used in French too (pointed out by Bruno Voisin).

```

30.585 \DeclareTextSymbol{\FBtextellipsis}{LY1}{133}

```

```

30.586 \DeclareTextCommandDefault{\FBtextellipsis}{%
30.587   .\kern\fontdimen3\font.\kern\fontdimen3\font.\xspace}

\Mdots@ and \Tdots@ORI hold the definitions of \dots in Math and Text mode.
They default to those of amsmath-2.0, and will revert to standard LATEX definitions
‘AtBeginDocument’, if amsmath has not been loaded. \Mdots@ doesn’t change
when switching from/to French, while \Tdots@ is \FBtextellipsis in French
and \Tdots@ORI otherwise.

30.588 \newcommand*{\Tdots@ORI}{\@xp\textellipsis}
30.589 \newcommand*{\Tdots@}{\Tdots@ORI}
30.590 \newcommand*{\Mdots@}{\@xp\mdots@}
30.591 \AtBeginDocument{\DeclareRobustCommand*{\dots}{\relax
30.592   \csname\ifmmode M\else T\fi dots@\endcsname}%
30.593   \@ifundefined{@xp}{\let\@xp\relax}{}%
30.594   \@ifundefined{mdots@}{\let\Tdots@ORI\textellipsis
30.595     \let\Mdots@\mathellipsis}{}}
30.596 \def\bbl@frenchdots{\let\Tdots@\FBtextellipsis}
30.597 \def\bbl@nonfrenchdots{\let\Tdots@\Tdots@ORI}
30.598 \expandafter\addto\csname extras\CurrentOption\endcsname{%
30.599   \bbl@frenchdots}
30.600 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
30.601   \bbl@nonfrenchdots}

```

30.13 Setup options: keyval stuff

We first define a collection of conditionals with their defaults (true or false).

```

30.602 \newif\ifFBStandardLayout           \FBStandardLayouttrue
30.603 \newif\ifFBGlobalLayoutFrench      \FBGlobalLayoutFrenchfalse
30.604 \newif\ifFBReduceListSpacing       \FBReduceListSpacingfalse
30.605 \newif\ifFBCompactItemize          \FBCompactItemizefalse
30.606 \newif\ifFBStandardItemLabels      \FBStandardItemLabelstrue
30.607 \newif\ifFBStandardLists            \FBStandardListstrue
30.608 \newif\ifFBIndentFirst              \FBIndentFirstfalse
30.609 \newif\ifFBFrenchFootnotes          \FBFrenchFootnotesfalse
30.610 \newif\ifFBAutoSpaceFootnotes      \FBAutoSpaceFootnotesfalse
30.611 \newif\ifFBOriginalTypewriter      \FBOriginalTypewriterfalse
30.612 \newif\ifFBThinColonSpace          \FBThinColonSpacefalse
30.613 \newif\ifFBThinSpaceInFrenchNumbers \FBThinSpaceInFrenchNumbersfalse
30.614 \newif\ifFBFrenchSuperscripts      \FBFrenchSuperscriptstrue
30.615 \newif\ifFBLowercaseSuperscripts   \FBLowercaseSuperscriptstrue
30.616 \newif\ifFBPartNameFull            \FBPartNameFulltrue
30.617 \newif\ifFBShowOptions             \FBShowOptionsfalse

```

The defaults values of these flags have been set so that `frenchb` does not change anything regarding the global layout. `\bbl@main@language` (set by the last option of `babel`) controls the global layout of the document. We check the current language ‘AtEndOfPackage’ (it is `\bbl@main@language`); if it is French, the values of some flags have to be changed to ensure a French looking layout for the whole document (even in parts written in languages other than French); the end-user will then be able to customise the values of all these flags with `\frenchbsetup{}`.

```

30.618 \AtEndOfPackage{%
30.619   \iflanguage{french}{\FBReduceListSpacingtrue
30.620     \FBCompactItemizetrue
30.621     \FBStandardItemLabelsfalse
30.622     \FBIndentFirsttrue
30.623     \FBFrenchFootnotestrue
30.624     \FBAutoSpaceFootnotestrue
30.625     \FBGlobalLayoutFrenchtrue}%
30.626   }%
30.627 }

```


`\frenchbsetup` From version 2.0 on, all setup options are handled by *one* command `\frenchbsetup` using the keyval syntax. Let's now define this command which reads and sets the options to be processed later (at `\begin{document}`) by `\FBprocess@options`. It can only be called in the preamble.

```
30.628 \newcommand*{\frenchbsetup}[1]{%
30.629   \setkeys{FB}{#1}%
30.630 }%
30.631 \@onlypreamble\frenchbsetup
```

`frenchb` being an option of `babel`, it cannot load a package (keyval) while `frenchb.ldf` is read, so we defer the loading of `keyval` and the options setup at the end of `babel`'s loading.

`StandardLayout` resets the layout in French to the standard layout defined par the `LATEX` class and packages loaded. It deals with lists, indentation of first paragraphs of sections and footnotes. Other keys, entered *after* `StandardLayout` in `\frenchbsetup`, can overrule some of the `StandardLayout` settings.

`GlobalLayoutFrench` forces the layout in French and (as far as possible) outside French to meet the French typographic standards.

```
30.632 \AtEndOfPackage{%
30.633   \RequirePackage{keyval}%
30.634   \define@key{FB}{StandardLayout}[true]%
30.635                                   {\csname FBStandardLayout#1\endcsname
30.636                                   \ifFBStandardLayout
30.637                                   \FBReduceListSpacingfalse
30.638                                   \FBCompactItemizefalse
30.639                                   \FBStandardItemLabelstrue
30.640                                   \FBIndentFirstfalse
30.641                                   \FBFrenchFootnotesfalse
30.642                                   \FBAutoSpaceFootnotesfalse
30.643                                   \FBGlobalLayoutFrenchfalse
30.644                                   \else
30.645                                   \FBReduceListSpacingtrue
30.646                                   \FBCompactItemizetrue
30.647                                   \FBStandardItemLabelsfalse
30.648                                   \FBIndentFirsttrue
30.649                                   \FBFrenchFootnotetrue
30.650                                   \FBAutoSpaceFootnotetrue
30.651                                   \fi}%
30.652   \define@key{FB}{GlobalLayoutFrench}[true]%
30.653                                   {\csname FBGlobalLayoutFrench#1\endcsname
30.654                                   \ifFBGlobalLayoutFrench
30.655                                   \iflanguage{french}%
30.656                                   {\FBReduceListSpacingtrue
30.657                                   \FBCompactItemizetrue
30.658                                   \FBStandardItemLabelsfalse
30.659                                   \FBIndentFirsttrue
30.660                                   \FBFrenchFootnotetrue
30.661                                   \FBAutoSpaceFootnotetrue}%
30.662                                   {\PackageWarning{frenchb.ldf}%
30.663                                   {Option 'GlobalLayoutFrench' skipped:
30.664                                   \MessageBreak French is *not*
30.665                                   babel's last option.\MessageBreak}}%
30.666                                   \fi}%
30.667   \define@key{FB}{ReduceListSpacing}[true]%
30.668                                   {\csname FBReduceListSpacing#1\endcsname}%
30.669   \define@key{FB}{CompactItemize}[true]%
30.670                                   {\csname FBCompactItemize#1\endcsname}%
30.671   \define@key{FB}{StandardItemLabels}[true]%
30.672                                   {\csname FBStandardItemLabels#1\endcsname}%
30.673   \define@key{FB}{ItemLabels}{%
30.674     \renewcommand*{\FrenchLabelItem}{#1}}%
30.675   \define@key{FB}{ItemLabeli}{%
```

```

30.676 \renewcommand*{\Frlabelitemi}{#1}}%
30.677 \define@key{FB}{ItemLabelii}{%
30.678 \renewcommand*{\Frlabelitemii}{#1}}%
30.679 \define@key{FB}{ItemLabeliii}{%
30.680 \renewcommand*{\Frlabelitemiii}{#1}}%
30.681 \define@key{FB}{ItemLabeliv}{%
30.682 \renewcommand*{\Frlabelitemiv}{#1}}%
30.683 \define@key{FB}{StandardLists}[true]%
30.684 {\csname FBStandardLists#1\endcsname
30.685 \ifFBStandardLists
30.686 \FBReduceListSpacingfalse
30.687 \FBCompactItemizefalse
30.688 \FBStandardItemLabelstrue
30.689 \else
30.690 \FBReduceListSpacingtrue
30.691 \FBCompactItemizetrue
30.692 \FBStandardItemLabelsfalse
30.693 \fi}%
30.694 \define@key{FB}{IndentFirst}[true]%
30.695 {\csname FBIndentFirst#1\endcsname}%
30.696 \define@key{FB}{FrenchFootnotes}[true]%
30.697 {\csname FBFrenchFootnotes#1\endcsname}%
30.698 \define@key{FB}{AutoSpaceFootnotes}[true]%
30.699 {\csname FBAutoSpaceFootnotes#1\endcsname}%
30.700 \define@key{FB}{AutoSpacePunctuation}[true]%
30.701 {\csname FBAutoSpacePunctuation#1\endcsname}%
30.702 \define@key{FB}{OriginalTypewriter}[true]%
30.703 {\csname FBOriginalTypewriter#1\endcsname}%
30.704 \define@key{FB}{ThinColonSpace}[true]%
30.705 {\csname FBThinColonSpace#1\endcsname}%
30.706 \define@key{FB}{ThinSpaceInFrenchNumbers}[true]%
30.707 {\csname FBThinSpaceInFrenchNumbers#1\endcsname}%
30.708 \define@key{FB}{FrenchSuperscripts}[true]%
30.709 {\csname FBFrenchSuperscripts#1\endcsname}%
30.710 \define@key{FB}{LowercaseSuperscripts}[true]%
30.711 {\csname FBLowercaseSuperscripts#1\endcsname}%
30.712 \define@key{FB}{PartNameFull}[true]%
30.713 {\csname FBPartNameFull#1\endcsname}%
30.714 \define@key{FB}{ShowOptions}[true]%
30.715 {\csname FBShowOptions#1\endcsname}%

```

Inputting French quotes as *single characters* when they are available on the keyboard (through a compose key for instance) is more comfortable than typing `\og` and `\fg`. The purpose of the following code is to map the French quote characters to `\og\ignorespaces` and `{\fg}` respectively when the current language is French, and to `\guillemotleft` and `\guillemotright` otherwise (think of German quotes); thus correct unbreakable spaces will be added automatically to French quotes. The quote characters typed in depend on the input encoding, it can be single-byte (latin1, latin9, applemac, ...) or multi-bytes (utf-8, utf8x). We first check whether XeTeX is used or not, if not the package `inputenc` has to be loaded before the `\begin{document}` with the proper coding option, so we check if `\DeclareInputText` is defined.

```

30.716 \define@key{FB}{og}{%
30.717 \newcommand*{\FB@og}{\iflanguage{french}%
30.718 {\FB@og\ignorespaces}{\guillemotleft}}%
30.719 \expandafter\ifx\csname XeTeXrevision\endcsname\relax
30.720 \AtBeginDocument{%
30.721 \@ifundefined{DeclareInputText}%
30.722 {\PackageWarning{frenchb.ldb}%
30.723 {Option ‘og’ requires package inputenc.\MessageBreak}%
30.724 }%
30.725 {\@ifundefined{uc@dc1c}%

```

```

if \uc@dc1c is undefined, utf8x is not loaded...
30.726      {\@ifundefined{DeclareUnicodeCharacter}%
if \DeclareUnicodeCharacter is undefined, utf8 is not loaded either, we as-
sume 8-bit character input encoding. Package MULEenc.sty (from CJK) defines
\mule@def to map characters to control sequences.
30.727      {\@tempcnta'#1\relax
30.728      \@ifundefined{mule@def}%
30.729      {\DeclareInputText{\the\@tempcnta}{\FB@@og}}%
30.730      {\mule@def{11}{\FB@@og}}}%
30.731      }%

utf8 loaded, use \DeclareUnicodeCharacter,
30.732      {\DeclareUnicodeCharacter{00AB}{\FB@@og}}%
30.733      }%

utf8x loaded, use \uc@dc1c,
30.734      {\uc@dc1c{171}{default}{\FB@@og}}%
30.735      }%
30.736      }%

XeTeX in use, the following trick for defining the active quote character is borrowed
from inputenc.dtx.

30.737      \else
30.738      \catcode'#1=\active
30.739      \bgroup
30.740      \uccode'\~'#1%
30.741      \uppercase{%
30.742      \egroup
30.743      \def~%
30.744      }{\FB@@og}%
30.745      \fi
30.746      }%

Same code for the closing quote.
30.747      \define@key{FB}{fg}{%
30.748      \newcommand*{\FB@@fg}{\iflanguage{french}%
30.749      {\FB@fg}{\guillemotright}}}%
30.750      \expandafter\ifx\csname XeTeXrevision\endcsname\relax
30.751      \AtBeginDocument{%
30.752      \@ifundefined{DeclareInputText}%
30.753      {\PackageWarning{frenchb.1df}%
30.754      {Option 'fg' requires package inputenc.\MessageBreak}%
30.755      }%
30.756      {\@ifundefined{uc@dc1c}%
30.757      {\@ifundefined{DeclareUnicodeCharacter}%
30.758      {\@tempcnta'#1\relax
30.759      \@ifundefined{mule@def}%
30.760      {\DeclareInputText{\the\@tempcnta}{\FB@@fg}}%
30.761      {\mule@def{27}{\FB@@fg}}}%
30.762      }%
30.763      {\DeclareUnicodeCharacter{00BB}{\FB@@fg}}}%
30.764      }%
30.765      {\uc@dc1c{187}{default}{\FB@@fg}}}%
30.766      }%
30.767      }%
30.768      \else
30.769      \catcode'#1=\active
30.770      \bgroup
30.771      \uccode'\~'#1%
30.772      \uppercase{%
30.773      \egroup
30.774      \def~%
30.775      }{\FB@@fg}}%

```

```

30.776      \fi
30.777      }%
30.778 }

```

`\FBprocess@options` `\FBprocess@options` processes the options, it is called *once* at `\begin{document}`.

```

30.779 \newcommand*{\FBprocess@options}{%

```

Nothing has to be done here for `StandardLayout` and `StandardLists` (the involved flags have already been set in `\frenchbsetup{}` or before (at babel's `EndOfPackage`).

The next three options deal with the layout of lists in French.

`ReduceListSpacing` reduces the vertical spaces between list items in French (done by changing `\list` to `\listFB`). When `GlobalLayoutFrench` is true (the default), the same is done outside French except for languages that force a different setting.

```

30.780 \ifFBReduceListSpacing
30.781   \addto\extrasfrench{\let\list\listFB
30.782                       \let\endlist\endlistFB}%
30.783   \addto\noextrasfrench{\ifFBGlobalLayoutFrench
30.784                       \let\list\listFB
30.785                       \let\endlist\endlistFB
30.786                       \else
30.787                       \let\list\listORI
30.788                       \let\endlist\endlistORI
30.789                       \fi}%
30.790 \else
30.791   \addto\extrasfrench{\let\list\listORI
30.792                       \let\endlist\endlistORI}%
30.793   \addto\noextrasfrench{\let\list\listORI
30.794                       \let\endlist\endlistORI}%
30.795 \fi

```

`CompactItemize` suppresses the vertical spacing between list items in French (done by changing `\itemize` to `\itemizeFB`). When `GlobalLayoutFrench` is true the same is done outside French.

```

30.796 \ifFBCompactItemize
30.797   \addto\extrasfrench{\let\itemize\itemizeFB
30.798                       \let\enditemize\enditemizeFB}%
30.799   \addto\noextrasfrench{\ifFBGlobalLayoutFrench
30.800                       \let\itemize\itemizeFB
30.801                       \let\enditemize\enditemizeFB
30.802                       \else
30.803                       \let\itemize\itemizeORI
30.804                       \let\enditemize\enditemizeORI
30.805                       \fi}%
30.806 \else
30.807   \addto\extrasfrench{\let\itemize\itemizeORI
30.808                       \let\enditemize\enditemizeORI}%
30.809   \addto\noextrasfrench{\let\itemize\itemizeORI
30.810                       \let\enditemize\enditemizeORI}%
30.811 \fi

```

`StandardItemLabels` resets labelitems in French to their standard values set by the \LaTeX class and packages loaded. When `GlobalLayoutFrench` is true labelitems are identical inside and outside French.

```

30.812 \ifFBStandardItemLabels
30.813   \addto\extrasfrench{\bbl@nonfrenchlabelitems}%
30.814   \addto\noextrasfrench{\bbl@nonfrenchlabelitems}%
30.815 \else
30.816   \addto\extrasfrench{\bbl@frenchlabelitems}%
30.817   \addto\noextrasfrench{\ifFBGlobalLayoutFrench
30.818                       \bbl@frenchlabelitems
30.819                       \else

```

```

30.820             \bbl@nonfrenchlabelitems
30.821             \fi}%
30.822 \fi

```

IndentFirst forces the first paragraphs of sections to be indented just like the other ones in French. When **GlobalLayoutFrench** is true (the default), the same is done outside French except for languages that force a different setting.

```

30.823 \ifBIndentFirst
30.824   \addto\extrasfrench{\bbl@frenchindent}%
30.825   \addto\noextrasfrench{\ifBGlobalLayoutFrench
30.826                       \bbl@frenchindent
30.827                       \else
30.828                       \bbl@nonfrenchindent
30.829                       \fi}%
30.830 \else
30.831   \addto\extrasfrench{\bbl@nonfrenchindent}%
30.832   \addto\noextrasfrench{\bbl@nonfrenchindent}%
30.833 \fi

```

The layout of footnotes is handled at the `\begin{document}` depending on the values of flags **FrenchFootnotes** and **AutoSpaceFootnotes** (see section 30.10), nothing has to be done here for footnotes.

AutoSpacePunctuation adds an unbreakable space (in French only) before the four active characters (,:!?) even if none has been typed before them.

```

30.834 \ifBAutoSpacePunctuation
30.835   \autospace@beforeFDP
30.836 \else
30.837   \noautospace@beforeFDP
30.838 \fi

```

When **OriginalTypewriter** is set to false (the default), `\ttfamily`, `\rmfamily` and `\sffamily` are redefined as `\ttfamilyFB`, `\rmfamilyFB` and `\sffamilyFB` respectively to prevent addition of automatic spaces before the four active characters in computer code.

```

30.839 \ifBOriginalTypewriter
30.840 \else
30.841   \let\ttfamily\ttfamilyFB
30.842   \let\rmfamily\rmfamilyFB
30.843   \let\sffamily\sffamilyFB
30.844 \fi

```

ThinColonSpace changes the normal unbreakable space typeset in French before ‘:’ to a thin space.

```

30.845 \ifBThinColonSpace\renewcommand*{\Fcolonspace}{\thinspace}\fi

```

When true, **ThinSpaceInFrenchNumbers** redefines `numprint.sty`’s command `\npstylefrench` to set `\npthousandsep` to `\,` (`thinspace`) instead of `~` (default). This option has no effect if package `numprint.sty` is not loaded with ‘`autolanguage`’. As old versions of `numprint.sty` did not define `\npstylefrench`, we have to provide this command.

```

30.846 \@ifpackageloaded{numprint}%
30.847 {\ifnprt@autolanguage
30.848   \providecommand*{\npstylefrench}{}%
30.849   \ifBThinSpaceInFrenchNumbers
30.850     \renewcommand*{\npstylefrench{%
30.851       \npthousandsep{\,}%
30.852       \npdecimalsign{,}%
30.853       \npproductsign{\cdot}%
30.854       \npunitseparator{\,}%
30.855       \npdegreeseperator{}%
30.856       \nppercentseparator{\nprt@unitsep}%
30.857     }}%
30.858 \else

```

```

30.859 \renewcommand*\npstylefrench{%
30.860 \npthousandsep{~}%
30.861 \npdecimalsign{,}%
30.862 \npproductsign{\cdot}%
30.863 \npunitseparator{\,}%
30.864 \npdegreeseperator{}%
30.865 \nppercentseparator{\np@unitsep}%
30.866 }%
30.867 \fi
30.868 \npaddtolanguage{french}{french}%
30.869 \fi}%

```

FrenchSuperscripts: if true `\up=\fup`, else `\up=\textsuperscript`. Any-way `\up*=\FB@up@fake`. The star-form `\up*{}` is provided for fonts that lack some superior letters: Adobe Jenson Pro and Utopia Expert have no “g superior” for instance.

```

30.870 \ifFBFrenchSuperscripts
30.871 \DeclareRobustCommand*\up*\@ifstar{\FB@up@fake}{\fup}}%
30.872 \else
30.873 \DeclareRobustCommand*\up*\@ifstar{\FB@up@fake}%
30.874 {\textsuperscript}}%
30.875 \fi

```

LowercaseSuperscripts: if true let `\FB@lc` be `\lowercase`, else `\FB@lc` is redefined to do nothing.

```

30.876 \ifFBLowercaseSuperscripts
30.877 \else
30.878 \renewcommand*\FB@lc[1]{##1}%
30.879 \fi

```

PartNameFull: if false, redefine `\partname`.

```

30.880 \ifFBPartNameFull
30.881 \else\addto\captionsfrench{\def\partname{Partie}}\fi

```

ShowOptions: if true, print the list of all options to the `.log` file.

```

30.882 \ifFBShowOptions
30.883 \GenericWarning{* }{%
30.884 * **** List of possible options for frenchb ****\MessageBreak
30.885 [Default values between brackets when frenchb is loaded *LAST*]%
30.886 \MessageBreak
30.887 ShowOptions=true [false]\MessageBreak
30.888 StandardLayout=true [false]\MessageBreak
30.889 GlobalLayoutFrench=false [true]\MessageBreak
30.890 StandardLists=true [false]\MessageBreak
30.891 ReduceListSpacing=false [true]\MessageBreak
30.892 CompactItemize=false [true]\MessageBreak
30.893 StandardItemLabels=true [false]\MessageBreak
30.894 ItemLabels=\textendash, \textbullet,
30.895 \protect\ding{43},... [\textendash]\MessageBreak
30.896 ItemLabeli=\textendash, \textbullet,
30.897 \protect\ding{43},... [\textendash]\MessageBreak
30.898 ItemLabelii=\textendash, \textbullet,
30.899 \protect\ding{43},... [\textendash]\MessageBreak
30.900 ItemLabeliii=\textendash, \textbullet,
30.901 \protect\ding{43},... [\textendash]\MessageBreak
30.902 ItemLabeliv=\textendash, \textbullet,
30.903 \protect\ding{43},... [\textendash]\MessageBreak
30.904 IndentFirst=false [true]\MessageBreak
30.905 FrenchFootnotes=false [true]\MessageBreak
30.906 AutoSpaceFootnotes=false [true]\MessageBreak
30.907 AutoSpacePunctuation=false [true]\MessageBreak
30.908 OriginalTypewriter=true [false]\MessageBreak
30.909 ThinColonSpace=true [false]\MessageBreak
30.910 ThinSpaceInFrenchNumbers=true [false]\MessageBreak

```

```

30.911 FrenchSuperscripts=false [true]\MessageBreak
30.912 LowercaseSuperscripts=false [true]\MessageBreak
30.913 PartNameFull=false [true]\MessageBreak
30.914 og= <left quote character>, fg= <right quote character>
30.915 \MessageBreak
30.916 *****
30.917 \MessageBreak\protect\frenchbsetup{ShowOptions}}
30.918 \fi
30.919 }

```

At `\begin{document}` we save again the definitions of the ‘list’ and ‘itemize’ environments and the values of `labelitems` so that all changes made in the preamble are taken into account in languages other than French and in French with the `StandardLayout` option. We also have to provide an `\xspace` command in case the `xspace.sty` package is not loaded.

```

30.920 \AtBeginDocument{%
30.921   \let\listORI\list
30.922   \let\endlistORI\endlist
30.923   \let\itemizeORI\itemize
30.924   \let\enditemizeORI\enditemize
30.925   \let\@ltiORI\labelitemi
30.926   \let\@ltiiORI\labelitemii
30.927   \let\@ltiiiORI\labelitemiii
30.928   \let\@ltivORI\labelitemiv
30.929   \providecommand*\xspace{\relax}%

```

Let’s redefine some commands in `hyperref`’s bookmarks.

```

30.930 \ifundefined{pdfstringdefDisableCommands}{}%
30.931   {\pdfstringdefDisableCommands{%
30.932     \let\up\relax
30.933     \def\ieme{e\xspace}%
30.934     \def\iemes{es\xspace}%
30.935     \def\ier{er\xspace}%
30.936     \def\iers{ers\xspace}%
30.937     \def\iere{re\xspace}%
30.938     \def\ieres{res\xspace}%
30.939     \def\FrenchEnumerate#1{#1\degre\space}%
30.940     \def\FrenchPopularEnumerate#1{#1\degre)\space}%
30.941     \def\No{N\degre\space}%
30.942     \def\no{n\degre\space}%
30.943     \def\Nos{N\degre\space}%
30.944     \def\nos{n\degre\space}%
30.945     \def\og{\guillemotleft\space}%
30.946     \def\fg{\space\guillemotright}%
30.947     \let\bsc\textsc
30.948     \let\degres\degre
30.949   }}%

```

It is time to process the options set with `\frenchboptions{}`. Then execute either `\extrasfrench` and `\captionsfrench` or `\noextrasfrench` according to the current language at the `\begin{document}` (these three commands are updated by `\FBprocess@options`).

```

30.950 \FBprocess@options
30.951 \iflanguage{french}{\extrasfrench\captionsfrench}{\noextrasfrench}%

```

Some warnings are issued when output font encodings are not properly set. With XeLaTeX, `fontspec.sty` and `xunicode.sty` should be loaded; with (pdf)LaTeX, a warning is issued when OT1 encoding is in use at the `\begin{document}`. Mind that `\encodingdefault` is defined as ‘long’, defining `\FBOTone` with `\newcommand*` would fail!

```

30.952 \expandafter\ifx\csname XeTeXrevision\endcsname\relax
30.953   \begingroup \newcommand{\FBOTone}{OT1}%

```

```

30.954 \ifx\encodingdefault\FBOTone
30.955 \PackageWarning{frenchb.ldf}%
30.956 {OT1 encoding should not be used for French.
30.957 \MessageBreak
30.958 Add \protect\usepackage[T1]{fontenc} to the
30.959 preamble\MessageBreak of your document,}
30.960 \fi
30.961 \endgroup
30.962 \else
30.963 \@ifundefined{DeclareUTFcharacter}%
30.964 {\PackageWarning{frenchb.ldf}%
30.965 {Add \protect\usepackage{fontspec} *and*\MessageBreak
30.966 \protect\usepackage{xunicode} to the preamble\MessageBreak
30.967 of your document,}}%
30.968 {}%
30.969 \fi
30.970 }

```

30.14 Clean up and exit

Load `frenchb.cfg` (should do nothing, just for compatibility).

```
30.971 \loadlocalcfg{frenchb}
```

Final cleaning. The macro `\ldf@quit` takes care for setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value. The config file searched for has to be `frenchb.cfg`, and `\CurrentOption` has been set to ‘french’, so `\ldf@finish\CurrentOption` cannot be used: we first load `frenchb.cfg`, then call `\ldf@quit\CurrentOption`.

```

30.972 \FBclean@on@exit
30.973 \ldf@quit\CurrentOption

```


31 The Italian language

The file `italian.dtx`³⁰ defines all the language-specific macros for the Italian language.

The features of this language definition file are the following:

1. The Italian hyphenation is invoked, provided that file `ithyph.tex` was loaded when the $\text{\LaTeX} 2_\epsilon$ format was built; in case it was not, read the information coming with your distribution of the \TeX software, and the `babel` documentation.
2. The language dependent fixed words to be inserted by such commands as `\chapter`, `\caption`, `\tableofcontents`, etc. are redefined in accordance with the Italian typographical practice.
3. Since Italian can be easily hyphenated and Italian practice allows to break a word before the last two letters, hyphenation parameters have been set accordingly, but a very high demerit value has been set in order to avoid word breaks in the penultimate line of a paragraph. Specifically the `\clubpenalty`, and the `\widowpenalty` are set to rather high values and `\finalhyphendemerits` is set to such a high value that hyphenation is prohibited between the last two lines of a paragraph. In order to make it consistent, also `\@clubpenalty` is set to the same value; actually the latter value is the reset value after every sectioning command, so that after the first section, `\clubpenalty` is reset to the low default value. Thanks to Enrico Gregorio for spotting this serious bug.
4. Some language specific shortcuts have been defined so as to allow etymological hyphenation, specifically " inserts a break point in any word boundary that the typesetter chooses, provided it is not followed by an accented letter (very unlikely in Italian, where compulsory accents fall only on the last and ending vowel of a word, but may take place with compound words that include foreign roots), and "|" when the desired break point falls before an accented letter.
5. The shortcut "" introduces the raised (English) opening double quotes; this shortcut proves its usefulness when one reminds that the Italian keyboard misses the backtick key, and the backtick on a Windows based platform may be obtained only by pressing the `Alt` key while inputting the numerical code 0096; very, very annoying!
6. The shortcuts "< and "> insert the French guillemots, sometimes used in Italian typography; with the T1 font encoding the ligatures << and >> should insert such signs directly, but not all the virtual fonts that claim to follow the T1 font encoding actually contain the guillemots; with the OT1 encoding the guillemots are not available and must be faked in some way. By using the "< and "> shortcuts (even with the T1 encoding) the necessary tests are performed and in case the suitable glyphs are taken from other fonts normally available with any good, modern \LaTeX distribution.
7. Three new specific commands `\unit`, `\ped`, and `\ap` are introduced so as to enable the correct composition of technical mathematics according to the ISO 31/XI recommendations. `\unit` does not get redefined if the `babel` package is loaded *after* the package `units.sty` whose homonymous command plays a different role and follows a different syntax.

³⁰The file described in this section has version number v1.2t and was last revised on 2008/03/14. The original author is Maurizio Codogno, (mau@beatles.cselt.stet.it). It has been largely revised by Johannes Braams and Claudio Beccari

For this language a limited number of shortcuts has been defined, table 6, some of which are used to overcome certain limitations of the Italian keyboard; in section 31.3 there are other comments and hints in order to overcome some other keyboard limitations.

"	inserts a compound word mark where hyphenation is legal; it allows etymological hyphenation which is recommended for technical terms, chemical names and the like; it does not work if the next character is represented with a control sequence or is an accented character.
"	the same as the above without the limitation on characters represented with control sequences or accented ones.
" "	inserts open quotes “.
"<	inserts open guillemots.
">	inserts closed guillemots.
"/	equivalent to <code>\slash</code>

Table 6: Shortcuts for the Italian language

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
31.1 (*code)
31.2 \LdfInit{italian}{captionsitalian}%
```

When this file is read as an option, i.e. by the `\usepackage` command, `italian` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@italian` to see whether we have to do something here.

```
31.3 \ifx\l@italian\@undefined
31.4     \@nopatterns{Italian}%
31.5     \adddialect\l@italian0\fi
```

The next step consists of defining commands to switch to (and from) the Italian language.

`\captionsitalian` The macro `\captionsitalian` defines all strings used in the four standard document classes provided with L^AT_EX.

```
31.6 \addto\captionsitalian{%
31.7     \def\prefacename{Prefazione}%
31.8     \def\refname{Riferimenti bibliografici}%
31.9     \def\abstractname{Sommario}%
31.10    \def\bibname{Bibliografia}%
31.11    \def\chaptername{Capitolo}%
31.12    \def\appendixname{Appendice}%
31.13    \def\contentsname{Indice}%
31.14    \def\listfigurename{Elenco delle figure}%
31.15    \def\listtablename{Elenco delle tabelle}%
31.16    \def\indexname{Indice analitico}%
31.17    \def\figurename{Figura}%
31.18    \def\tablename{Tabella}%
31.19    \def\partname{Parte}%
31.20    \def\enclname{Allegati}%
31.21    \def\ccname{e`p.~c.}%
31.22    \def\headtoname{Per}%
31.23    \def\pagename{Pag.}%      % in Italian the abbreviation is preferred
31.24    \def\seename{vedi}%
31.25    \def\alsoname{vedi anche}%
31.26    \def\proofname{Dimostrazione}%
31.27    \def\glossaryname{Glossario}%
31.28 }
```

`\dateitalian` The macro `\dateitalian` redefines the command `\today` to produce Italian dates.

```
31.29 \def\dateitalian{%
31.30   \def\today{\number\day~\ifcase\month\or
31.31     gennaio\or febbraio\or marzo\or aprile\or maggio\or giugno\or
31.32     luglio\or agosto\or settembre\or ottobre\or novembre\or
31.33     dicembre\fi\space \number\year}}%
```

`\italianhyphenmins` The italian hyphenation patterns can be used with both `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
31.34 \providehyphenmins{\CurrentOption}{\tw@\tw@}
```

`\extrasitalian` Lower the chance that clubs or widows occur.

```
\noextrasitalian 31.35 \addto\extrasitalian{%
31.36   \babel@savevariable\clubpenalty
31.37   \babel@savevariable\widowpenalty
31.38   \babel@savevariable\clubpenalty
31.39   \clubpenalty3000\widowpenalty3000\clubpenalty\clubpenalty}%
```

Never ever break a word between the last two lines of a paragraph in italian texts.

```
31.40 \addto\extrasitalian{%
31.41   \babel@savevariable\finalhyphendemerits
31.42   \finalhyphendemerits50000000}%
```

In order to enable the hyphenation of words such as “nell’altezza” we give the ‘ a non-zero lower case code. When we do that T_EX finds the following hyphenation points nel-l’al-tez-za instead of none.

```
31.43 \addto\extrasitalian{%
31.44   \lccode‘’=‘’}%
31.45 \addto\noextrasitalian{%
31.46   \lccode‘’=0}%
```

31.1 Support for etymological hyphenation

In his article on Italian hyphenation [1] Beccari pointed out that the Italian language gets hyphenated on a phonetic basis, although etymological hyphenation is allowed; this is in contrast with what happens in Latin, for example, where etymological hyphenation is always used. Since the patterns for both languages would become too complicated in order to cope with etymological hyphenation, in his paper Beccari proposed the definition of an active character ‘_’ such that it could insert a “soft” discretionary hyphen at the compound word boundary. For several reasons that idea and the specific active character proved to be unpractical and was abandoned.

This problem is so important with the majority of the European languages, that `babel` from the very beginning developed the tradition of making the “ character active so as to perform several actions that turned useful with every language. One of these actions consisted in defining the shortcut “|” that was extensively used in German and in many other languages in order to insert a discretionary hyphen such that hyphenation would not be precluded in the rest of the word as it happens with the standard T_EX command `\-`.

Meanwhile the `ec` fonts with the double Cork encoding (thus formerly called the `dc` fonts) have become more or less standard and are widely used by virtually all Europeans that write languages with many special national characters; by so doing they avoid the use of the `\accent` primitive which would be required with the standard `cm` fonts; with the latter fonts the primitive command `\accent` is such that hyphenation becomes almost impossible, in any case strongly impeached.

The `ec` fonts contain a special character, named “compound word mark”, that occupies position 23 in the font scheme and may be input with the sequence `^^W`.

Up to now, apparently, this special character has never been used in a practical way for the typesetting of languages rich of compound words; also it has never been inserted in the hyphenation pattern files of any language. Beccari modified his pattern file `ithyph.tex v4.8b` for Italian so as to contain five new patterns that involve `^^W`, and he tried to give the `babel` active character `"` a new shortcut definition, so as to allow the insertion of the “compound word mark” in the proper place within any word where two semantic fragments join up. With such facility for marking the compound word boundaries, etymological hyphenation becomes possible even if the patterns know nothing about etymology (but the typesetter hopefully does!). In Italian such etymological hyphenation is desirable with technical terms, chemical names, and the like.

Even this solution proved to be inconvenient on certain UN*X platforms, so Beccari resorted to another approach that uses the `babel` active character `"` and relies on the category code of the character that follows `"`.

```
31.47 \initiate@active@char{"}%
31.48 \addto\extrasitalian{\bbl@activate{"}\languageshorthands{italian}}%
```

`\it@cwm` The active character `"` is now defined for language `italian` so as to perform different actions in math mode compared to text mode; specifically in math mode a double quote is inserted so as to produce a double prime sign, while in text mode the temporary macro `\it@next` is defined so as to defer any further action until the next token category code has been tested.

```
31.49 \declare@shorthand{italian}{"}{%
31.50 \ifmmode
31.51   \def\it@next{''}%
31.52 \else
31.53   \def\it@next{\futurelet\it@temp\it@cwm}%
31.54 \fi
31.55 \it@next
31.56 }%
```

`\it@cwm` The `\it@next` service control sequence is such that upon its execution a temporary variable `\it@temp` is made equivalent to the next token in the input list without actually removing it. Such temporary token is then tested by the macro `\it@cwm` and if it is found to be a letter token, then it introduces a compound word separator control sequence `\it@allowhyphens` whose expansion introduces a discretionary hyphen and an unbreakable space; in case the token is not a letter, then it is tested against `|12`: if so a compound word separator is inserted and the `|` token is removed, otherwise another test is performed so as to see if another double quote sign follows: in this case a double open quote mark is inserted, otherwise two other tests are performed so as to see if guillemets have to be inserted, otherwise nothing is done. The double quote shortcut for inserting a double open quote sign is useful for people who are inputting Italian text by means of an Italian keyboard that unfortunately misses the grave or backtick key. By this shortcut `"` becomes equivalent to `“` for inserting raised open high double quotes.

```
31.57 \def\it@@cwm{\nobreak\discretionary{-}{-}{\nobreak\hskip\z@skip}%
31.58 \def\it@@ocap#1{\it@ocap}\def\it@@ccap#1{\it@ccap}%
31.59 \DeclareRobustCommand*\it@cwm{\let\it@next\relax
31.60 \ifcat\noexpand\it@temp a%
31.61   \def\it@@next{\it@@cwm}%
31.62 \else
31.63   \if\noexpand\it@temp \string|%
31.64     \def\it@@next{\it@@cwm\@gobble}%
31.65   \else
31.66     \if\noexpand\it@temp \string<%
31.67       \def\it@@next{\it@@ocap}%
31.68     \else
31.69       \if\noexpand\it@temp \string>%
31.70         \def\it@@next{\it@@ccap}%

```

```

31.71         \else
31.72         \if\noexpand\it@temp\string/%
31.73         \def\it@@next{\slash\@gobble}%
31.74         \else
31.75         \ifx\it@temp"%
31.76         \def\it@@next{''\@gobble}%
31.77         \fi
31.78         \fi
31.79     \fi
31.80 \fi
31.81 \fi
31.82 \fi
31.83 \it@@next}%

```

By this definition of " if one types macro"istruzione the possible break points become ma-cro-istru-zio-ne, while without the " mark they would be ma-croi-stru-zio-ne, according to the phonetic rules such that the macro prefix is not taken as a unit. A chemical name such as des"clor"fenir"amina"cloridrato is breakable as des-clor-fe-nir-ami-na-clo-ri-dra-to instead of de-sclor-fe-ni-ra-mi-na-...

In other language description files a shortcut is defined so as to allow a break point without actually inserting any hyphen sign; examples are given such as entrada/salida; actually if one wants to allow a breakpoint after the slash, it is much clearer to type \slash instead of / and L^AT_EX does everything by itself; here the shortcut "/" was introduced to stand for \slash so that one can type input"/output and allow a line break after the slash. This shortcut works only for the slash, since in Italian such constructs are extremely rare.

Attention: the expansion of " takes place before the actual expansion of OT1 or T1 accented sequences such as \'{a}; therefore this etymological hyphenation facility works as it should only when the semantic word fragments *do not start* with an accented letter; this in Italian is always avoidable, because compulsory accents fall only on the last vowel, but it may be necessary to mark a compound word where one or more components come from a foreign language and contain diacritical marks according to the spelling rules of that language. In this case the special shorthand "| may be used that performs exactly as " normally does, except that the | sign is removed from the token input list: kilo"|\o}rsted gets hyphenated as ki-lo-ör-sted.

31.2 Facilities required by the ISO 31/XI regulations

The ISO 31/XI regulations require that units of measure are typeset in upright font in any circumstance, math or text, and that in text mode they are separated from the numerical indication of the measure with an unbreakable (thin) space. The command \unit that was defined for achieving this goal happened to conflict with the homonymous command defined by the package `units.sty`; we therefore need to test if that package has already been loaded so as to avoid conflicts; we assume that if the user loads that package, s/he wants to use that package facilities and command syntax.

The same regulations require also that super and subscripts (apices and pedices) are in upright font, *not in math italics*, when they represent "adjectives" or appositions to mathematical or physical variables that do not represent countable or measurable entities such as, for example, V_{\max} or V_{rms} for a maximum or a root mean square voltage, compared to V_i or V_T as the i -th voltage in a set, or a voltage that depends on the thermodynamic temperature T . See [2] for a complete description of the ISO regulations in connection with typesetting.

More rarely it happens to use superscripts that are not mathematical variables, such as the notation \mathbf{A}^T to denote the transpose of matrix \mathbf{A} ; text superscripts are useful also as ordinals or in old fashioned abbreviations in text mode; for example the feminine ordinal 1^a or the old fashioned obsolete abbreviation F^{lli} for Fratelli

in company names (compare with “Bros.” for brothers in American English); text subscripts are mostly used in logos.

`\unit` First we define the new (internal) commands `\bbl@unit`, `\bbl@ap`, and `\bbl@ped`
`\ap` as robust ones.

```
\ped 31.84 \@ifpackageloaded{units}{-}{%
31.85   \DeclareRobustCommand*{\bbl@unit}[1]{%
31.86     \textormath{\,\mbox{#1}}{-}\,\mathrm{#1}}}%
31.87   }%
31.88 \DeclareRobustCommand*{\bbl@ap}[1]{%
31.89   \textormath{\textsuperscript{#1}}{-}\mathrm{#1}}}%
31.90 \DeclareRobustCommand*{\bbl@ped}[1]{%
31.91   \textormath{\$_{\mbox{\fontsize\sf@size\z@
31.92     \selectfont#1}}$}{-}\mathrm{#1}}}%
```

Then we can use `\let` to define the user level commands, but in case the macros already have a different meaning before entering in Italian mode typesetting, we first memorize their meaning so as to restore them on exit.

```
31.93 \@ifpackageloaded{units}{-}{%
31.94   \addto\extrasitalian{%
31.95     \babel@save\unit\let\unit\bbl@unit}%
31.96   }%
31.97 \addto\extrasitalian{%
31.98   \babel@save\ap\let\ap\bbl@ap
31.99   \babel@save\ped\let\ped\bbl@ped
31.100 }%
```

31.3 Accents

Most of the other language description files introduce a number of shortcuts for inserting accents and other language specific diacritical marks in a more comfortable way compared with the lengthy standard \TeX conventions. When an Italian keyboard is being used on a Windows based platform, it exhibits such limitations that up to now no convenient shortcuts have been developed; the reason lies in the fact that the Italian keyboard lacks the grave accent (also known as “backtick”), which is compulsory on all accented vowels except the ‘e’, but, on the opposite, it carries the keys with all the accented lowercase vowels; the keyboard lacks also the tie \sim (tilde) key, while the curly braces require pressing three keys simultaneously.

The best solution Italians have found so far is to use a smart editor that accepts shortcut definitions such that, for example, by striking “(one gets directly { on the screen and the same sign is saved into the `.tex` file; the same smart editor should be capable of translating the accented characters into the standard \TeX sequences when writing a file to disk (for the sake of file portability), and to transform the standard \TeX sequences into the corresponding signs when loading a `.tex` file from disk to memory. Such smart editors do exist and can be downloaded from the CTAN archives.

For what concerns the missing backtick key, which is used also for inputting the open quotes, it must be noticed that the shortcut “” described above completely solves the problem for *double* raised open quotes; according to the traditions of particular publishing houses, since there are no compulsory regulations on the matter, the French guillemets may be used; in this case the T1 font encoding solves the problem by means of its built in ligatures << and >>. But...

31.4 *Caporali* or French double quotes

Although the T1 font encoding ligatures solve the problem, there are some circumstances where even the T1 font encoding cannot be used, either because the author/typesetter wants to use the OT1 encoding, or because s/he uses a font set that does not comply completely with the T1 font encoding; some virtual fonts,

for example, are supposed to implement the double Cork font encoding but actually miss some glyphs; one such virtual font set is given by the `ae` virtual fonts, because they are supposed to implement such double font encoding simply using the `cm` fonts, of which the type 1 PostScript version exists and is freely available. Since guillemets (in Italian *caporali*) do not exist in any `cm` latin font, their glyphs must be substituted with something else that approaches them.

Since in French typesetting guillemets are compulsory, the French language definition file resorts to a clever font substitution; such file exploits the $\text{\LaTeX} 2_{\epsilon}$ font selection machinery so as to get the guillemets from the Cyrillic fonts, because it suffices to locally change the default encoding. There are several sets of Cyrillic fonts, but the ones that obey the OT2 font encoding are generally distributed with all recent implementations of the \TeX software; they are part of the American Mathematical Society fonts and come both as METAFONT source files and Type 1 PostScript `.pfb` files. The availability of such fonts should be guaranteed by the presence of the `OT2cmr.fd` font description file. Actually the presence of this file does not guarantee the completeness of your \TeX implementation; should \LaTeX complain about a missing Cyrillic `.tfm` file (that kind of file that contains the font metric parameters) and/or about missing Cyrillic (`.mf`) files, then your \TeX system is *incomplete* and you should download such fonts from the CTAN archives. Temporarily you may issue the command `\LtxSymbCaporali` so as to approximate the missing glyphs with the \LaTeX symbol fonts. In some case warning messages are issued so as to inform the typesetter about the necessity of resorting to some *poor man* solution.

In spite of these alternate fonts, we must avoid invoking unusual fonts if the available encoding allows to use built in caporali. As far as I know (CB) the only T1-encoded font families that miss the guillemets are the AE ones; we therefore first test if the default encoding is the T1 one and in this case if the AE families are the default ones; in order for this to work properly it is necessary to load these optional packages *before* `babel`. If the T1 encoding is not the default one when the Italian language is specified, then some substitutions must be done.

`\LtxSymbCaporali` We define some macros for substituting the default guillemets; first the emulation
`\it@ocap` by means of the \LaTeX symbols; each one of these macro sets actually redefines the
`\it@ccap` control sequences `\it@ocap` and `\it@ccap` that are the ones effectively activated
by the shortcuts "< and ">.

```

31.101 \def\LtxSymbCaporali{%
31.102   \DeclareRobustCommand*\it@ocap*\mbox{%
31.103     \fontencoding{U}\fontfamily{lasy}\selectfont(\kern-0.20em){}%
31.104     \ignorespaces}%
31.105   \DeclareRobustCommand*\it@ccap*\ifdim\lastskip>\z@&\unskip\fi
31.106   \mbox{%
31.107     \fontencoding{U}\fontfamily{lasy}\selectfont)\kern-0.20em)}}%
31.108 }%
```

Then the substitution with any specific font that contains such glyphs; it might be the CBgreek fonts, the Cyrillic one, the super-cm ones, the `lm` ones, or any other the user might prefer (the code is adapted from the one that appears in the `frenchb.ld` file; thanks to Daniel Flipo). By default if the user did not select the T1 encoding, the existence of the CBgreek fonts is tested; if they exist the guillemets are taken from this font, and since its families are a superset of the default CM ones and they apply also to typeset slides with the standard class `slides`. If the CBgreek fonts are not found, then the existence of the Cyrillic ones is tested, although this choice is not suited for typesetting slides; otherwise the poor man solution of the \LaTeX special symbols is used. In any case the user can force the use of the Cyrillic guillemets substitution by issuing the declaration `\CyrillicCaporali` just before the `\begin{document}` statement; in alternative the user can specify with

`\CaporaliFrom{<encoding>}{<family>}{<opening number>}{<closing number>}`

the encoding and family of the font s/he prefers, and the slot numbers of the opening and closing guillemets respectively. For example if the T1-encoded Latin Modern fonts are desired the specific command should be

```
\CaporaliFrom{T1}{lmr}{19}{20}
```

These user choices might be necessary for assuring the correct typesetting with fonts that contain the required glyphs and are available also in PostScript form so as to use them directly with `pdflatex`, for example.

```
31.109 \def\CaporaliFrom#1#2#3#4{%
31.110   \DeclareFontEncoding{#1}{-}{-}%
31.111   \DeclareTextCommand{\it@ocap}{T1}{%
31.112     {\fontencoding{#1}\fontfamily{#2}\selectfont\char#3\ignorespaces}}%
31.113   \DeclareTextCommand{\it@ccap}{T1}{\ifdim\lastskip>z@\unskip\fi%
31.114     {\fontencoding{#1}\fontfamily{#2}\selectfont\char#4}}%
31.115   \DeclareTextCommand{\it@ocap}{OT1}{%
31.116     {\fontencoding{#1}\fontfamily{#2}\selectfont\char#3\ignorespaces}}%
31.117   \DeclareTextCommand{\it@ccap}{OT1}{\ifdim\lastskip>z@\unskip\fi%
31.118     {\fontencoding{#1}\fontfamily{#2}\selectfont\char#4}}}
```

Then we set a boolean variable and test the default family; if such family has a name that starts with the letters “ae” then we have no built in guillemets; of course if the AE font family is chosen after the `babel` package is loaded, the test does not perform as required.

```
31.119 \def\get@ae#1#2#3!\def\bbl@ae{#1#2}}%
31.120 \def\@ifT@one@noCap{\expandafter\get@ae\fontfamily!%
31.121 \def\bbl@temp{ae}\ifx\bbl@ae\bbl@temp\expandafter\@firstoftwo\else
31.122   \expandafter\@secondoftwo\fi}%

```

We set another couple of boolean variables for testing the existence of the CBgreek or the Cyrillic fonts

```
31.123 \newif\if@CBgreekEncKnown
31.124 \IfFileExists{lgrcmr.fd}%
31.125   {\@CBgreekEncKnowntrue}{\@CBgreekEncKnownfalse}
31.126 \newif\if@CyrEncKnown
31.127 \IfFileExists{ot2cmr.fd}%
31.128   {\@CyrEncKnowntrue}{\@CyrEncKnownfalse}%

```

`\CBgreekCaporali` Next we define the macros `\CBgreekCaporali`, `\T@unoCaporali`, and `\CyrillicCaporali`; with the latter one we test the loaded class, and if it’s `slides` nothing gets done. In `\CyrillicCaporali` any case each one of these declarations, if used, must be specified in the preamble.

```
31.129 \def\CBgreekCaporali{\@ifclassloaded{slides}{%
31.130   \IfFileExists{lgrlcmss.fd}{\DeclareFontEncoding{LGR}{-}{-}%
31.131   \DeclareRobustCommand*\it@ccap{%
31.132     {\ifdim\lastskip>z@\unskip\fi
31.133     {\fontencoding{LGR}\selectfont}}}%
31.134   \DeclareRobustCommand*\it@ocap{%
31.135     {\fontencoding{LGR}\selectfont(\)\ignorespaces}}%
31.136   {\LtxSymbCaporali}}%
31.137   {\DeclareFontEncoding{LGR}{-}{-}%
31.138   \DeclareRobustCommand*\it@ccap{%
31.139     {\ifdim\lastskip>z@\unskip
31.140     \fi{\fontencoding{LGR}\selectfont}}}%
31.141   \DeclareRobustCommand*\it@ocap{%
31.142     {\fontencoding{LGR}\selectfont(\)\ignorespaces}}%
31.143   }%
31.144 \def\CyrillicCaporali{\@ifclassloaded{slides}{\relax}%
31.145   {\DeclareFontEncoding{OT2}{-}{-}%
31.146   \DeclareRobustCommand*\it@ccap{%
31.147     {\ifdim\lastskip>z@\unskip\fi
31.148     {\fontencoding{OT2}\selectfont\char62\relax}}%
31.149   \DeclareRobustCommand*\it@ocap{%
31.150     {\fontencoding{OT2}\selectfont\char60\relax}\ignorespaces}}}%

```



```

31.151 \@onlypreamble{\CBgreekCaporali}\@onlypreamble{\CyrillicCaporali}%
31.152 \def\T@unoCaporali{\DeclareRobustCommand*{\it@ocap}{<<\ignorespaces}%
31.153     \DeclareRobustCommand*{\it@ccap}{\ifdim\lastskip>z@\unskip\fi>>}}%

```

Now we can do some real setting; first we start testing the encoding; if the encoding is T1 we test if the font family is the AE one; if so, we further test for other possibilities

```

31.154 \ifx\cf@encoding\bbl@t@one
31.155   \@ifT@one@noCap{%
31.156     \if@CBgreekEncKnown
31.157       \CBgreekCaporali
31.158     \else
31.159       \if@CyrEncKnown
31.160         \CyrillicCaporali
31.161       \else
31.162         \LtxSymbCaporali
31.163     \fi
31.164   \fi}%
31.165   {\T@unoCaporali}%

```

But if the default encoding is not the T1 one, then the substitutions must be performed.

```

31.166 \else
31.167   \if@CBgreekEncKnown
31.168     \CBgreekCaporali
31.169   \else
31.170     \if@CyrEncKnown
31.171       \CyrillicCaporali
31.172     \else
31.173       \LtxSymbCaporali
31.174   \fi
31.175 \fi
31.176 \fi

```

31.5 Finishing commands

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

31.177 \ldf@finish{italian}%
31.178 </code>

```

31.6 References

- [1] Beccari C., “Computer Aided Hyphenation for Italian and Modern Latin”, TUGboat vol. 13, n. 1, pp. 23-33 (1992).
- [2] Beccari C., “Typesetting mathematics for science and technology according to ISO 31/XI”, TUGboat vol. 18, n. 1, pp. 39-48 (1997).

32 The Latin language

The file `latin.dtx`³¹ defines all the language-specific macros for the Latin language both in modern and medieval spelling.

For this language the `\clubpenalty`, `\widowpenalty` are set to rather high values and `\finalhyphendemerits` is set to such a high value that hyphenation is prohibited between the last two lines of a paragraph.

For this language two “styles” of typesetting are implemented: “regular” or modern-spelling Latin, and medieval Latin. The medieval Latin specific commands can be activated by means of the language attribute `medieval`; the medieval spelling differs from the modern one by the systematic use of the lower case ‘u’ also where in modern spelling the letter ‘v’ is used; when typesetting with capital letters, on the opposite, the letter ‘V’ is used also in place of ‘U’. Medieval spelling also includes the ligatures `\ae` (æ), `\oe` (œ), `\AE` (Æ), and `\OE` (E) that are not used in modern spelling, nor were used in the classical times.

Furthermore a third typesetting style `withprosodicmarks` is defined in order to use special shortcuts for inserting breves and macrons when typesetting grammars, dictionaries, teaching texts, and the like, where prosodic marks are important for the complete information on the words or the verses. The shortcuts, listed in table 7 and described in section 33, may interfere with other packages; therefore by default this third style is off and no interference is introduced. If this third style is used and interference is experienced, there are special commands for turning on and off the specific short hand commands of this style.

For what concerns `babel` and typesetting with \LaTeX , the differences between the two styles of spelling reveal themselves in the strings used to name for example the “Preface” that becomes “Praefatio” or “Præfatio” respectively. Hyphenation rules are also different, but the hyphenation pattern file `lahyph.tex` takes care of both versions of the language. Needless to say that such patterns must be loaded in the \LaTeX format by running `initex` (or whatever the name of the initializer) on `latex.ltx`.

The name strings for chapters, figures, tables, etcetera, are suggested by prof. Raffaella Tabacco, a classicist of the University of Turin, Italy, to whom we address our warmest thanks. The names suggested by Krzysztof Konrad Żelechowski, when different, are used as the names for the medieval variety, since he made a word and spelling choice more suited for this variety.

For this language some shortcuts are defined according to table 7; all of them are supposed to work with both spelling styles, except where the opposite is explicitly stated.

<code>~i</code>	inserts the breve accent as ĭ; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures æ and œ.
<code>=a</code>	inserts the macron accent as ā; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures æ and œ.
<code>"</code>	inserts a compound word mark where hyphenation is legal; the next character must not be a medieval ligature æ or œ, nor an accented letter (foreign names).
<code>" </code>	same as above, but operates also when the next character is a medieval ligature or an accented letter.

Table 7: Shortcuts defined for the Latin language. The characters `~` and `=` are active only when the language attribute `withprosodicmarks` has been declared, otherwise they are disabled; see section 33 for more details.

³¹The file described in this section has version number v2.0l and was last revised on 2008/07/06. The original author is Claudio Beccari with contributions by Krzysztof Konrad Żelechowski, (`kkz@alfa.mimuw.edu.pl`)

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
32.1 (*code)
32.2 \LdfInit{latin}{captionslatin}
```

When this file is read as an option, i.e. by the `\usepackage` command, `latin` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@latin` to see whether we have to do something here.

```
32.3 \ifx\l@latin\@undefined
32.4     \@nopatterns{Latin}
32.5     \adddialect\l@latin0\fi
```

Now we declare the `medieval` language attribute.

```
32.6 \bbl@declare@ttribute{latin}{medieval}{%
32.7     \addto\captionslatin{\def\prefacename{Pr{\ae}fatio}}%
32.8     \def\november{Nouembris}%
32.9     \expandafter\addto\expandafter\extraslatin
32.10    \expandafter{\extrasmedievallatin}%
32.11 }
```

The third typesetting style with `prosodicmarks` is defined here

```
32.12 \bbl@declare@ttribute{latin}{withprosodicmarks}{%
32.13     \expandafter\addto\expandafter\extraslatin
32.14     \expandafter{\extraswithprosodicmarks}%
32.15 }
```

It must be remembered that the `medieval` and the `withprosodicmarks` styles may be used together.

The next step consists of defining commands to switch to (and from) the Latin language³².

`\captionslatin` The macro `\captionslatin` defines all strings used in the four standard document classes provided with L^AT_EX.

```
32.16 \namedef{captionslatin}{%
32.17     \def\prefacename{Praefatio}%
32.18     \def\refname{Conspectus librorum}%
32.19     \def\abstractname{Summarium}%
32.20     \def\bibname{Conspectus librorum}%
32.21     \def\chaptername{Caput}%
32.22     \def\appendixname{Additamentum}%
32.23     \def\contentsname{Index}%
32.24     \def\listfigurename{Conspectus descriptionum}%
32.25     \def\listtablename{Conspectus tabularum}%
32.26     \def\indexname{Index rerum notabilium}%
32.27     \def\figurename{Descriptio}%
32.28     \def\tablename{Tabula}%
32.29     \def\partname{Pars}%
32.30     \def\enclname{Adduntur}% Or " Additur" ? Or simply Add.?
32.31     \def\ccname{Exemplar}% Use the recipient's dative
32.32     \def\headtoname{\ignorespaces}% Use the recipient's dative
32.33     \def\pagename{Charta}%
32.34     \def\seename{cfr.}%
32.35     \def\alsoname{cfr.}% R.Tabacco never saw "cfr. atque" or similar forms
32.36     \def\proofname{Demonstratio}%
32.37     \def\glossaryname{Glossarium}%
32.38 }
```

In the above definitions there are some points that might change in the future or that require a minimum of attention from the typesetter.

³²Most of these names were kindly suggested by Raffaella Tabacco.

1. the `\enclname` is translated by a passive verb, that literally means “(they) are being added”; if just one enclosure is joined to the document, the plural passive is not suited any more; nevertheless a generic plural passive might be incorrect but suited for most circumstances. On the opposite “Additur”, the corresponding singular passive, might be more correct with one enclosure and less suited in general: what about the abbreviation “Add.” that works in both cases, but certainly is less elegant?
2. The `\headtoname` is empty and gobbles the possible following space; in practice the typesetter should use the dative of the recipient’s name; since nowadays not all such names can be translated into Latin, they might result indeclinable. The clever use of an appellative by the typesetter such as “Domino” or “Dominae” might solve the problem, but the header might get too impressive. The typesetter must make a decision on his own.
3. The same holds true for the copy recipient’s name in the “Cc” field of `\ccname`.

`\datelatin` The macro `\datelatin` redefines the command `\today` to produce Latin dates; the choice of faked small caps Latin numerals is arbitrary and may be changed in the future. For medieval latin the spelling of ‘Novembris’ should be *Novembris*. This is taken care of by using a control sequence which can be redefined when the attribute ‘medieval’ is selected.

```

32.39 \def\datelatin{%
32.40   \def\november{Novembris}%
32.41   \def\today{%
32.42     {\check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont
32.43       \uppercase\expandafter{\romannumeral\day}}~\ifcase\month\or
32.44       Ianuarii\or Februarii\or Martii\or Aprilis\or Maii\or Iunii\or
32.45       Iulii\or Augusti\or Septembris\or Octobris\or \november\or
32.46       Decembris\fi
32.47   \space{\uppercase\expandafter{\romannumeral\year}}}
```

`\romandate` Thomas Martin Widmann (viralbus@daimi.au.dk) developed a macro originally named `\latindate` (but to be renamed `\romandate` so as not to conflict with the standard `babel` conventions) that should compute and translate the current date into a date *ab urbe condita* with days numbered according to the kalendae and idus; for the moment this is a placeholder for Thomas’ macro, waiting for a self standing one that keeps local all the intermediate data, counters, etc. If he succeeds, here is the place to add his macro.

`\latinhyphenmins` The Latin hyphenation patterns can be used with both `\lefthyphenmin` and `\righthyphenmin` set to 2.

```

32.48 \providehyphenmins{\CurrentOption}{\tw@\tw@}
```

`\extraslatin` Lower the chance that clubs or widows occur.

```

\noextraslatin 32.49 \addto\extraslatin{%
32.50   \babel@savevariable\clubpenalty
32.51   \babel@savevariable\@clubpenalty
32.52   \babel@savevariable\widowpenalty
32.53   \clubpenalty3000\@clubpenalty3000\widowpenalty3000}
```

Never ever break a word between the last two lines of a paragraph in latin texts.

```

32.54 \addto\extraslatin{%
32.55   \babel@savevariable\finalhyphendemerits
32.56   \finalhyphendemerits50000000}
```

With medieval Latin we need the suitable correspondence between upper case V and lower case u, since in that spelling there is only one sign, and the u shape is the (uncial) version of the capital V. Everything else is identical with Latin.

```

32.57 \addto\extrasmedievallatin{%
```

```

32.58 \babel@savevariable{\lccode'\V}%
32.59 \babel@savevariable{\uccode'\u}%
32.60 \lccode'\V='\'u \uccode'\u='\'V}

```

\SetLatinLigatures We need also the lccodes for æ and œ; since they occupy different positions in the OT1 T_EX-fontencoding compared to the T1 one, we must save the lc- and the uccodes for both encodings, but we specify the new lc- and uccodes separately as it appears natural not to change encoding while typesetting the same language. The encoding is assumed to be set before starting to use the Latin language, so that if Latin is the default language, the font encoding must be chosen before requiring the **babel** package with the **latin** option, in any case before any **\selectlanguage** or **\foreignlanguage** command.

All this fuss is made in order to allow the use of the medieval ligatures æ and œ while typesetting with the medieval spelling; I have my doubts that the medieval spelling should be used at all in modern books, reports, and the like; the uncial ‘u’ shape of the lower case ‘v’ and the above ligatures were fancy styles of the copyists who were able to write faster with those rounded glyphs; with typesetting there is no question of handling a quill penn. . . Since my (CB) opinion may be wrong, I managed to set up the instruments and it is up to the typesetter to use them or not.

```

32.61 \addto\extrasmedievallatin{%
32.62 \babel@savevariable{\lccode'\^e6}% T1 \ae
32.63 \babel@savevariable{\uccode'\^e6}% T1 \ae
32.64 \babel@savevariable{\lccode'\^c6}% T1 \AE
32.65 \babel@savevariable{\lccode'\^f7}% T1 \oe
32.66 \babel@savevariable{\uccode'\^f7}% T1 \OE
32.67 \babel@savevariable{\lccode'\^d7}% T1 \OE
32.68 \babel@savevariable{\lccode'\^1a}% OT1 \ae
32.69 \babel@savevariable{\uccode'\^1a}% OT1 \ae
32.70 \babel@savevariable{\lccode'\^1d}% OT1 \AE
32.71 \babel@savevariable{\lccode'\^1b}% OT1 \oe
32.72 \babel@savevariable{\uccode'\^1b}% OT1 \OE
32.73 \babel@savevariable{\lccode'\^1e}% OT1 \OE
32.74 \SetLatinLigatures}
32.75 \providecommand\SetLatinLigatures{%
32.76 \def\@tempA{T1}\ifx\@tempA\fontencoding
32.77 \catcode'\^e6=11 \lccode'\^e6='\'e6 \uccode'\^e6='\'c6 % \ae
32.78 \catcode'\^c6=11 \lccode'\^c6='\'e6 % \AE
32.79 \catcode'\^f7=11 \lccode'\^f7='\'f7 \uccode'\^f7='\'d7 % \oe
32.80 \catcode'\^d7=11 \lccode'\^d7='\'f7 % \OE
32.81 \else
32.82 \catcode'\^1a=11 \lccode'\^1a='\'1a \uccode'\^1a='\'1d % \ae
32.83 \catcode'\^1d=11 \lccode'\^1d='\'1a % \AE (^)]
32.84 \catcode'\^1b=11 \lccode'\^1b='\'1b \uccode'\^1b='\'1e % \oe
32.85 \catcode'\^1e=11 \lccode'\^1e='\'1b % \OE (^^)
32.86 \fi
32.87 \let\@tempA\@undefined
32.88 }

```

With the above definitions we are sure that **\MakeUppercase** works properly and **\MakeUppercase{C{\ae}sar}** correctly ‘yields’ ‘CÆSAR’; correspondingly **\MakeUppercase{Heluetia}** correctly yields ‘HELVETIA’.

33 Latin shortcuts

For writing dictionaries or didactic texts (in modern spelling only) we defined a third language attribute, or a third typesetting style, a couple of other active characters are defined: **˘** for marking a vowel with the breve sign, and **=** for marking a vowel with the macro sign. Please take notice that neither the OT1 font encoding,

nor the T1 one for most vowels, contain directly the marked vowels, therefore hyphenation of words containing these “accents” may become problematic; for this reason the above active characters not only introduce the required accent, but also an unbreakable zero skip that in practice does not introduce a discretionary break, but allows breaks in the rest of the word.

It must be remarked that the active characters \sim and $=$ may have other meanings in other contexts. For example the equals sign is used by the graphic extensions for specifying keyword options for handling the graphic elements to be included in the document. At the same time, as mentioned in the previous paragraph, diacritical marking in Latin is used only for typesetting certain kind of documents, such as grammars and dictionaries. It is reasonable that the breve and macron active characters are switched on and off at will, and in particular that they are off by default if the attribute `withprosodicmarks` has not been set.

`\ProsodicMarksOn` We begin by adding to the normal typesetting style the definitions of the new commands `\ProsodicMarksOn` and `\ProsodicMarksOff` that should produce error messages when the third style is not declared:

```
33.1 \addto\extraslatin{\def\ProsodicMarksOn{%
33.2   \GenericError{(latin)\@spaces\@spaces\@spaces\@spaces}%
33.3     {Latin language error: \string\ProsodicMarksOn\space
33.4     is defined by setting the\MessageBreak
33.5     language attribute to ‘withprosodicmarks’\MessageBreak
33.6     If you continue you are likely to encounter\MessageBreak
33.7     fatal errors that I can’t recover}%
33.8     {See the Latin language description in the babel
33.9     documentation for explanation}{\@ehd}}}
33.10 \addto\extraslatin{\let\ProsodicMarksOff\relax}
```

Then we temporarily set the caret and the equals sign to active characters so that they can receive their definitions. But first we store their current category codes to restore them later on.

```
33.11 \@tempcnta=\catcode‘=
33.12 \@tempcntb=\catcode‘^
33.13 \catcode‘= \active
33.14 \catcode‘^ \active
```

Now we can add the necessary declarations to the macros that are being activated when the Latin language and its typesetting styles are declared:

```
33.15 \addto\extraslatin{\languageshorthands{latin}}%
33.16 \addto\extraswithprosodicmarks{\bbl@activate{~}}%
33.17 \addto\extraswithprosodicmarks{\bbl@activate{=}}%
33.18 \addto\noextraswithprosodicmarks{\bbl@deactivate{~}}%
33.19 \addto\noextraswithprosodicmarks{\bbl@deactivate{=}}%
33.20 \addto\extraswithprosodicmarks{\ProsodicMarks}
```

`\ProsodicMarks` Next we define the defining macro for the active characters

```
33.21 \def\ProsodicMarks{%
33.22   \def\ProsodicMarksOn{\catcode‘^ \active\catcode‘= \active}%
33.23   \def\ProsodicMarksOff{\catcode‘^ 7\catcode‘= 12\relax}%
```

Notice that with the above redefinitions of the commands `\ProsodicMarksOn` and `\ProsodicMarksOff`, the operation of the newly defined shortcuts may be switched on and off at will, so that even if a picture has to be inserted in the document by means of the commands and keyword options of the `graphicx` package, it suffices to switch them off before invoking the picture including command, and switched on again afterwards; or, even better, since the picture very likely is being inserted within a `figure` environment, it suffices to switch them off within the environment, being conscious that their deactivation remains local to the environment.

```
33.24 \initiate@active@char{~}%
33.25 \initiate@active@char{=}%
33.26 \declare@shorthand{latin}{^a}{%
```

```

33.27 \textormath{\u{a}\bbl@allowhyphens}{\hat{a}}}%
33.28 \declare@shorthand{latin}{^e}{%
33.29 \textormath{\u{e}\bbl@allowhyphens}{\hat{e}}}%
33.30 \declare@shorthand{latin}{^i}{%
33.31 \textormath{\u{i}\bbl@allowhyphens}{\hat{\imath}}}%
33.32 \declare@shorthand{latin}{^o}{%
33.33 \textormath{\u{o}\bbl@allowhyphens}{\hat{o}}}%
33.34 \declare@shorthand{latin}{^u}{%
33.35 \textormath{\u{u}\bbl@allowhyphens}{\hat{u}}}%
33.36 %
33.37 \declare@shorthand{latin}{=a}{%
33.38 \textormath{\={a}\bbl@allowhyphens}{\bar{a}}}%
33.39 \declare@shorthand{latin}{=e}{%
33.40 \textormath{\={e}\bbl@allowhyphens}{\bar{e}}}%
33.41 \declare@shorthand{latin}{=i}{%
33.42 \textormath{\={i}\bbl@allowhyphens}{\bar{\imath}}}%
33.43 \declare@shorthand{latin}{=o}{%
33.44 \textormath{\={o}\bbl@allowhyphens}{\bar{o}}}%
33.45 \declare@shorthand{latin}{=u}{%
33.46 \textormath{\={u}\bbl@allowhyphens}{\bar{u}}}%
33.47 }

```

Notice that the short hand definitions are given only for lower case letters; it would not be difficult to extend the set of definitions to upper case letters, but it appears of very little use in view of the kind of documents where prosodic marks are supposed to be used. Nevertheless in those rare cases when it's required to set some uppercase letters with their prosodic marks, it is always possible to use the standard L^AT_EX commands such as `\u{I}` for typesetting \hat{I} , or `\={A}` for typesetting \bar{A} .

Finally we restore the caret and equals sign initial default category codes.

```

33.48 \catcode'\= \@tempcnta
33.49 \catcode'\^ \@tempcntb

```

so as to avoid conflicts with other packages or other `babel` options.

`\LatinMarksOn` We define two new commands so as to switch on and off the breve and macron shortcuts.

```

33.50 \addto\extraswithprosodicmarks{\let\LatinMarksOn\ProsodicMarksOn}
33.51 \addto\extraswithprosodicmarks{\let\LatinMarksOff\ProsodicMarksOff}

```

It must be understood that by using the above prosodic marks, line breaking is somewhat impeached; since such prosodic marks are used almost exclusively in dictionaries, grammars, and poems (only in school textbooks), this shouldn't be of any importance for what concerns the quality of typesetting.

34 Etymological hyphenation

In order to deal in a clean way with prefixes and compound words to be divided etymologically, the active character " is given a special definition so as to behave as a discretionary break with hyphenation allowed after it. Most of the code for dealing with the active " is already contained in the core of `babel`, but we are going to use it as a single character shorthand for Latin.

```

34.1 \initiate@active@char{"}%
34.2 \addto\extraslatin{\bbl@activate{"}%
34.3 }

```

A temporary macro is defined so as to take different actions in math mode and text mode: specifically in the former case the macro inserts a double quote as it should in math mode, otherwise another delayed macro comes into action.

```

34.4 \declare@shorthand{latin}{"}{%
34.5 \ifmmode

```

```

34.6     \def\lt@@next{''}%
34.7   \else
34.8     \def\lt@@next{\futurelet\lt@temp\lt@cwm}%
34.9   \fi
34.10  \lt@@next
34.11 }%

```

In text mode the `\lt@next` control sequence is such that upon its execution a temporary variable `\lt@temp` is made equivalent to the next token in the input list without actually removing it. Such temporary token is then tested by the macro `\lt@cwm` and if it is found to be a letter token, then it introduces a compound word separator control sequence `\lt@allowhyphens` whose expansion introduces a discretionary hyphen and an unbreakable space; in case the token is not a letter, the token is tested again to find if it is the character `|`, in which case it is gobbled and a discretionary break is introduced.

```

34.12 \def\lt@allowhyphens{\nobreak\discretionary{-}{-}{\nobreak\hskip\z@skip}
34.13 \newcommand*{\lt@cwm}{\let\lt@n@xt\relax
34.14   \ifcat\noexpand\lt@temp a%
34.15     \let\lt@n@xt\lt@allowhyphens
34.16   \else
34.17     \if\noexpand\lt@temp|string|%
34.18       \def\lt@n@xt{\lt@allowhyphens\@gobble}%
34.19     \fi
34.20   \fi
34.21 \lt@n@xt}%

```

Attention: the category code comparison does not work if the temporary control sequence `\lt@temp` has been let equal to an implicit character, such as `\ae`; therefore this etymological hyphenation facility does not work with medieval Latin spelling when " immediately precedes a ligature. In order to overcome this drawback the shorthand `"|` may be used in such cases; it behaves exactly as `"`, but it does not test the implicit character control sequence. An input such as `super"|{\ae}quitas`³³ gets hyphenated as `su-per-æqui-tas` instead of `su-pe-ræ-qui-tas`.

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

34.22 \ldf@finish{latin}
34.23 </code>

```

³³This word does not exist in “regular” Latin, and it is used just as an example.

35 The Portuguese language

The file `portuges.dtx`³⁴ defines all the language-specific macros for the Portuguese language as well as for the Brazilian version of this language.

For this language the character " is made active. In table 8 an overview is given of its purpose.

"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
""	like "-", but producing no hyphen sign (for words that should break at some sign such as "entrada/salida.")
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).
\-	like the old \-, but allowing hyphenation in the rest of the word.

Table 8: The extra definitions made by `portuges.ldf`

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
35.1 (*code)
35.2 \LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `portuges` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@portuges` to see whether we have to do something here. Since it is possible to load this file with any of the following four options to `babel`: `portuges`, `portuguese`, `brazil` and `brazilian` we also allow that the hyphenation patterns are loaded under any of these four names. We just have to find out which one was used.

```
35.3 \ifx\l@portuges\@undefined
35.4   \ifx\l@portuguese\@undefined
35.5     \ifx\l@brazil\@undefined
35.6       \ifx\l@brazilian\@undefined
35.7         \@nopatterns{Portuguese}
35.8         \adddialect\l@portuges0
35.9       \else
35.10        \let\l@portuges\l@brazilian
35.11      \fi
35.12    \else
35.13      \let\l@portuges\l@brazil
35.14    \fi
35.15  \else
35.16    \let\l@portuges\l@portuguese
35.17  \fi
35.18 \fi
```

By now `\l@portuges` is defined. When the language definition file was loaded under a different name we make sure that the hyphenation patterns can be found.

```
35.19 \expandafter\ifx\csname l@\CurrentOption\endcsname\relax
35.20   \expandafter\let\csname l@\CurrentOption\endcsname\l@portuges
35.21 \fi
```

Now we have to decide whether this language definition file was loaded for Portuguese or Brazilian use. This can be done by checking the contents of `\CurrentOption`. When it doesn’t contain either ‘portuges’ or ‘portuguese’ we make `\bbl@tempb` empty.

³⁴The file described in this section has version number v1.2q and was last revised on 2008/03/18. Contributions were made by Jose Pedro Ramalhete (JRAMALHE@CERNVM or Jose-Pedro_Ramalhete@MACMAIL) and Arnaldo Viegas de Lima arnaldo@VNET.IBM.COM.

```

35.22 \def\bbl@tempa{portuguese}
35.23 \ifx\CurrentOption\bbl@tempa
35.24   \let\bbl@tempb\@empty
35.25 \else
35.26   \def\bbl@tempa{portuges}
35.27   \ifx\CurrentOption\bbl@tempa
35.28     \let\bbl@tempb\@empty
35.29   \else
35.30     \def\bbl@tempb{brazil}
35.31   \fi
35.32 \fi
35.33 \ifx\bbl@tempb\@empty

```

The next step consists of defining commands to switch to (and from) the Portuguese language.

`\captionsportuges` The macro `\captionsportuges` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```

35.34 \@namedef{captions\CurrentOption}{%
35.35   \def\prefacename{Pref\'acio}%
35.36   \def\refname{Refer\~encias}%
35.37   \def\abstractname{Resumo}%
35.38   \def\bibname{Bibliografia}%
35.39   \def\chaptername{Cap\'{\i}tulo}%
35.40   \def\appendixname{Ap\~endice}%

```

Some discussion took place around the correct translations for ‘Table of Contents’ and ‘Index’. the translations differ for Portuguese and Brazilian based the following history:

The whole issue is that some books without a real index at the end misused the term ‘Índice’ as table of contents. Then, what happens is that some books appeared with ‘Índice’ at the beginning and a ‘Índice Remissivo’ at the end. Remissivo is a redundant word in this case, but was introduced to make up the difference. So in Brasil people started using ‘Sumário’ and ‘Índice Remissivo’. In Portugal this seems not to be very common, therefore we chose ‘Índice’ instead of ‘Índice Remissivo’.

```

35.41   \def\contentsname{Conte\'udo}%
35.42   \def\listfigurename{Lista de Figuras}%
35.43   \def\listtablename{Lista de Tabelas}%
35.44   \def\indexname{\’Índice}%
35.45   \def\figurename{Figura}%
35.46   \def\tablename{Tabela}%
35.47   \def\partname{Parte}%
35.48   \def\enclname{Anexo}%
35.49   \def\ccname{Com c\’opia a}%
35.50   \def\headtoname{Para}%
35.51   \def\pagename{P\’agina}%
35.52   \def\seename{ver}%
35.53   \def\alsoname{ver tamb\’em}%

```

An alternate term for ‘Proof’ could be ‘Prova’.

```

35.54   \def\proofname{Demonstra\c{c}\~ao}%
35.55   \def\glossaryname{Gloss\’ario}%
35.56   }

```

`\dateportuges` The macro `\dateportuges` redefines the command `\today` to produce Portuguese dates.

```

35.57 \@namedef{date\CurrentOption}{%
35.58   \def\today{\number\day\space de\space\ifcase\month\or
35.59     Janeiro\or Fevereiro\or Mar\c{c}o\or Abril\or Maio\or Junho\or

```

```

35.60      Julho\or Agosto\or Setembro\or Outubro\or Novembro\or Dezembro%
35.61      \fi
35.62      \space de\space\number\year}}
35.63 \else

```

For the Brazilian version of these definitions we just add a “dialect”.

```

35.64 \expandafter
35.65 \addialect\csname l@CurrentOption\endcsname\l@portuges

```

`\captionsbrazil` The “captions” are different for both versions of the language, so we define the macro `\captionsbrazil` here.

```

35.66 \namedef{captions\CurrentOption}{%
35.67 \def\prefacename{Pref\'acio}%
35.68 \def\refname{Refer\^encias}%
35.69 \def\abstractname{Resumo}%
35.70 \def\bibname{Refer\^encias Bibliogr\'aficas}%
35.71 \def\chaptername{Cap\'{i}tulo}%
35.72 \def\appendixname{Ap\^endice}%
35.73 \def\contentsname{Sum\'ario}%
35.74 \def\listfigurename{Lista de Figuras}%
35.75 \def\listtablename{Lista de Tabelas}%
35.76 \def\indexname{\'Indice Remissivo}%
35.77 \def\figurename{Figura}%
35.78 \def\tablename{Tabela}%
35.79 \def\partname{Parte}%
35.80 \def\enclname{Anexo}%
35.81 \def\ccname{C\'opia para}%
35.82 \def\headtoname{Para}%
35.83 \def\pagename{P\'agina}%
35.84 \def\seename{veja}%
35.85 \def\alsoname{veja tamb\'em}%
35.86 \def\proofname{Demonstra\c{c}\~ao}%
35.87 \def\glossaryname{Gloss\'ario}%
35.88 }

```

`\datebrazil` The macro `\datebrazil` redefines the command `\today` to produce Brazilian dates, for which the names of the months are not capitalized.

```

35.89 \namedef{date\CurrentOption}{%
35.90 \def\today{\number\day\space de\space\ifcase\month\or
35.91 janeiro\or fevereiro\or mar\c{c}o\or abril\or maio\or junho\or
35.92 julho\or agosto\or setembro\or outubro\or novembro\or dezembro%
35.93 \fi
35.94 \space de\space\number\year}}
35.95 \fi

```

`\portugeshyphenmins` Set correct values for `\lefthyphenmin` and `\righthyphenmin`.

```

35.96 \providehyphenmins{CurrentOption}{\tw@\thr@@}

```

`\extrasportuges` The macro `\extrasportuges` will perform all the extra definitions needed for the Portuguese language. The macro `\noextrasportuges` is used to cancel the actions of `\extrasportuges`.

For Portuguese the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the portuguese group of shorthands should be used.

```

35.97 \initiate@active@char{"}
35.98 \namedef{extras\CurrentOption}{\languageshorthands{portuges}}
35.99 \expandafter\addto\csname extras\CurrentOption\endcsname{%
35.100 \bbl@activate{}}

```

Don't forget to turn the shorthands off again.

```

35.101 \addto\noextrasportuges{\bbl@deactivate{}}

```

First we define access to the guillemets for quotations,

```
35.102 \declare@shorthand{portuges}{"<"}{%
35.103   \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
35.104 \declare@shorthand{portuges}{">"}{%
35.105   \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```
35.106 \declare@shorthand{portuges}{"-"}{\nobreak-\bbl@allowhyphens}
35.107 \declare@shorthand{portuges}{""}{\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
35.108 \declare@shorthand{portuges}{"|"}{%
35.109   \textormath{\discretionary{-}{-}{\kern.03em}}{}}
```

\- All that is left now is the redefinition of \-. The new version of \- should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of T_EX in this respect is very unfortunate for languages such as Dutch and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that T_EX can generate from the hyphenation patterns.

```
35.110 \expandafter\addto\csname extras\CurrentOption\endcsname{%
35.111   \babel@save\-\}
35.112 \expandafter\addto\csname extras\CurrentOption\endcsname{%
35.113   \def\-\allowhyphens\discretionary{-}{-}{\allowhyphens}}
```

\ord We also provide an easy way to typeset ordinals, both in the male (\ord or \ro) and the female (orda or \ra) form.

```
\orda35.114 \def\ord{$~{\rm o}$}
\ra35.115 \def\orda{$~{\rm a}$}
35.116 \let\ro\ord\let\ra\orda
```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at \begin{document} and resetting the category code of @ to its original value.

```
35.117 \ldf@finish\CurrentOption
35.118 \code>
```

36 The Spanish language

The file `spanish.dtx`³⁵ defines all the language-specific macros for the Spanish language.

Spanish support is implemented following mainly the guidelines given by José Martínez de Sousa. You may get the full documentation (more comprehensive, but regrettably only in Spanish) by typesetting `spanish.dtx` directly. There are examples and some additional features documented in the Spanish version only. Cross-references in this section point to that document.

Features This style provides:

- Translations following the International L^AT_EX conventions, as well as `\today`.
- Shorthands listed in Table 9. Examples in subsection 3.4 are illustrative. Notice that `"~` has a special meaning in `spanish` different to other languages, and is used mainly in linguistic contexts.

<code>'a</code>	Acute accented a. Works for e, i, o, u, too (both lowercase and uppercase).
<code>'n</code>	ñ (uppercase too).
<code>"i</code>	ï (uppercase too).
<code>"u</code>	ü (uppercase too).
<code>"a "o</code>	Ordinal numbers (uppercase "A, "O too).
<code>"er "ER</code>	Ordinal 1. ^{er} 1. ^{ER}
<code>"c</code>	ç (uppercase too).
<code>"rr</code>	rr, but -r when hyphenated.
<code>"y</code>	An old ligature for “et” (like the English &).
<code>"-</code>	Like <code>\-</code> , but allowing hyphenation in the rest the word.
<code>"=</code>	Like <code>-</code> , but allowing hyphenation in the rest the word.
<code>"~</code>	The hyphen is repeated at the very beginning of the next line if the word is hyphenated at this point.
<code>" "</code>	Like <code>"-</code> but producing no hyphen sign.
<code>~-</code>	Like <code>"-</code> but with no break after the hyphen. Works for en-dashes (<code>~--</code>) and em-dashes (<code>~---</code>). <code>"+</code> , <code>"+-</code> and <code>"+-</code> are synonymous.
<code>"/</code>	A slash slightly lowered, if necessary.
<code>" </code>	Disable ligatures at this point.
<code>"<</code>	Left guillemets.
<code>"></code>	Right guillemets.
<code><< >></code>	<code>\begin{quoting}</code> and <code>\end{quoting}</code> . (See below.) "‘ and "’ are synonymous.
<code>"? "!</code>	Opening question and exclamation marks (¿¡) aligned on the baseline, useful for all-caps headings, etc.

Table 9: Extra definitions made by file `spanish.ldf`

- `\frenchspacing`.
- *In math mode*, a dot followed by a digit is replaced by a decimal comma.
- Spanish ordinals and abbreviations with the `\sptext{<text>}` command as, for instance, `1\sptext{er}`. The preceptive dot is included.

³⁵The file described in this section has version number v5.0h and was last revised on 2009/01/02. The maintainer from v4.0 on is Javier Bezos (<http://www.tex-tipografia.com>). Previous versions were made by Julio Sánchez. The English documentation has been improved by JosÃ© Luis Rivera; thanks to him it is now a lot clearer.

- Accented (lím, máx, mín, mód) and spaced (arc cos, etc.) functions.
- `\dotlessi` is provided for use in math mode.
- A `quoting` environment and a related pair of shorthands `<<` and `>>`. Useful for traditional spanish multi-paragraph quoting.
- There is a small space before the percent `\%` sign.
- `\lsc` provides lowercase small caps. (See subsection 3.10.)
- Ellipsis is best typed as `\dots` or, within a sentence, as `\...`

If `spanish` is the main language, the command `\layoutspanish` is added to the main group, modifying the standard classes throughout the whole document in the following way:

- Paragraphs are set with `\indentfirst`.
- Both `enumerate` and `itemize` are adapted to Spanish rules.
- Both `\alph` and `\Alph` include \tilde{n} after n .
- Symbol footmarks are one, two, three, etc., asterisks.
- OT1 guillemets are generated with two `lasy` symbols instead of small `\l1` and `\gg`.
- `\roman` is redefined to write small caps Roman numerals, since lowercase Roman numerals are discouraged (see below).
- There is a dot after section numbers in titles, headings, and toc.

A subset of these features is implemented for Plain \TeX (accessible with the command `\input spanish.sty`). Most significantly, `\lsc`, the `quoting` environment, and features provided by `\layoutspanish` are missing.

Customization Beginning with version 5.0, customization is made following two paths: via `options` or via `commands`; these options and commands override the layout for Spanish documents at different levels: options are meant for use at the preamble only, while commands may be used in the configuration file or at document level.

Global options control the overall appearance of the document, and may be set on the `{babel}` call, right after calling `spanish`, or shortly before the call to `{babel}`, to ensure their proper loading at runtime. Thus, the following calls are roughly equivalent:

```
\usepackage[... ,spanish,es-nosectiondot,es-nodecimaldot,...]{babel}

\def\spanishoptions{es-nosectiondot,es-nodecimaldot}
\usepackage[... ,spanish,...]{babel}
```

Some global options are built upon lower level options, and may be used as shorthand for more global customizations. Table 10 gives an overview of the global options constructed this way. Most of these options are self-explanatory: they disable the changes made to the basic \LaTeX layout by `spanish`. `es-lcroman` however, and a few others, need a bit of explanation, and they may be described as follows:

Basic Options	es-minimal	es-sloppy	es-noshorthands
<code>es-noindentfirst</code>	X	X	
<code>es-nosectiondot</code>	X	X	
<code>es-nolists</code>	X	X	
<code>es-noquoting</code>	X	X	X
<code>es-notilde</code>	X	X	X
<code>es-nodecimaldot</code>	X	X	X
<code>es-nolayout</code>		X	
<code>es-ucroman</code>	X		
<code>es-lcroman</code>	X	X	

Table 10: Spanish Customization Options

- Traditional Spanish typography discourages the use of lowercase Roman numerals; instead, a smallcaps variant is implemented. However, since `Makeindex` seems to choke on the code implementing lowercase Roman numerals (via the `\lsc` macro), two workarounds are implemented: the `es-ucroman` option converts all Roman numerals to uppercase, and the `es-lcroman` option turns all Roman numerals to lowercase; the former should be preferred over the latter. Three macros control local changes to Roman numbers: `\spanishscroman`, `\spanishucroman`, and `\spanishlcroman`.
- The `es-preindex` option calls the `romanidx.sty` package automatically to fix index entries in smallcaps roman form. An additional macro, `\spanishindexchars{⟨encap⟩}{⟨openrange⟩}{⟨closerange⟩}` determines the characters delimiting index entries. Defaults are `\spanishindexchars{|}{(}{)}`.
- The `es-tilden` option restores the old tilde `~` shorthand for `ñ`. This shorthand is however *strongly* deprecated.
- The `es-nolayout` option disables layout changes in the document when `spanish` is the main language. These changes affect enumerated and itemized lists, enumerations (alphabetic order excludes `ñ`), and symbolic footnotes.
- The `es-noshorthands` disables the shorthand mechanism completely: neither `"` nor `'` nor `<` nor `>` nor `~` nor `.` work at all.
- The `es-noquoting` option disables the macros `<<` and `>>` calling the `quoting` environment; the alternative macros `"'` and `"'` are still available.
- The `es-uppernames` option makes uppercase versions of captions for chapter, tablename, etc.
- The `es-tabla` option changes “cuadro” for “tabla” in captions.

Finally, the Spanish 5 series begins the implementation of national variations of Spanish typography, beginning with Mexico. Thus the global options `mexico` and `mexico-com` are adapted to practices spread in Mexico, and perhaps Central America, the Caribbean, and some countries in South America.³⁶

Many of the global options are implemented via macros, which may be included in the configuration file `spanish.cfg`, in the preamble, after the call to `babel`, and in the body of the document. These macros are the following.

- The macros `\spanishdashitems` and `\spanishsignitems` change the values of itemized lists to a series of dashes or an alternative series of symbols, respectively.

³⁶The main difference is that `mexico` disables the `decimaldot` mechanism, while `mexico-com` keeps it enabled; both change the `quoting` environment, disabling the use of guillemets.

<code>\lquote</code>	"<
<code>\rquote</code>	">
<code>\lquotii</code>	“
<code>\rquotii</code>	”
<code>\lquotiii</code>	‘
<code>\rquotiii</code>	’

Table 11: Default quoting signs set for the `quoting` environment.

- The command `\deactivatequoting` deactivates the `<<` and `>>` shorthands if you want to use `<` and `>` in numerical comparisons and some `AMSTeX` commands.
- You may kill the space in spaced operators with `\unspacedoperators`.
- You may kill the accents on accented operators with `\unaccentedoperators`.
- The command `\decimalpoint` resets the decimal separator to its default (dot) value, while `\spanishdecimal{⟨symbol⟩}` allows for an arbitrary definition.
- `\spanishplainpercent` prevents the addition of a thinspace before the percent sign in texts. This might be useful for parenthesized percent signs in tables, etc.
- The macros `\spanishdatedel` and `\spanishdatede` control the if the article is given in years (`del` or `de`).
- The macro `\spanishreverseddate` sets the date of the format “Month Day del Year”.
- The macro `\Today` gives months in uppercase.
- The macros `\spanishcaption` change the value of the *caption* automatically (no need to add an `\addto`).
- The command `\spanishdeactivate{⟨characters⟩}` disables the shorthand characters listed in the argument. Eligible characters are the set `.'~"<>`. These shorthand characters may be globally deactivated for Spanish adding this command to `\shorthandsspanish`.
- Extras are divided in groups controlled by the commands `\textspanish`, `\mathspanish`, `\shorthandsspanish` y `\layoutspanish`; their values may be cancelled typing `\renewcommand{⟨command⟩}{}`, or changed at will (check the Spanish documentation or the code for details).
- The command `\spanishoperators{⟨operators⟩}` defines command names for operators in Spanish. There is no standard name for some of them, so they may be created or changed at will. For instance, the command `\renewcommand{\spanishoperators}{arc\,ctg m\acute{i}n}` creates commands for these functions. The command `\,` adds thinspaces at the appropriate places for spaced operators (like `\arcctg` in this case), and the command `\acute{⟨letter⟩}` adds an accent to the letter included in the definition (thus, `m\acute{i}n` defines the accented function `\min` (mín); please notice that `\dotlessi` is not necessary).
- The commands `\lquote{⟨string⟩}` `\rquote{⟨string⟩}` `\lquotii{⟨string⟩}` `\rquotii{⟨string⟩}` `\lquotiii{⟨string⟩}` `\rquotiii{⟨string⟩}` set the quoting signs in the `quoting` environment, nested from outside in. They may be `\renewed` at will. Default values are shown in table 11.

- The command `\selectspanish*` is obsolete: if `spanish` is the main language, all its features are available right after loading `babel`. The `es-delayed` option is provided to restore the previous behavior and macros for backwards compatibility.

36.1 The Code

This file provides definition for both L^AT_EX 2_ε and non L^AT_EX 2_ε formats.

```

36.1 \*code>
36.2 \ProvidesLanguage{spanish.1df}
36.3 [2009/01/02 v5.0h Spanish support from the babel system]
36.4 \LdfInit{spanish}\captionsspanish
36.5
36.6 \edef\es@savedcatcodes{%
36.7 \catcode'\noexpand\~=\the\catcode'\~
36.8 \catcode'\noexpand\"=\the\catcode'\"}
36.9 \catcode'\~=\active
36.10 \catcode\"=12
36.11
36.12 \ifx\undefined\l@spanish
36.13 \nopatterns{Spanish}
36.14 \adddialect\l@spanish0
36.15 \fi
36.16
36.17 \def\es@sdef#1{\babel@save#1\def#1}
36.18
36.19 \@ifundefined{documentclass}
36.20 {\let\ifes@latex\iffalse}
36.21 {\let\ifes@latex\iftrue}

Package options for spanish. To avoid error messages dummy options are
created on the fly when necessary.

36.22 \ifes@latex
36.23
36.24 \@ifundefined{spanishoptions}{}
36.25 {\PassOptionsToPackage{\spanishoptions}{babel}}
36.26
36.27 \def\es@genoption#1#2#3{%
36.28 \DeclareOption{#1}{}%
36.29 \@ifpackagewith{babel}{#1}%
36.30 {\def\es@a{#1}%
36.31 \expandafter\let\expandafter\es@b\csname opt@babel.sty\endcsname
36.32 \addto\es@b{, #2}%
36.33 \expandafter\let\csname opt@babel.sty\endcsname\es@b
36.34 \AtEndOfPackage{#3}}%
36.35 {}}
36.36
36.37 \es@genoption{es-minimal}
36.38 {es-ucroman,es-noindentfirst,es-nosectiondot,es-noenumerate,%
36.39 es-noitemize,es-noquoting,es-notilde,es-nodecimaldot}
36.40 {\spanishplainpercent
36.41 \let\es@operators\relax}
36.42 \es@genoption{es-nolists}
36.43 {es-noenumerate,es-noitemize}{}
36.44 \es@genoption{es-sloppy}
36.45 {es-nolayout,es-noshorthands}{}
36.46 \es@genoption{es-noshorthands}
36.47 {es-noquoting,es-nodecimaldot,es-notilde}{}
36.48 \es@genoption{mexico}
36.49 {mexico-com,es-nodecimaldot}{}
36.50 \es@genoption{mexico-com}
36.51 {es-tabla,es-noquoting}

```

```

36.52 {\def\lquotei{'}\def\rquotei{'}}%
36.53 \def\lquoteii{'}\def\rquoteii{'}}%
36.54 \def\lquoteiii{\guillemotleft{}}%
36.55 \def\rquoteiii{\guillemotright{}}}%
36.56
36.57 \def\es@ifoption#1#2#3{%
36.58 \DeclareOption{es-#1}{}%
36.59 \@ifpackagewith{babel}{es-#1}{#2}{#3}}%
36.60
36.61 \def\es@optlayout#1#2{\es@ifoption{#1}{}{\addto\layoutspanish{#2}}}%
36.62
36.63 \else
36.64
36.65 \def\es@ifoption#1#2#3{\@namedef{spanish#1}{#2}}%
36.66
36.67 \fi
36.68
36.69 \let\es@uclc\@secondoftwo
36.70 \es@ifoption{uppernames}{\let\es@uclc\@firstoftwo}{}%
36.71
36.72 \def\es@tablename{Ccuadro}
36.73 \es@ifoption{tabla}{\def\es@tablename{Ttabla}}{}%
36.74 \es@ifoption{cuadro}{\def\es@tablename{Ccuadro}}{}%

```

Captions follow a two step schema, so that, say, `\refname` is defined as `\spanishrefname` which in turn contains the string to be printed. The final definition of `\captionsspanish` is built below.

```

36.75 \def\captionsspanish{%
36.76 \es@a{preface}{Prefacio}%
36.77 \es@a{ref}{Referencias}%
36.78 \es@a{abstract}{Resumen}%
36.79 \es@a{bib}{Bibliograf'\{i\}a}%
36.80 \es@a{chapter}{Cap'\{i\}tulo}%
36.81 \es@a{appendix}{Ap'\{e\}ndice}%
36.82 \es@a{listfigure}{\{I\}ndice de \es@uclc Ffiguras}%
36.83 \es@a{listtable}{\{I\}ndice de \expandafter\es@uclc\es@tablename s}%
36.84 \es@a{index}{\{I\}ndice \es@uclc Aalfab'\{e\}tico}%
36.85 \es@a{figure}{Figura}%
36.86 \es@a{table}{\expandafter\@firstoftwo\es@tablename}%
36.87 \es@a{part}{Parte}%
36.88 \es@a{encl}{Adjunto}%
36.89 \es@a{cc}{Copia a}%
36.90 \es@a{headto}{A}%
36.91 \es@a{page}{p'\{a\}gina}%
36.92 \es@a{see}{v'\{e\}ase}%
36.93 \es@a{also}{v'\{e\}ase tambi'\{e\}n}%
36.94 \es@a{proof}{Demostraci'\{o\}n}%
36.95 \es@a{glossary}{Glosario}%
36.96 \@ifundefined{chapter}
36.97 {\es@a{contents}{\{I\}ndice}}%
36.98 {\es@a{contents}{\{I\}ndice \es@uclc Ggeneral}}}%
36.99
36.100 \def\es@a#1{\@namedef{spanish#1name}}%
36.101 \captionsspanish
36.102 \def\es@a#1#2{%
36.103 \expandafter\noexpand\csname#1name\endcsname
36.104 {\expandafter\noexpand\csname spanish#1name\endcsname}}%
36.105 \edef\captionsspanish{\captionsspanish}

```

Now two macros for dates (upper and lowercase).

```

36.106 \def\es@month#1{%
36.107 \expandafter#1\ifcase\month\or Eenero\or Ffebrero\or
36.108 Mmarzo\or Aabril\or Mmayo\or Jjunio\or Jjulio\or Aagosto\or

```

```

36.109 Sseptiembre\or Ooctubre\or Nnoviembre\or Ddiciembre\fi}
36.110
36.111 \def\es@today#1{%
36.112 \ifcase\es@datefmt
36.113 \the\day~de \es@month#1%
36.114 \else
36.115 \es@month#1~\the\day
36.116 \fi
36.117 \ de\ifnum\year>1999\es@year1\fi~\the\year}
36.118
36.119 \def\datespanish{%
36.120 \def\today{\es@today\@secondoftwo}%
36.121 \def\Today{\es@today\@firstoftwo}}
36.122 \newcount\es@datefmt
36.123 \def\spanishreverseddate{\es@datefmt\@one}
36.124 \def\spanishdatedel{\def\es@year1{1}}
36.125 \def\spanishdatede{\let\es@year1\@empty}
36.126 \spanishdatede

```

The basic macros to select the language in the preamble or the config file. Use of `\selectlanguage` should be avoided at this early stage because the active chars are not yet active. `\selectspanish` makes them active.

```

36.127 \def\selectspanish{%
36.128 \def\selectspanish{%
36.129 \def\selectspanish{%
36.130 \PackageWarning{spanish}{Extra \string\selectspanish ignored}}%
36.131 \es@select}}
36.132 \@onlypreamble\selectspanish
36.133 \def\es@select{%
36.134 \let\es@select\@undefined
36.135 \selectlanguage{spanish}}
36.136
36.137 \let\es@shlist\@empty

```

Instead of joining all the extras directly in `\extrasspanish`, we subdivide them in three further groups.

```

36.138 \def\extrasspanish{%
36.139 \textspanish
36.140 \mathspanish
36.141 \ifx\shorthandsspanish\@empty
36.142 \expandafter\spanishdeactivate\expandafter{\es@shlist}%
36.143 \languageshorthands{none}%
36.144 \else
36.145 \shorthandsspanish
36.146 \fi}
36.147 \def\noextrasspanish{%
36.148 \ifx\textspanish\@empty\else
36.149 \notextspanish
36.150 \fi
36.151 \ifx\mathspanish\@empty\else
36.152 \nomathspanish
36.153 \fi
36.154 \ifx\shorthandsspanish\@empty\else
36.155 \noshorthandsspanish
36.156 \fi
36.157 \csname es@restorelist\endcsname}
36.158
36.159 \addto\textspanish{\es@sdef\sptext{\protect\es@sptext}}
36.160
36.161 \def\es@orddot{.}

```

The definition of `\sptext` is more elaborated than that of `\textsuperscript`. With uppercase superscript text the `scriptscriptsize` is used. The mandatory dot

is already included. There are two versions, depending on the format.

```

36.162 \ifes@latex
36.163   \def\es@sptext#1{%
36.164     {\es@orddot
36.165       \setbox\z@\hbox{8}\dimen@ht\z@
36.166       \csname S@f@size\endcsname
36.167       \edef\@tempa{\def\noexpand\@tempc{#1}%
36.168         \lowercase{\def\noexpand\@tempb{#1}}}\@tempa
36.169       \ifx\@tempb\@tempc
36.170         \fontsize\sf@size\z@
36.171         \selectfont
36.172         \advance\dimen@-1.15ex
36.173       \else
36.174         \fontsize\ssf@size\z@
36.175         \selectfont
36.176         \advance\dimen@-1.5ex
36.177       \fi
36.178       \math@fontsfalse\raise\dimen@\hbox{#1}}
36.179 \else
36.180   \let\sptextfont\rm
36.181   \def\es@sptext#1{%
36.182     {\es@orddot
36.183       \setbox\z@\hbox{8}\dimen@ht\z@
36.184       \edef\@tempa{\def\noexpand\@tempc{#1}%
36.185         \lowercase{\def\noexpand\@tempb{#1}}}\@tempa
36.186       \ifx\@tempb\@tempc
36.187         \advance\dimen@-0.75ex
36.188         \raise\dimen@\hbox{$\scriptstyle\sptextfont#1$}%
36.189       \else
36.190         \advance\dimen@-0.8ex
36.191         \raise\dimen@\hbox{$\scriptscriptstyle\sptextfont#1$}%
36.192       \fi}}
36.193 \fi

```

Now, lowercase small caps. First, we test if there are actual small caps for the current font. If not, faked small caps are used. The `\selectfont` in `\es@lsc` could seem redundant, but it's not. An intermediate macro allows using an optimized variant for Roman numerals.

```

36.194 \ifes@latex
36.195   \addto\textspanish{\es@sdef\lsc{\protect\es@lsc}}
36.196   \def\es@lsc{\es@xlsc\MakeUppercase\MakeLowercase}
36.197   \def\es@xlsc#1#2#3{%
36.198     \leavevmode
36.199     \hbox{%
36.200       \scshape\selectfont
36.201       \expandafter\ifx\csname\fontencoding/\fontfamily/\fontseries
36.202         /n/\fontsize\expandafter\endcsname
36.203       \csname\curr@fontshape/\fontsize\endcsname
36.204       \csname S@f@size\endcsname
36.205       \fontsize\sf@size\z@\selectfont
36.206       \PackageWarning{spanish}{Replacing '\curr@fontshape' by
36.207         \MessageBreak faked small caps}%
36.208       #1{#3}%
36.209     \else
36.210       #2{#3}%
36.211     \fi}}
36.212 \fi

```

The quoting environment. This part is not available in Plain. Overriding the default `\everypar` is a bit tricky.

```

36.213 \newif\ifes@listquot
36.214

```

```

36.215 \ifes@latex
36.216 \csname newtoks\endcsname\es@quottoks
36.217 \csname newcount\endcsname\es@quotdepth
36.218 \newenvironment{quoting}
36.219 {\leavevmode
36.220 \advance\es@quotdepth\@ne
36.221 \csname lquot\romannumeral\es@quotdepth\endcsname%
36.222 \ifnum\es@quotdepth=\@ne
36.223 \es@listquotfalse
36.224 \let\es@quotpar\everypar
36.225 \let\everypar\es@quottoks
36.226 \everypar\expandafter{\the\es@quotpar}%
36.227 \es@quotpar{\the\everypar
36.228 \ifes@listquot\global\es@listquotfalse\else\es@quotcont\fi}%
36.229 \fi
36.230 \toks@\expandafter{\es@quotcont}%
36.231 \edef\es@quotcont{\the\toks@
36.232 \expandafter\noexpand
36.233 \csname rquot\romannumeral\es@quotdepth\endcsname}}
36.234 {\csname rquot\romannumeral\es@quotdepth\endcsname}
36.235 \def\lquoti{\guillemotleft{}}
36.236 \def\rquoti{\guillemotright{}}
36.237 \def\lquotii{'{'}
36.238 \def\rquotii{'}}
36.239 \def\lquotiii{'{'}
36.240 \def\rquotiii{'}}
36.241 \let\es@quotcont\@empty

```

If there is a margin par inside quoting, we don't add the quotes. `\es@listquot` stores the quotes to be used before item labels; otherwise they could appear after the labels.

```

36.242 \addto\@marginparreset{\let\es@quotcont\@empty}
36.243 \DeclareRobustCommand\es@listquot{%
36.244 \csname rquot\romannumeral\es@quotdepth\endcsname
36.245 \global\es@listquottrue}
36.246 \fi

```

Now, the `\frenchspacing`, followed by `\...` and `\%`.

```

36.247 \addto\textspanish{\bbl@frenchspacing}
36.248 \addto\notextspanish{\bbl@nonfrenchspacing}
36.249 \addto\textspanish{%
36.250 \let\es@save@dot\.%
36.251 \es@sdef\.{\ifnextchar.{\es@dots}{\es@save@dot}}}
36.252 \def\es@dots..\{\leavevmode\hbox{...}\spacefactor\@M}
36.253 \def\es@sppercent{\unskip\textormath{\$m@th\,$}\{,\}}
36.254 \def\spanishplainpercent{\let\es@sppercent\@empty}
36.255 \addto\textspanish{%
36.256 \let\percentsign\%%
36.257 \es@sdef\%{\es@sppercent\percentsign{}}}

```

We follow with the math group. It's not easy to add an accent to an operator. The difficulty is that we must avoid using text (that is, `\mbox`) because we have no control on font and size, and at time we should access `\i`, which is a text command forbidden in math mode. `\dotlessi` must be converted to uppercase if necessary in $\text{\LaTeX 2}_{\varepsilon}$. There are two versions, depending on the format.

```

36.258 \addto\mathspanish{\es@sdef\dotlessi{\protect\es@dotlessi}}
36.259 \let\nomathspanish\relax
36.260
36.261 \ifes@latex
36.262 \def\es@texti{\i}
36.263 \addto\@uclclist{\dotlessi\es@texti}
36.264 \fi

```

```

36.265
36.266 \ifes@latex
36.267 \def\es@dotlessi{%
36.268 \ifmmode
36.269 {\ifnum\mathgroup=\m@ne
36.270 \imath
36.271 \else
36.272 \count@\escapechar \escapechar=\m@ne
36.273 \expandafter\expandafter\expandafter
36.274 \split@name\expandafter\string\the\textfont\mathgroup\@nil
36.275 \escapechar=\count@
36.276 \@ifundefined{f@encoding\string\i}%
36.277 {\edef\f@encoding{\string?}}{}%
36.278 \expandafter\count@\the\csname\f@encoding\string\i\endcsname
36.279 \advance\count@"7000
36.280 \mathchar\count@
36.281 \fi}%
36.282 \else
36.283 \i
36.284 \fi}
36.285 \else
36.286 \def\es@dotlessi{\textormath\i}{\mathchar"7010}}
36.287 \fi
36.288
36.289 \def\accentedoperators{%
36.290 \def\es@op@ac##1{\acute{\if i##1\dotlessi\else##1\fi}}}
36.291 \def\unaccentedoperators{%
36.292 \def\es@op@ac##1{##1}}
36.293 \accentedoperators
36.294 \def\spacedoperators{\let\es@op@sp\,}
36.295 \def\unspacedoperators{\let\es@op@sp@empty}
36.296 \spacedoperators
36.297 \addto\mathspanish{\es@operators}
36.298
36.299 \ifes@latex\else
36.300 \let\operator@font\rm
36.301 \fi

```

The operators are stored in `\es@operators`, which in turn is included in the math group. Since `\operator@font` is defined in $\text{\LaTeX 2}_{\epsilon}$ only, we defined in the plain variant.

```

36.302 \def\es@operators{%
36.303 \es@sdef\bmod{\nonscript\mskip-\medmuskip\mkern5mu
36.304 \mathbin{\operator@font m\es@op@ac od}\penalty900\mkern5mu
36.305 \nonscript\mskip-\medmuskip}%
36.306 \@ifundefined{amsmath@err}%
36.307 {\es@sdef\pmod##11{\allowbreak\mkern18mu
36.308 ({\operator@font m\es@op@ac od}\,,\,,##11)}}%
36.309 {\es@sdef\mod##11{\allowbreak\if@display\mkern18mu
36.310 \else\mkern12mu\fi{\operator@font m\es@op@ac od}\,,\,,##11}%
36.311 \es@sdef\pmod##11{\pod{{\operator@font m\es@op@ac od}%
36.312 \mkern6mu##11}}}%
36.313 \def\es@a##1 {%
36.314 \if^##1~% empty? continue
36.315 \bbl@afterelse
36.316 \es@a
36.317 \else
36.318 \bbl@afterfi
36.319 {\if&##1% &? finish
36.320 \else
36.321 \bbl@afterfi
36.322 \begingroup
36.323 \let\,\@empty % ignore when def'ing name

```

```

36.324 \let\acute\@firstofone % id
36.325 \edef\es@b{\expandafter\noexpand\csname##1\endcsname}%
36.326 \def\,{\noexpand\es@op@sp}%
36.327 \def\acute{\noexpand\es@op@ac}%
36.328 \edef\es@a{\endgroup
36.329 \noexpand\es@sdef\expandafter\noexpand\es@b{%
36.330 \mathop{\noexpand\operator@font##1}\es@c}}%
36.331 \es@a % restores itself
36.332 \es@a
36.333 \fi}%
36.334 \fi}%
36.335 \let\es@b\spanishoperators
36.336 \addto\es@b{ }%
36.337 \let\es@c\@empty
36.338 \expandafter\es@a\es@b l\acute{i}m l\acute{i}m\,sup
36.339 l\acute{i}m\,inf m\acute{a}x \acute{i}nf m\acute{i}n & %
36.340 \def\es@c{\nolimits}%
36.341 \expandafter\es@a\es@b sen tg arc\,sen arc\,cos arc\,tg & }
36.342 \def\spanishoperators{cotg cosec senh tgh }

```

Now comes the text shorthands. They are grouped in `\shorthandsspanish` and this style performs some operations before the babel shorthands are called. The aims are to allow expression like $a^{\{x\}}$ and to deactivate shorthands by making them of category ‘other.’ After providing a `\i` shorthand, the new macros are defined.

```

36.343 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'\i}
36.344
36.345 \def\es@set@shorthand#1{%
36.346 \expandafter\edef\csname es@savecat\string#1\endcsname
36.347 {\the\catcode'#1}%
36.348 \initiate@active@char{#1}%
36.349 \catcode'#1=\csname es@savecat\string#1\endcsname\relax
36.350 \if.#1\else
36.351 \addto\es@restorelist{\es@restore{#1}}%
36.352 \addto\es@select{\shorthandon{#1}}%
36.353 \addto\shorthandsspanish{\es@activate{#1}}%
36.354 \addto\es@shlist{#1}%
36.355 \fi}
36.356
36.357 \def\es@use@shorthand{%
36.358 \ifx\thepage\relax
36.359 \bbl@afterelse
36.360 \string
36.361 \else
36.362 \bbl@afterfi
36.363 {\ifx\protect\@unexpandable@protect
36.364 \bbl@afterelse
36.365 \noexpand
36.366 \else
36.367 \bbl@afterfi
36.368 \es@use@sh
36.369 \fi}%
36.370 \fi}
36.371
36.372 \def\es@use@sh#1{%
36.373 \if@safe@actives
36.374 \bbl@afterelse
36.375 \string#1%
36.376 \else%
36.377 \bbl@afterfi
36.378 \textormath
36.379 {\csname active@char\string#1\endcsname}%

```

```

36.380   {\csname normal@char\string#1\endcsname}%
36.381 \fi}
36.382
36.383 \gdef\es@activate#1{%
36.384 \begingroup
36.385 \lccode'\~='#1
36.386 \lowercase{%
36.387 \endgroup
36.388 \def~{\es@use@shorthand~}}
36.389
36.390 \def\spanishdeactivate#1{%
36.391 \@tfor\@tempa:=#1\do{\expandafter\es@spdeactivate\@tempa}}
36.392
36.393 \def\es@spdeactivate#1{%
36.394 \if.#1%
36.395 \mathcode'\.\=\es@period@code
36.396 \else
36.397 \begingroup
36.398 \lccode'\~='#1
36.399 \lowercase{%
36.400 \endgroup
36.401 \expandafter\let\expandafter~%
36.402 \csname normal@char\string#1\endcsname}%
36.403 \catcode'#1=\csname es@savecat\string#1\endcsname\relax
36.404 \fi}

```

`\es@restore` is used in the list `\es@restorelist`, which in turn restores all shorthands as defined by `babel`. The latter macros also has `\es@quoting`.

```

36.405 \def\es@restore#1{%
36.406 \shorthandon{#1}%
36.407 \begingroup
36.408 \lccode'\~='#1
36.409 \lowercase{%
36.410 \endgroup
36.411 \bbl@deactivate{~}}

```

To selectively define the shorthands we have a couple of macros, which defines a certain combination if the first character has been activated as a shorthand. The second one is intended for a few shorthands with an alternative form.

```

36.412 \def\es@declare#1{%
36.413 \@ifundefined{es@savecat\expandafter\string\@firstoftwo#1}%
36.414 {\@gobble}%
36.415 {\declare@shorthand{spanish}{#1}}}
36.416 \def\es@declarealt#1#2#3{%
36.417 \es@declare{#1}{#3}%
36.418 \es@declare{#2}{#3}}
36.419
36.420 \if@latex\else
36.421 \def\@tabacckludge#1{\csname\string#1\endcsname}
36.422 \fi
36.423
36.424 \@ifundefined{add@accent}{\def\add@accent#1#2{\accent#1 #2}}{}

```

Instead of redefining `\'`, we redefine the internal macro for the OT1 encoding.

```

36.425 \if@latex
36.426 \def\es@accent#1#2#3{%
36.427 \expandafter\@text@composite
36.428 \csname OT1\string#1\endcsname#3\@empty\@text@composite
36.429 {\bbl@allowhyphens\add@accent{#2}{#3}\bbl@allowhyphens
36.430 \setbox\@tempboxa\hbox{#3%
36.431 \global\mathchardef\accent@spacefactor\spacefactor}%
36.432 \spacefactor\accent@spacefactor}}
36.433 \else

```



```

36.434 \def\es@accent#1#2#3{%
36.435 \bbl@allowhyphens\add@accent{#2}{#3}\bbl@allowhyphens
36.436 \spacefactor\sfcode'#3 }
36.437 \fi
36.438
36.439 \addto\shorthandsspanish{\languageshorthands{spanish}}%
36.440 \es@ifoption{noshorthands}{-}{\es@set@shorthand{}}

We override the default " of babel, intended for german.

36.441 \def\es@umlaut#1{%
36.442 \bbl@allowhyphens\add@accent{127}#1\bbl@allowhyphens
36.443 \spacefactor\sfcode'#1 }
36.444
36.445 \addto\shorthandsspanish{%
36.446 \babel@save\bbl@umlauta
36.447 \let\bbl@umlauta\es@umlaut}
36.448 \let\noshorthandsspanish\relax
36.449
36.450 \ifes@latex
36.451 \addto\shorthandsspanish{%
36.452 \expandafter\es@sdef\csname OT1\string\~\endcsname{\es@accent\~{126}}%
36.453 \expandafter\es@sdef\csname OT1\string\' \endcsname{\es@accent\'{19}}}
36.454 \else
36.455 \addto\shorthandsspanish{%
36.456 \es@sdef\~{\es@accent\~{126}}%
36.457 \es@sdef\'#1{\if#1i\es@accent\'{19}\i\else\es@accent\'{19}{#1}\fi}}
36.458 \fi
36.459
36.460 \def\es@sptext@r#1#2{\es@sptext{#1#2}}
36.461 \es@declare{"a}{\sptext{a}}
36.462 \es@declare{"A}{\sptext{A}}
36.463 \es@declare{"o}{\sptext{o}}
36.464 \es@declare{"O}{\sptext{O}}
36.465 \es@declare{"e}{\protect\es@sptext@r{e}}
36.466 \es@declare{"E}{\protect\es@sptext@r{E}}
36.467 \es@declare{"u}{\sptext{u}}
36.468 \es@declare{"U}{\sptext{U}}
36.469 \es@declare{"i}{\sptext{i}}
36.470 \es@declare{"I}{\sptext{I}}
36.471 \es@declare{"c}{\c{c}}
36.472 \es@declare{"C}{\c{C}}
36.473 \es@declare{"<}{\guillemotleft}
36.474 \es@declare{">}{\guillemotright}
36.475 \def\es@chf{\char\hyphenchar\font}
36.476 \es@declare{"-}{\bbl@allowhyphens\~\bbl@allowhyphens}
36.477 \es@declare{"="}{\bbl@allowhyphens\es@chf\hskip\z@skip}
36.478 \es@declare{"~}{\bbl@allowhyphens}
36.479 {\bbl@allowhyphens
36.480 \discretionary{\es@chf}{\es@chf}{\es@chf}%
36.481 \bbl@allowhyphens}
36.482 \es@declare{"r}{\bbl@allowhyphens
36.483 {\bbl@allowhyphens
36.484 \discretionary{\es@chf}{\es@chf}{r}%
36.485 \bbl@allowhyphens}
36.486 \es@declare{"R}{\bbl@allowhyphens
36.487 {\bbl@allowhyphens
36.488 \discretionary{\es@chf}{\es@chf}{R}%
36.489 \bbl@allowhyphens}
36.490 \es@declare{"y}{\@ifundefined{scalebox}%
36.491 {\ensuremath{\tau}}%
36.492 {\raisebox{1ex}{\scalebox{-1}{\resizebox{.45em}{1ex}{2}}}}
36.493 \es@declare{""}{\hskip\z@skip}

```

```

36.495 \es@declare{"/}%
36.496 {\setbox\z@\hbox{/}%
36.497 \dimen@\ht\z@
36.498 \advance\dimen@-1ex
36.499 \advance\dimen@\dp\z@
36.500 \dimen@.31\dimen@
36.501 \advance\dimen@-\dp\z@
36.502 \ifdim\dimen@>0pt
36.503 \kern.01em\lower\dimen@\box\z@\kern.03em
36.504 \else
36.505 \box\z@
36.506 \fi}
36.507 \es@declare{"?}%
36.508 {\setbox\z@\hbox{"?'}%
36.509 \leavevmode\raise\dp\z@\box\z@}
36.510 \es@declare{"!}%
36.511 {\setbox\z@\hbox{"!' '%
36.512 \leavevmode\raise\dp\z@\box\z@}
36.513
36.514 \def\spanishdecimal#1{\def\es@decimal{#1}}
36.515 \def\decimalcomma{\spanishdecimal{,}}
36.516 \def\decimalpoint{\spanishdecimal{.}}
36.517 \decimalcomma
36.518 \es@ifoption{nodecimaldot}{%
36.519 {\AtBeginDocument{\bgroup\@fileswfalse}%
36.520 \es@set@shorthand{.}%
36.521 \AtBeginDocument{\egroup}%
36.522 \@namedef{normal@char\string.}{%
36.523 \@ifnextchar\egroup
36.524 {\mathchar\es@period@code\relax}%
36.525 {\csname active@char\string.\endcsname}}%
36.526 \declare@shorthand{system}{.}{\mathchar\es@period@code\relax}%
36.527 \addto\shorthandsspanish{%
36.528 \mathchardef\es@period@code\the\mathcode'\.%
36.529 \babel@savevariable{\mathcode'\.%}
36.530 \mathcode'\.= "8000 %
36.531 \es@activate{.}}%
36.532 \def\es@a#1{\es@declare{.#1}{\es@decimal#1}}%
36.533 \es@a1\es@a2\es@a3\es@a4\es@a5\es@a6\es@a7\es@a8\es@a9\es@a0}
36.534
36.535 \es@ifoption{notilde}{%}{\es@set@shorthand{~}}
36.536 \def\deactivatetilden{%
36.537 \expandafter\let\csname spanish@sh@\string~@N\endcsname\relax
36.538 \expandafter\let\csname spanish@sh@\string~@N\endcsname\relax}
36.539 \es@ifoption{tilden}
36.540 {\es@declare{~n}{\~n}%
36.541 \es@declare{~N}{\~N}}
36.542 {\let\deactivatetilden\relax}
36.543 \es@declarealt{~-}{~+}%
36.544 \leavevmode
36.545 \bgroup
36.546 \let\@sptoken\es@dashes % Changes \@ifnextchar behaviour
36.547 \@ifnextchar-%
36.548 {\es@dashes}%
36.549 {\hbox{\es@chf}\egroup}}
36.550 \def\es@dashes-{%
36.551 \@ifnextchar-%
36.552 {\bbl@allowhyphens\hbox{---}\bbl@allowhyphens\egroup\@gobble}%
36.553 {\bbl@allowhyphens\hbox{--}\bbl@allowhyphens\egroup}}
36.554
36.555 \es@ifoption{noquoting}%
36.556 {\let\es@quoting\relax

```

```

36.557 \let\activatequoting\relax
36.558 \let\deactivatequoting\relax}
36.559 {\@ifundefined{XML@catcodes}%
36.560 {\es@set@shorthand{<}%
36.561 \es@set@shorthand{>%
36.562 \declare@shorthand{system}{<}{\csname normal@char\string<\endcsname}%
36.563 \declare@shorthand{system}{>}{\csname normal@char\string>\endcsname}%
36.564 \addto\es@restorelist{\es@quoting}%
36.565 \addto\es@select{\es@quoting}%
36.566 \if@latex
36.567 \AtBeginDocument{%
36.568 \es@quoting
36.569 \if@filesw
36.570 \immediate\write\@mainaux{\string\@nameuse{es@quoting}}%
36.571 \fi}%
36.572 \fi
36.573 \def\activatequoting{%
36.574 \shorthandon{<>%
36.575 \let\es@quoting\activatequoting}%
36.576 \def\deactivatequoting{%
36.577 \shorthandoff{<>%
36.578 \let\es@quoting\deactivatequoting}}{}%
36.579
36.580 \es@declarealt{<<}{"'}{\begin{quoting}}
36.581 \es@declarealt{>>}{"'}{\end{quoting}}

```

Acute accent shorthands are stored in a macro. If `activeacute` was set as an option it's executed. If not is not deleted for a possible later use in the `cfg` file. In non L^AT_EX 2_ε formats it's always executed.

```

36.582 \begingroup
36.583 \catcode'\'=12
36.584 \gdef\es@activeacute{%
36.585 \es@set@shorthand{'}%
36.586 \def\es@a##1{\es@declare{'##1}{\@tabacckludge'##1}}%
36.587 \es@a a\es@a e\es@a i\es@a o\es@a u%
36.588 \es@a A\es@a E\es@a I\es@a O\es@a U%
36.589 \es@declare{'n}{\~n}%
36.590 \es@declare{'N}{\~N}%
36.591 \es@declare{''}{''}%

```

But spanish allows two category codes for `'`, so both should be taken into account in `\bbl@pr@m@s`.

```

36.592 \let\es@pr@m@s\bbl@pr@m@s
36.593 \def\bbl@pr@m@s{%
36.594 \ifx'\@let@token
36.595 \bbl@afterelse
36.596 \pr@@@s
36.597 \else
36.598 \bbl@afterfi
36.599 \es@pr@m@s
36.600 \fi}%
36.601 \let\es@activeacute\relax}
36.602 \endgroup
36.603
36.604 \if@latex
36.605 \@ifpackagewith{babel}{activeacute}{\es@activeacute}{\
36.606 \else
36.607 \es@activeacute
36.608 \fi

```

And the customization. By default these macros only store the values and do nothing.

```

36.609 \def\es@enumerate#1#2#3#4{\def\es@enum{#1}{#2}{#3}{#4}}
36.610 \def\es@itemize#1#2#3#4{\def\es@item{#1}{#2}{#3}{#4}}
36.611
36.612 \ifes@latex
36.613 \es@enumerate{1.}{a)}{1)}{a'$}
36.614 \def\spanishdashitems{\es@itemize{---}{---}{---}{---}}
36.615 \def\spanishsybitems{%
36.616 \es@itemize
36.617 {\leavevmode\hbox to 1.2ex
36.618 {\hss\vrule height .9ex width .7ex depth -.2ex\hss}}%
36.619 {\textbullet}%
36.620 {\m@th\circ}%
36.621 {\m@th\diamond}}
36.622 \def\spanishsignitems{%
36.623 \es@itemize{\textbullet}%
36.624 {\m@th\circ}%
36.625 {\m@th\diamond}%
36.626 {\m@th\triangleright}}
36.627 \spanishsybitems
36.628 \def\es@enumdef#1#2#3\@@{%
36.629 \if#21%
36.630 \@namedef{theenum#1}{\arabic{enum#1}}%
36.631 \else\if#2a%
36.632 \@namedef{theenum#1}{\emph{\alph{enum#1}}}%
36.633 \else\if#2A%
36.634 \@namedef{theenum#1}{\Alph{enum#1}}%
36.635 \else\if#2i%
36.636 \@namedef{theenum#1}{\roman{enum#1}}%
36.637 \else\if#2I%
36.638 \@namedef{theenum#1}{\Roman{enum#1}}%
36.639 \else\if#2o%
36.640 \@namedef{theenum#1}{\arabic{enum#1}\sptext{o}}%
36.641 \fi\fi\fi\fi\fi
36.642 \toks@{\expandafter{\csname theenum#1\endcsname}}%
36.643 \expandafter\edef\csname labelenum#1\endcsname
36.644 {\noexpand\es@listquot\the\toks@#3}}
36.645 \def\es@guillemot#1#2{%
36.646 \ifmmode#1%
36.647 \else
36.648 \save@sf@q{\penalty\@M
36.649 \leavevmode\hbox{\usefont{U}{lasy}{m}{n}%
36.650 \char#2 \kern-0.19em\char#2 }}%
36.651 \fi}
36.652 \def\layoutspanish{%
36.653 \let\layoutspanish\@empty
36.654 \DeclareTextCommand{\guillemotleft}{OT1}{\es@guillemot\ll{40}}%
36.655 \DeclareTextCommand{\guillemotright}{OT1}{\es@guillemot\gg{41}}%
36.656 \def\@fnsymbol##1%
36.657 {\ifcase##1\or*\or**\or***\or****\or
36.658 *****\or*****\else\@ctrerr\fi}%
36.659 \def\@alph##1%
36.660 {\ifcase##1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
36.661 k\or l\or m\or n\or \~n\or o\or p\or q\or r\or s\or t\or u\or v\or
36.662 w\or x\or y\or z\else\@ctrerr\fi}%
36.663 \def\@Alph##1%
36.664 {\ifcase##1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
36.665 K\or L\or M\or N\or \~N\or O\or P\or Q\or R\or S\or T\or U\or V\or
36.666 W\or X\or Y\or Z\else\@ctrerr\fi}}
36.667
36.668 \es@optlayout{noenumerate}{%
36.669 \def\es@enumerate#1#2#3#4{%
36.670 \es@enumdef{i}#1\@empty\@empty\@@

```

```

36.671 \es@enumdef{ii}#2\@empty\@empty\@@
36.672 \es@enumdef{iii}#3\@empty\@empty\@@
36.673 \es@enumdef{iv}#4\@empty\@empty\@@}%
36.674 \def\p@enumii{\theenumi}%
36.675 \def\p@enumiii{\p@enumii\theenumii}%
36.676 \def\p@enumiv{\p@enumiii\theenumiii}%
36.677 \expandafter\es@enumerate\es@enum}
36.678 \es@optlayout{noitemize}{%
36.679 \def\es@itemize#1#2#3#4{%
36.680 \def\labelitemi{\es@listquot#1}%
36.681 \def\labelitemii{\es@listquot#2}%
36.682 \def\labelitemiii{\es@listquot#3}%
36.683 \def\labelitemiv{\es@listquot#4}}%
36.684 \expandafter\es@itemize\es@item}
36.685 \let\esromanindex\@secondoftwo
36.686 \es@ifoption{ucroman}
36.687 {\def\es@romandef{%
36.688 \def\esromanindex##1#2{##1\uppercase{##2}}}%
36.689 \def\@roman{\@Roman}}}%
36.690 {\def\es@romandef{%
36.691 \def\esromanindex##1#2{##1\protect\es@roman{##2}}}%
36.692 \def\@roman##1{\es@roman{\number##1}}}%
36.693 \def\es@roman##1{\es@scroman{\romannumeral##1}}}%
36.694 \DeclareRobustCommand\es@scroman{\es@xlsc\uppercase\@firstofone}}}%
36.695 \es@optlayout{lcroman}{\es@romandef}
36.696 \newcommand\spanishlcroman{\def\@roman##1{\romannumeral##1}}
36.697 \newcommand\spanishucroman{\def\@roman{\@Roman}}
36.698 \newcommand\spanishscroman{\def\@roman##1{\es@roman{\romannumeral##1}}}
36.699 \es@optlayout{noindentfirst}{%
36.700 \let\@afterindentfalse\@afterindenttrue
36.701 \@afterindenttrue}
36.702 \es@optlayout{nosectiondot}{%
36.703 \def\@secntformat#1{\csname the#1\endcsname.\quad}%
36.704 \def\numberline#1{\hb@xt@{\@tempdima{#1}\if#1&\else.\fi\hfil}}%
36.705 \es@ifoption{nolayout}{\let\layoutspanish\relax}{%
36.706 \es@ifoption{sloppy}{\let\textspanish\relax\let\mathspanish\relax}{%
36.707 \es@ifoption{delayed}{\def\es@layoutspanish{\layoutspanish}}%
36.708 \es@ifoption{preindex}{\AtEndOfPackage{\RequirePackage{romanidx}}}{%

```

We need to execute the following code when babel has been run, in order to see if `spanish` is the main language.

```

36.709 \AtEndOfPackage{%
36.710 \let\es@activeacute\@undefined
36.711 \def\bbl@tempa{spanish}%
36.712 \ifx\bbl@main@language\bbl@tempa
36.713 \nameuse{es@layoutspanish}%
36.714 \addto\es@select{%
36.715 \@ifstar{\PackageError{spanish}%
36.716 {Old syntax--use es-nolayout}%
36.717 {If you don't want changes in layout\MessageBreak
36.718 use the es-nolayout package option}}}%
36.719 {}}%
36.720 \AtBeginDocument{\layoutspanish}%
36.721 \fi
36.722 \selectspanish}
36.723 \fi

```

After restoring the catcode of `~` and setting the minimal values for hyphenation, the `.ldf` is finished.

```

36.724 \es@savecatcodes
36.725 \providehyphenmins{\CurrentOption}{\tw@{\tw@}
36.726 \ifes@latex\else
36.727 \es@select

```

```

36.728 \fi
36.729 \ldf@finish{spanish}
36.730 \csname activatequoting\endcsname
36.731 \endcode

```

That's all in the main file.

The `spanish` option writes a macro in the page field of *MakeIndex* in entries with small caps number, and they are rejected. This program is a preprocessor which moves this macro to the entry field. It can be called from the main document as a package or with the package option `es-preindex`.

```

36.732 (*indexes)
36.733 \makeatletter
36.734
36.735 \ifundefined{es@idxfile}
36.736   {\def\spanishindexchars#1#2#3{%
36.737     \edef\es@encap{'\expandafter\noexpand\csname\string#1\endcsname}%
36.738     \edef\es@openrange{'\expandafter\noexpand\csname\string#2\endcsname}%
36.739     \edef\es@closerange{'\expandafter\noexpand\csname\string#3\endcsname}}%
36.740   \spanishindexchars{|}{(}{)}%
36.741   \ifx\documentclass\@twoclasseserror
36.742     \edef\es@idxfile{\jobname}%
36.743     \AtEndDocument{%
36.744       \addto\@defaultsubs{%
36.745         \immediate\closeout\@indexfile
36.746         \input{romanidx.sty}}}%
36.747     \expandafter\endinput
36.748   \fi}{
36.749
36.750 \newcount\es@converted
36.751 \newcount\es@processed
36.752
36.753 \def\es@split@file#1.#2\@{#1}
36.754 \def\es@split@ext#1.#2\@{#2}
36.755
36.756 \ifundefined{es@idxfile}
36.757   {\typein[\answer]{^^JArchivo que convertir^^J%
36.758     (extension por omision .idx):}}
36.759   {\let\answer\es@idxfile}
36.760
36.761 \expandafter\in@{.}{\answer}
36.762 \ifin@
36.763   \edef\es@input@file{\expandafter\es@split@file\answer\@}
36.764   \edef\es@input@ext{\expandafter\es@split@ext\answer\@}
36.765 \else
36.766   \edef\es@input@file{\answer}
36.767   \def\es@input@ext{idx}
36.768 \fi
36.769
36.770 \ifundefined{es@idxfile}
36.771   {\typein[\answer]{^^JArchivo de destino^^J%
36.772     (archivo por omision: \es@input@file.eix,^^J%
36.773     extension por omision .eix):}}
36.774   {\let\answer\es@idxfile}
36.775 \ifx\answer\@empty
36.776   \edef\es@output{\es@input@file.eix}
36.777 \else
36.778   \expandafter\in@{.}{\answer}
36.779   \ifin@
36.780     \edef\es@output{\answer}
36.781   \else
36.782     \edef\es@output{\answer.eix}
36.783   \fi

```

```

36.784 \fi
36.785
36.786 \@ifundefined{es@idxfile}
36.787 {\typein[\answer]{%
36.788   ^^J?Se ha usado algun esquema especial de controles^^J%
36.789   de MakeIndex para encap, open_range o close_range?^^J%
36.790   [s/n] (n por omision)}}
36.791 {\def\answer{n}}
36.792
36.793 \if s\answer
36.794 \typein[\answer]{^^JCaracter para 'encap'^^J%
36.795   (\string| por omision)}
36.796 \ifx\answer@empty\else
36.797   \edef\es@encap{%
36.798     '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
36.799 \fi
36.800 \typein[\answer]{^^JCaracter para 'open_range'^^J%
36.801   (\string( por omision)}
36.802 \ifx\answer@empty\else
36.803   \edef\es@openrange{%
36.804     '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
36.805 \fi
36.806 \typein[\answer]{^^JCaracter para 'close_range'^^J%
36.807   (\string) por omision)}
36.808 \ifx\answer@empty\else
36.809   \edef\es@closerange{%
36.810     '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
36.811 \fi
36.812 \fi
36.813
36.814 \newwrite\es@indexfile
36.815 \immediate\openout\es@indexfile=\es@output
36.816
36.817 \newif\ifes@encapsulated
36.818
36.819 \def\es@roman#1{#1}
36.820 \edef\es@slash{\expandafter\@gobble\string\\}
36.821
36.822 \def\indexentry{%
36.823   \begingroup
36.824   \@sanitize
36.825   \es@indexentry}
36.826
36.827 \begingroup
36.828
36.829 \catcode'\|=12 \lccode'\|= \es@encap\relax
36.830 \catcode'\(=12 \lccode'\(= \es@openrange\relax
36.831 \catcode'\)=12 \lccode'\)= \es@closerange\relax
36.832
36.833 \lowercase{
36.834 \gdef\es@indexentry#1{%
36.835   \endgroup
36.836   \advance\es@processed@ne
36.837   \es@encapsulatedfalse
36.838   \es@bar@idx#1|\@@
36.839   \es@idxentry}%
36.840 }
36.841
36.842 \lowercase{
36.843 \gdef\es@idxentry#1{%
36.844   \in@{\es@roman}{#1}%
36.845   \ifin@

```

```

36.846 \advance\es@converted\@ne
36.847 \immediate\write\es@indexfile{%
36.848 \string\indexentry{\es@b\ifes@encapsulated\es@p\fi esromanindex%
36.849 {\ifx\es@a\@empty\else\es@slash\es@a\fi}}{#1}}%
36.850 \else
36.851 \immediate\write\es@indexfile{%
36.852 \string\indexentry{\es@b\ifes@encapsulated|\es@p\es@a\fi}{#1}}%
36.853 \fi}
36.854 }
36.855
36.856 \lowercase{
36.857 \gdef\es@bar@idx#1|#2\@@{%
36.858 \def\es@b{#1}\def\es@a{#2}%
36.859 \ifx\es@a\@empty\else\es@encapsulatedtrue\es@bar@eat#2\fi}
36.860 }
36.861
36.862 \lowercase{
36.863 \gdef\es@bar@eat#1#2|{\def\es@p{#1}\def\es@a{#2}%
36.864 \edef\es@t{(\ifx\es@t\es@p
36.865 \else\edef\es@t{)}\ifx\es@t\es@p
36.866 \else
36.867 \edef\es@a{\es@p\es@a}\let\es@p\@empty%
36.868 \fi\fi}
36.869 }
36.870
36.871 \endgroup
36.872
36.873 \input \es@input@file.\es@input@ext
36.874
36.875 \immediate\closeout\es@indexfile
36.876
36.877 \typeout{*****}
36.878 \typeout{Se ha procesado: \es@input@file.\es@input@ext }
36.879 \typeout{Lineas leidas: \the\es@processed}
36.880 \typeout{Lineas convertidas: \the\es@converted}
36.881 \typeout{Resultado en: \es@output}
36.882 \ifnum\es@converted>\z@
36.883 \typeout{Genere el indice a partir de ese archivo}
36.884 \else
36.885 \typeout{No se ha convertido nada. Se puede generar}
36.886 \typeout{el .ind directamente de \es@input@file.\es@input@ext}
36.887 \fi
36.888 \typeout{*****}
36.889
36.890 \@ifundefined{es@sdef}{\@@end}{}
36.891
36.892 \endinput
36.893 </indexes>

```


37 The Catalan language

The file `catalan.dtx`³⁷ defines all the language-specific macro's for the Catalan language.

For this language only the double quote character (") is made active by default. In table 12 an overview is given of the new macros defined and the new meanings of ". Additionally to that, the user can explicitly activate the acute accent or apostrophe (') and/or the grave accent (`) characters by using the `activeacute` and `activegrave` options. In that case, the definitions shown in table 13 also become available³⁸.

<code>\l.l</code>	geminated-l digraph (similar to l·l). <code>\L.L</code> produces the uppercase version.
<code>\lgem</code>	geminated-l digraph (similar to l·l). <code>\Lgem</code> produces the uppercase version.
<code>\up</code>	Macro to help typing raised ordinals, like 1 ^{er} . Takes one argument.
<code>\-</code>	like the old <code>\-</code> , but allowing hyphenation in the rest of the word.
<code>"i</code>	i with diaeresis, allowing hyphenation in the rest of the word. Valid for the following vowels: i, u (both lowercase and uppercase).
<code>"c</code>	c-cedilla (ç). Valid for both uppercase and lowercase c.
<code>"l</code>	geminated-l digraph (similar to l·l). Valid for both uppercase and lowercase l.
<code>"<</code>	French left double quotes (similar to <<).
<code>"></code>	French right double quotes (similar to >>).
<code>"-</code>	explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>" </code>	disable ligature at this position.

Table 12: Extra definitions made by file `catalan.ldf` (activated by default)

<code>'e</code>	acute accented a, allowing hyphenation in the rest of the word. Valid for the following vowels: e, i, o, u (both lowercase and uppercase).
<code>'a</code>	grave accented a, allowing hyphenation in the rest of the word. Valid for the following vowels: a, e, o (both lowercase and uppercase).

Table 13: Extra definitions made by file `catalan.ldf` (activated only when using the options `activeacute` and `activegrave`)

These active accents characters behave according to their original definitions if not followed by one of the characters indicated in that table.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

37.1 `{*code}`

37.2 `\LdfInit{catalan}\captionscatalan`

When this file is read as an option, i.e. by the `\usepackage` command, `catalan` could be an 'unknown' language in which case we have to make it known. So we

³⁷The file described in this section has version number v2.2p and was last revised on 2005/03/29.

³⁸Please note that if the acute accent character is active, it is necessary to take special care of coding apostrophes in a way which cannot be confounded with accents. Therefore, it is necessary to type `l'{}estri` instead of `l'estri`.

check for the existence of `\l@catalan` to see whether we have to do something here.

```
37.3 \ifx\l@catalan\@undefined
37.4 \@nopatterns{Catalan}
37.5 \adddialect\l@catalan0
37.6 \fi
```

The next step consists of defining commands to switch to (and from) the Catalan language.

`\catalanhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
37.7 \providehyphenmins{catalan}{\tw@{\tw@}}
```

`\captionscatalan` The macro `\captionscatalan` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
37.8 \addto\captionscatalan{%
37.9 \def\prefacename{Pr\‘oleg}%
37.10 \def\refname{Refer\‘encies}%
37.11 \def\abstractname{Resum}%
37.12 \def\bibname{Bibliografia}%
37.13 \def\chaptername{Cap\‘\{i\}tol}%
37.14 \def\appendixname{Ap\‘endix}%
37.15 \def\contentsname{\‘Index}%
37.16 \def\listfigurename{\‘Index de figures}%
37.17 \def\listtablename{\‘Index de taules}%
37.18 \def\indexname{\‘Index alfab\‘etic}%
37.19 \def\figurename{Figura}%
37.20 \def\tablename{Taula}%
37.21 \def\partname{Part}%
37.22 \def\enclname{Adjunt}%
37.23 \def\ccname{C\‘opies a}%
37.24 \def\headtoname{A}%
37.25 \def\pagename{P\‘agina}%
37.26 \def\seename{Vegeu}%
37.27 \def\alsoname{Vegeu tamb\‘e}%
37.28 \def\proofname{Demostraci\‘o}%
37.29 \def\glossaryname{Glossari}%
37.30 }
```

`\datecatalan` The macro `\datecatalan` redefines the command `\today` to produce Catalan dates. Months are written in lowercase³⁹.

```
37.31 \def\datecatalan{%
37.32 \def\today{\number\day~\ifcase\month\or
37.33 de gener\or de febrer\or de mar\c{c}\or d’abril\or de maig\or
37.34 de juny\or de juliol\or d’agost\or de setembre\or d’octubre\or
37.35 de novembre\or de desembre\fi
37.36 \space de~\number\year}}
```

`\extrascatalan` The macro `\extrascatalan` will perform all the extra definitions needed for the Catalan language. The macro `\noextrascatalan` is used to cancel the actions of `\extrascatalan`.

To improve hyphenation we give the grave character (‘) a non-zero lower case code; when we do that T_EX will find more breakpoints in words that contain this character in its rôle as apostrophe.

```
37.37 \addto\extrascatalan{%
37.38 \lccode‘=‘}
37.39 \addto\noextrascatalan{%
37.40 \lccode‘=0}
```

³⁹This seems to be the common practice. See for example: E. Coromina, *El 9 Nou: Manual de redacció i estil*, Ed. Eumo, Vic, 1993

For Catalan, some characters are made active or are redefined. In particular, the " character receives a new meaning; this can also happen for the ' character and the ` character when the options `activegrave` and/or `activeacute` are specified.

```
37.41 \addto\extrascatalan{\languageshorthands{catalan}}
37.42 \initiate@active@char{"}
37.43 \addto\extrascatalan{\bbl@activate{"}}
```

Because the grave character is being used in constructs such as `\catcode`=\active` it needs to have it's original category code" when the auxiliary file is being read. Note that this file is read twice, once at the beginning of the document; then there is no problem; but the second time it is read at the end of the document to check whether any labels changes. It's this second time round that the activated grave character leads to error messages.

```
37.44 \@ifpackagewith{babel}{activegrave}{%
37.45   \AtBeginDocument{%
37.46     \if@filesw\immediate\write\@auxout{\catcode096=12}\fi}
37.47   \initiate@active@char{'}%
37.48   }{}
37.49 \@ifpackagewith{babel}{activegrave}{%
37.50   \addto\extrascatalan{\bbl@activate{'}}%
37.51   }{}
37.52 \@ifpackagewith{babel}{activeacute}{%
37.53   \initiate@active@char{'}%
37.54   }{}
37.55 \@ifpackagewith{babel}{activeacute}{%
37.56   \addto\extrascatalan{\bbl@activate{'}}%
37.57   }{}

```

Now make sure that the characters that have been turned into shorthanfd characters expand to 'normal' characters outside the catalan environment.

```
37.58 \addto\noextrascatalan{\bbl@deactivate{"}}
37.59 \@ifpackagewith{babel}{activegrave}{%
37.60   \addto\noextrascatalan{\bbl@deactivate{'}}{}
37.61 \@ifpackagewith{babel}{activeacute}{%
37.62   \addto\noextrascatalan{\bbl@deactivate{'}}{}

```

Apart from the active characters some other macros get a new definition. Therefore we store the current ones to be able to restore them later. When their current meanings are saved, we can safely redefine them.

We provide new definitions for the accent macros when one or both of the options `activegrave` or `activeacute` were specified.

```
37.63 \addto\extrascatalan{%
37.64   \babel@save\"%
37.65   \def\"{\protect\@umlaut}}%
37.66 \@ifpackagewith{babel}{activegrave}{%
37.67   \babel@save\'%
37.68   \addto\extrascatalan{\def\'{\protect\@grave}}
37.69   }{}
37.70 \@ifpackagewith{babel}{activeacute}{%
37.71   \babel@save\'%
37.72   \addto\extrascatalan{\def\'{\protect\@acute}}
37.73   }{}

```

All the code above is necessary because we need a few extra active characters. These characters are then used as indicated in tables 12 and 13.

<code>\dieresis</code>	The original definition of \" is stored as <code>\dieresis</code> , because the definition of
<code>\textacute</code>	\" might not be the default plain T _E X one. If the user uses POSTSCRIPT fonts
<code>\textgrave</code>	with the Adobe font encoding the " character is not in the same position as in Knuth's font encoding. In this case \" will not be defined as <code>\accent"7F 1</code> , but as <code>\accent'310 #1</code> . Something similar happens when using fonts that follow the

Cork encoding. For this reason we save the definition of `\` and use that in the definition of other macros. We do likewise for `\‘`, and `\’`.

```

37.74 \let\dieresis\"
37.75 \@ifpackagewith{babel}{activegrave}{\let\textgrave\'}{}
37.76 \@ifpackagewith{babel}{activeacute}{\let\textacute\'}{}

\@umlaut We check the encoding and if not using T1, we make the accents expand but
\@acute enabling hyphenation beyond the accent. If this is the case, not all break positions
\@grave will be found in words that contain accents, but this is a limitation in TEX. An
unsolved problem here is that the encoding can change at any time. The definitions
below are made in such a way that a change between two 256-char encodings
are supported, but changes between a 128-char and a 256-char encoding are not
properly supported. We check if T1 is in use. If not, we will give a warning and
proceed redefining the accent macros so that TEX at least finds the breaks that
are not too close to the accent. The warning will only be printed to the log file.

37.77 \ifx\DeclareFontShape\@undefined
37.78 \wlog{Warning: You are using an old LaTeX}
37.79 \wlog{Some word breaks will not be found.}
37.80 \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
37.81 \@ifpackagewith{babel}{activeacute}{%
37.82 \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
37.83 \@ifpackagewith{babel}{activegrave}{%
37.84 \def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{}
37.85 \else
37.86 \ifx\f@encoding\bbl@t@one
37.87 \let\@umlaut\dieresis
37.88 \@ifpackagewith{babel}{activeacute}{%
37.89 \let\@acute\textacute}{%
37.90 \@ifpackagewith{babel}{activegrave}{%
37.91 \let\@grave\textgrave}{%
37.92 \else
37.93 \wlog{Warning: You are using encoding \f@encoding\space
37.94 instead of T1.}
37.95 \wlog{Some word breaks will not be found.}
37.96 \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
37.97 \@ifpackagewith{babel}{activeacute}{%
37.98 \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
37.99 \@ifpackagewith{babel}{activegrave}{%
37.100 \def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{}
37.101 \fi
37.102 \fi

```

If the user setup has extended fonts, the Ferguson macros are required to be defined. We check for their existence and, if defined, expand to whatever they are defined to. For instance, `\’a` would check for the existence of a `\@ac@a` macro. It is assumed to expand to the code of the accented letter. If it is not defined, we assume that no extended codes are available and expand to the original definition but enabling hyphenation beyond the accent. This is as best as we can do. It is better if you have extended fonts or ML- $\text{T}_{\text{E}}\text{X}$ because the hyphenation algorithm can work on the whole word. The following macros are directly derived from ML- $\text{T}_{\text{E}}\text{X}$.⁴⁰

Now we can define our shorthands: the diaeresis and “*ela geminada*” support,

```

37.103 \declare@shorthand{catalan}{\i}{\textormath{\@umlaut\i}{\ddot\imath}}

```

⁴⁰A problem is perceived here with these macros when used in a multilingual environment where extended hyphenation patterns are available for some but not all languages. Assume that no extended patterns exist at some site for French and that `french.sty` would adopt this scheme too. In that case, `\’e` in French would produce the combined accented letter, but hyphenation around it would be suppressed. Both language options would need an independent method to know whether they have extended patterns available. The precise impact of this problem and the possible solutions are under study.

```

37.104 \declare@shorthand{catalan}{\l}{\lgem{}}
37.105 \declare@shorthand{catalan}{\u}{\textormath{\@umlaut u}{\ddot u}}
37.106 \declare@shorthand{catalan}{\I}{\textormath{\@umlaut I}{\ddot I}}
37.107 \declare@shorthand{catalan}{\L}{\Lgem{}}
37.108 \declare@shorthand{catalan}{\U}{\textormath{\@umlaut U}{\ddot U}}

cedille,
37.109 \declare@shorthand{catalan}{\c}{\textormath{\c c}{\text{\prime} c}}
37.110 \declare@shorthand{catalan}{\C}{\textormath{\c C}{\text{\prime} C}}

‘french’ quote characters,
37.111 \declare@shorthand{catalan}{\<}{\%
37.112 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
37.113 \declare@shorthand{catalan}{\>}{\%
37.114 \textormath{\guillemotright}{\mbox{\guillemotright}}}

grave accents,
37.115 \@ifpackagewith{babel}{activegrave}{\%
37.116 \declare@shorthand{catalan}{\a}{\textormath{\@grave a}{\grave a}}
37.117 \declare@shorthand{catalan}{\e}{\textormath{\@grave e}{\grave e}}
37.118 \declare@shorthand{catalan}{\o}{\textormath{\@grave o}{\grave o}}
37.119 \declare@shorthand{catalan}{\A}{\textormath{\@grave A}{\grave A}}
37.120 \declare@shorthand{catalan}{\E}{\textormath{\@grave E}{\grave E}}
37.121 \declare@shorthand{catalan}{\O}{\textormath{\@grave O}{\grave O}}
37.122 \declare@shorthand{catalan}{\‘}{\textquotedblleft}{\%}
37.123 \}{}

acute accents,
37.124 \@ifpackagewith{babel}{activeacute}{\%
37.125 \declare@shorthand{catalan}{\a}{\textormath{\@acute a}{\text{\prime} a}}
37.126 \declare@shorthand{catalan}{\e}{\textormath{\@acute e}{\text{\prime} e}}
37.127 \declare@shorthand{catalan}{\i}{\textormath{\@acute i}{\text{\prime} i}}
37.128 \declare@shorthand{catalan}{\o}{\textormath{\@acute o}{\text{\prime} o}}
37.129 \declare@shorthand{catalan}{\u}{\textormath{\@acute u}{\text{\prime} u}}
37.130 \declare@shorthand{catalan}{\A}{\textormath{\@acute A}{\text{\prime} A}}
37.131 \declare@shorthand{catalan}{\E}{\textormath{\@acute E}{\text{\prime} E}}
37.132 \declare@shorthand{catalan}{\I}{\textormath{\@acute I}{\text{\prime} I}}
37.133 \declare@shorthand{catalan}{\O}{\textormath{\@acute O}{\text{\prime} O}}
37.134 \declare@shorthand{catalan}{\U}{\textormath{\@acute U}{\text{\prime} U}}
37.135 \declare@shorthand{catalan}{\’}{\%
37.136 \textormath{\csname normal@char\string\endcsname}{\text{\prime}}

the acute accent,
37.137 \declare@shorthand{catalan}{\’}{\%
37.138 \textormath{\textquotedblright}{\sp\bggroup\prim@s}}
37.139 \}{}

and finally, some support definitions
37.140 \declare@shorthand{catalan}{\nobreak}{\nobreak-\bbl@allowhyphens}
37.141 \declare@shorthand{catalan}{\discretionary}{\kern.03em}{\allowhyphens}
37.142 \textormath{\nobreak\discretionary}{\kern.03em}{\allowhyphens}
37.143 \allowhyphens}

```

\- All that is left now is the redefinition of \-. The new version of \- should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of TeX in this respect is unfortunate for Catalan but not as much as for Dutch or German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that TeX can generate from the hyphenation patterns. However, the average length of words in Catalan makes this desirable and so it is kept here.

```

37.144 \addto\extrascatalan{\%
37.145 \babel@save{\-}{\%
37.146 \def\-\{\bbl@allowhyphens\discretionary{-}{-}{\bbl@allowhyphens}}

```

`\lgem` Here we define a macro for typing the catalan “ela geminada” (geminated l). The macros `\lgem` and `\Lgem` have been chosen for its lowercase and uppercase representation, respectively⁴¹.

The code used in the actual macro used is a combination of the one proposed by Feruglio and Fuster⁴² and the proposal⁴³ from Valiente presented at the T_EX Users Group Annual Meeting in 1995. This last proposal has not been fully implemented due to its limitation to CM fonts.

```

37.147 \newdimen\leftllkern \newdimen\rightllkern \newdimen\raiselldim
37.148 \def\lgem{%
37.149   \ifmmode
37.150     \csname normal@char\string\endcsname 1%
37.151   \else
37.152     \leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
37.153     \setbox0\hbox{1}\setbox1\hbox{1/}\setbox2\hbox{.}%
37.154     \advance\raiselldim by \the\fontdimen5\the\font
37.155     \advance\raiselldim by -\ht2%
37.156     \leftllkern=-.25\wd0%
37.157     \advance\leftllkern by \wd1%
37.158     \advance\leftllkern by -\wd0%
37.159     \rightllkern=-.25\wd0%
37.160     \advance\rightllkern by -\wd1%
37.161     \advance\rightllkern by \wd0%
37.162     \allowhyphens\discretionary{1-}{1}%
37.163     {\hbox{1}\kern\leftllkern\raise\raiselldim\hbox{.}}%
37.164     \kern\rightllkern\hbox{1}}\allowhyphens
37.165   \fi
37.166 }
37.167 \def\Lgem{%
37.168   \ifmmode
37.169     \csname normal@char\string\endcsname L%
37.170   \else
37.171     \leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
37.172     \setbox0\hbox{L}\setbox1\hbox{L/}\setbox2\hbox{.}%
37.173     \advance\raiselldim by .5\ht0%
37.174     \advance\raiselldim by -.5\ht2%
37.175     \leftllkern=-.125\wd0%
37.176     \advance\leftllkern by \wd1%
37.177     \advance\leftllkern by -\wd0%
37.178     \rightllkern=-\wd0%
37.179     \divide\rightllkern by 6%
37.180     \advance\rightllkern by -\wd1%
37.181     \advance\rightllkern by \wd0%
37.182     \allowhyphens\discretionary{L-}{L}%
37.183     {\hbox{L}\kern\leftllkern\raise\raiselldim\hbox{.}}%
37.184     \kern\rightllkern\hbox{L}}\allowhyphens
37.185   \fi
37.186 }

```

`\l.1` It seems to be the most natural way of entering the “ela geminda” to use the sequences `\l.1` and `\L.L`. These are not really macro’s by themselves but the macros `\l` and `\L` with delimited arguments. Therefor we define two macros that check if the next character is a period. If not the “polish l” will be typeset, otherwise a “ela geminada” will be typeset and the next two tokens will be ‘eaten’.

```

37.187 \AtBeginDocument{%
37.188   \let\lslash\l
37.189   \let\Lslash\L

```

⁴¹The macro names `\l1` and `\LL` were not taken because of the fact that `\l1` is already used in mathematical mode.

⁴²G. Valiente and R. Fuster, Typesetting Catalan Texts with T_EX, *TUGboat* **14**(3), 1993.

⁴³G. Valiente, Modern Catalan Typographical Conventions, *TUGboat* **16**(3), 1995.

```

37.190 \DeclareRobustCommand\l{\@ifnextchar.\bbl@l\lslash}
37.191 \DeclareRobustCommand\L{\@ifnextchar.\bbl@L\Lslash}}
37.192 \def\bbl@l#1#2{\lgem}
37.193 \def\bbl@L#1#2{\Lgem}

```

`\up` A macro for typesetting things like 1^{er} as proposed by Raymon Seroul⁴⁴.

```

37.194 \DeclareRobustCommand*\up}[1]{\textsuperscript{#1}}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

37.195 \ldf@finish{catalan}
37.196 \code

```

⁴⁴This macro has been borrowed from `francais.dtx`

38 This file

This file defines all the language-specific macros for the Galician language. The file `galician.dtx` was translated in January 2007 by Javier A. Múgica from `spanish.dtx`. It was given the version number 4.3, based on the version for `spanish.dtx` at those times, that was 4.2b. The original author from v4.0 to 4.2b was Javier Bezoz. Previous versions were written by Julio Sánchez.

I decided to make *tabula rasa* of all `\changes` logs. Only changes from `spanish` 4.2b to `galician` 4.3 and thereafter are documented. The change history for the original `spanish.dtx` can be found in that file.

39 The Galician language

Customization is made following mainly the books on the subject by José Martínez de Sousa and Xosé Feixó Cid. By typesetting `galician.dtx` directly you will get the full documentation (regrettably is in Galician only, but it is pretty long). References in this part refers to that document. There are several additional features documented in the Galician version only.

This style provides:

- Translations following the International L^AT_EX conventions, as well as `\today`.
- Shorthands listed in Table 14. Examples in subsection 3.4 are illustrative. Note that `"~` has a special meaning in `galician` different to other languages, and is used mainly in linguistic contexts.

<code>'a</code>	acute accented a. Also for: e, i, o, u (both lowercase and uppercase).
<code>'n</code>	<code>ñ</code> (also uppercase).
<code>~n</code>	<code>ñ</code> (also uppercase). Deprecated.
<code>"u</code>	<code>ü</code> (also uppercase).
<code>"i</code>	<code>ï</code> (also uppercase).
<code>"a</code>	Ordinal numbers (also <code>"A</code> , <code>"o</code> , <code>"O</code>).
<code>"rr</code>	<code>rr</code> , but <code>-r</code> when hyphenated
<code>"-</code>	Like <code>\-</code> , but allowing hyphenation in the rest the word.
<code>"=</code>	Like <code>-</code> , but allowing hyphenation in the rest the word.
<code>"~</code>	The hyphen is repeated at the very beginning of the next line if the word is hyphenated at this point.
<code>"</code>	Like <code>"-</code> but producing no hyphen sign.
<code>~-</code>	Like <code>-</code> but with no break after the hyphen. Also for: en-dashes (<code>~--</code>) and em-dashes (<code>~---</code>).
<code>"/</code>	A slash slightly lowered, if necessary.
<code>" </code>	disable ligatures at this point.
<code><<</code>	Left guillemets.
<code>>></code>	Right guillemets.
<code>"<</code>	<code>\begin{quoting}</code> . (See text.)
<code>"></code>	<code>\end{quoting}</code> . (See text.)

Table 14: Extra definitions made by file `galician.ldf`

- `\deactivatetilden` deactivates the `~n` and `~N` shorthands.
- *In math mode* a dot followed by a digit is replaced by a decimal comma.
- Galicians ordinals and abbeviations with `\sptext` as, for instance, `1\sptext{o}`. The preceptive dot is included.

- Accented functions: `lím`, `máx`, `mín`, `mód`. You may globally omit the accents with `\unaccentedoperators`. Spaced functions: `arc cos`, etc. You may globally kill that space with `\unspacedoperators`. `\dotlessi` is provided for use in math mode.
- A `quoting` environment and a related pair of shorthands `<<` and `>>`. The command `\deactivatequoting` deactivates these shorthand in case you want to use `<` and `>` in some AMS commands and numerical comparisons.
- The command `\selectgalician` selects the `galician` language *and* its shorthands. (Intended for the preamble.)
- `\frenchspacing` is used.
- `\dots` is redefined. It is now equal to typing tree points in a row (it preserves the space following).
- There is a small space before `\%`.
- `\msc` provides lowercase small caps. (See subsection 3.10.)

Just in case `galician` is the main language, the group `\layoutgalician` is activated, which modifies the standard classes through the whole document (it cannot be deactivated) in the following way:

- Both `enumerate` and `itemize` are adapted to Galician rules.
- Both `\alph` and `\Alph` include \tilde{n} after n .
- Symbol footmarks are one, two, three, etc., asteriscs.
- OT1 guillemets are generated with two `lasy` symbols instead of small `\ll` and `\gg`.
- `\roman` is redefined to write small caps roman numerals, since lowercase roman numerals are not allowed. However, *MakeIndex* rejects entries containing pages in that format. The `.idx` file must be preprocessed if the document has this kind of entries with the provided `romanidx.tex` tool—just `TEX` it and follow the instructions.
- There is a dot after section numbers in titles and `toc`.

This group is ignored if you write `\selectgalician*` in the preamble.

Some additional commands are provided to be used in the `galician.cfg` file:

- With `\gl@activeacute` acute accents are always active, overriding the default `babel` behaviour.
- `\gl@enumerate` sets the labels to be used by `enumerate`. The same applies to `\gl@itemize` and `itemize`.
- `\gl@operators` stores the operator commands. All of them are canceled with

```
\let\gl@operators\relax
```

The commands `\deactivatequoting`, `\deactivatetilden` and `\selectgalician` may be used in this file, too.

A subset of these commands is provided for use in Plain `TEX` (with `\input galician.sty`).

39.1 The Code

This file provides definition for both L^AT_EX 2_ε and non L^AT_EX 2_ε formats.
Identify the ldf file.

```
39.1 (*code)
39.2 \ProvidesLanguage{galician.lda}
39.3 [2008/07/06 v4.3c Galician support from the babel system]
```

The macro \LdfInit takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc. When this file is read as an option, i.e. by the \usepackage command, galician will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of \l@galician to see whether we have to do something here.

```
39.4 \LdfInit{galician}\captionsgalician
39.5 \ifx\undefined\l@galician
39.6   \@nopatterns{Galician}
39.7   \adddialect\l@galician0
39.8 \fi
```

We define some tools which will be used in that style file: (1) we make sure that ~ is active, (2) \gl@delayed delays the expansion of the code in conditionals (in fact, quite similar to \bbl@afterfi).

```
39.9 \edef\gl@savcatcodes{%
39.10   \catcode'\noexpand\~=\the\catcode'\~
39.11   \catcode'\noexpand\"=\the\catcode\"}
39.12 \catcode'\~=\active
39.13 \catcode\"=12
39.14 \long\def\gl@delayed#1\then#2\else#3\fi{%
39.15   #1%
39.16   \expandafter\@firstoftwo
39.17   \else
39.18   \expandafter\@secondoftwo
39.19   \fi
39.20   {#2}{#3}}
```

Two tests are introduced. The first one tells us if the format is L^AT_EX 2_ε, and the second one if the format is Plain or any other. If both are false, the format is L^AT_EX 2.09.

```
39.21 \gl@delayed
39.22 \expandafter\ifx\csname documentclass\endcsname\relax\then
39.23   \let\ifes@LaTeXe\iffalse
39.24 \else
39.25   \let\ifes@LaTeXe\iftrue
39.26 \fi
39.27 \gl@delayed
39.28 \expandafter\ifx\csname newenvironment\endcsname\relax\then
39.29   \let\ifes@plain\iftrue
39.30 \else
39.31   \let\ifes@plain\iffalse
39.32 \fi
```

Translations for captions.

```
39.33 \addto\captionsgalician{%
39.34   \def\prefacename{Prefacio}%
39.35   \def\refname{Referencias}%
39.36   \def\abstractname{Resumo}%
39.37   \def\bibname{Bibliograf'\{i}a}%
39.38   \def\chaptername{Cap'\{i}tulo}%
39.39   \def\appendixname{Ap'endice}%
39.40   \def\listfigurename{'Indice de figuras}%
39.41   \def\listtablename{'Indice de cadros}%
39.42   \def\indexname{'Indice alfab'etico}%
39.43   \def\figurename{Figura}%
39.44   \def\tablename{Cadro}%
```

```

39.45 \def\partname{Parte}%
39.46 \def\enclname{Adxunto}%
39.47 \def\ccname{Copia a}%
39.48 \def\headtoname{A}%
39.49 \def\pagename{P\'axina}%
39.50 \def\seenname{v\'exase}%
39.51 \def\alsoname{v\'exase tam\'en}%
39.52 \def\proofname{Demostraci\'on}%
39.53 \def\glossaryname{Glosario}}
39.54
39.55 \expandafter\ifx\csname chapter\endcsname\relax
39.56 \addto\captionsgalician{\def\contentsname{\'Indice}}
39.57 \else
39.58 \addto\captionsgalician{\def\contentsname{\'Indice xeral}}
39.59 \fi

```

And the date.

```

39.60 \def\dategalician{%
39.61 \def\today{\the\day~de \ifcase\month\or xaneiro\or febreiro\or
39.62     marzo\or abril\or maio\or xu~no\or xullo\or agosto\or
39.63     setembro\or outubro\or novembro\or decembro\fi
39.64     \ \ifnum\year>1999\gl@yearl\else de\fi~\the\year}}
39.65 \def\galiciandatedo{\def\gl@yearl{do}}
39.66 \def\galiciandatede{\def\gl@yearl{de}}
39.67 \galiciandatedo

```

The basic macros to select the language, in the preamble or the config file.

Use of `\selectlanguage` should be avoided at this early stage because the active chars are not yet active. `\selectgalician` makes them active.

```

39.68 \def\selectgalician{%
39.69 \def\selectgalician{%
39.70 \def\selectgalician{%
39.71 \PackageWarning{galician}{Extra \string\selectgalician ignored}}%
39.72 \gl@select}}
39.73
39.74 \@onlypreamble\selectgalician
39.75
39.76 \def\gl@select{%
39.77 \let\gl@select\@undefined
39.78 \selectlanguage{galician}%
39.79 \catcode\'"\active\catcode\'~=\active}

```

Instead of joining all the extras directly in `\extrasgalician`, we subdivide them in three further groups.

```

39.80 \def\extrasgalician{%
39.81 \textgalician
39.82 \mathgalician
39.83 \ifx\shorthandsgalician\@empty
39.84 \galiciandeactivate{.'~<>}%
39.85 \languageshorthands{none}%
39.86 \else
39.87 \shorthandsgalician
39.88 \fi}
39.89 \def\noextrasgalician{%
39.90 \ifx\textgalician\@empty\else
39.91 \notextgalician
39.92 \fi
39.93 \ifx\mathgalician\@empty\else
39.94 \nomathgalician
39.95 \fi
39.96 \ifx\shorthandsgalician\@empty\else
39.97 \noshorthandsgalician
39.98 \fi
39.99 \gl@reviveshorthands}

```

And the first of these sub-groups is defined.

```
39.100 \addto\textgalician{%
39.101   \babel@save\sptext
39.102   \def\sptext{\protect\gl@sptext}}
```

The definition of `\sptext` is more elaborated than that of `\textsuperscript`.

With uppercase superscript text the `scriptscriptsize` is used. The mandatory dot is already included. There are two versions, depending on the format.

```
39.103 \ifcsLaTeXe %<<<<<<
39.104   \newcommand\gl@sptext[1]{%
39.105     {\setbox\z@\hbox{8}\dimen@ \ht\z@
39.106       \csname S@f@size\endcsname
39.107       \edef\@tempa{\def\noexpand\@tempc{#1}%
39.108         \lowercase{\def\noexpand\@tempb{#1}}}\@tempa
39.109       \ifx\@tempb\@tempc
39.110         \fontsize\sf@size\z@
39.111         \selectfont
39.112         \advance\dimen@-1.15ex
39.113       \else
39.114         \fontsize\ssf@size\z@
39.115         \selectfont
39.116         \advance\dimen@-1.5ex
39.117       \fi
39.118       \math@fontsfalse\raise\dimen@\hbox{#1}}}%
39.119 \else %<<<<<<
39.120   \let\sptextfont\rm
39.121   \newcommand\gl@sptext[1]{%
39.122     {\setbox\z@\hbox{8}\dimen@ \ht\z@
39.123       \edef\@tempa{\def\noexpand\@tempc{#1}%
39.124         \lowercase{\def\noexpand\@tempb{#1}}}\@tempa
39.125       \ifx\@tempb\@tempc
39.126         \advance\dimen@-0.75ex
39.127         \raise\dimen@\hbox{$\scriptstyle\sptextfont#1$}%
39.128       \else
39.129         \advance\dimen@-0.8ex
39.130         \raise\dimen@\hbox{$\scriptscriptstyle\sptextfont#1$}%
39.131       \fi}}
39.132 \fi %<<<<<<
```

Now, lowercase small caps. First, we test if there are actual small caps for the current font. If not, faked small caps are used. `\msc` tries a slightly larger font. Javier B. wrote: “The `\selectfont` in `\gl@lsc` could seem redundant, but it’s not”. I cannot see how it can’t be redundant (it is the last thing executed by `\scshape`), but I keep it.

```
39.133 \ifcsLaTeXe %<<<<<<
39.134   \addto\textgalician{%
39.135     \babel@save\lsc
39.136     \def\lsc{\protect\gl@lsc}
39.137     \babel@save\msc
39.138     \def\msc{\protect\gl@msc}}
39.139
39.140     \def\gl@@msc{\expandafter\@tempdima\f@size pt \divide\@tempdima by 200 \multiply\@tem
39.141       \edef\f@size{\strip@pt\@tempdima}\selectfont}
39.142     \def\gl@msc{\let\gl@do@msc\gl@@msc\lsc}
39.143     \let\gl@do@msc\relax
39.144
39.145   \def\gl@lsc#1{%
39.146     \leavevmode
39.147     \hbox{\gl@do@msc\scshape\selectfont
39.148       \expandafter\ifx\csname\f@encoding/\f@family/\f@series
39.149         /n/\f@size\expandafter\endcsname
39.150         \csname\curr@fontshape/\f@size\endcsname
39.151         \csname S@f@size\endcsname
```

```

39.152      \fontsize\sf@size\z@\selectfont
39.153      \PackageInfo{galician}{Replacing undefined sc font\MessageBreak
39.154                          shape by faked small caps}%
39.155      \MakeUppercase{#1}%
39.156      \else
39.157      \MakeLowercase{#1}%
39.158      \fi}\let\gl@do@msc\relax}
39.159 \fi      %<<<<<<

```

The quoting environment. This part is not available in Plain, hence the test.

Overriding the default `\everypar` is a bit tricky.

```

39.160 \newif\ifgl@listquot
39.161
39.162 \ifcsname plain\else %<<<<<<
39.163   \csname newtoks\endcsname\gl@quottoks
39.164   \csname newcount\endcsname\gl@quotdepth
39.165
39.166   \ifx\quoting\c@undefined\def\next{\let\next\relax\newenvironment}
39.167   \else\def\next{\PackageInfo{galician}{Redefining quoting}\let\next\relax\renewenvironment}
39.168   \fi
39.169 \next{quoting}
39.170 {\leavevmode
39.171   \advance\gl@quotdepth1
39.172   \csname lquot\romannumeral\gl@quotdepth\endcsname%
39.173   \ifnum\gl@quotdepth=\@ne
39.174     \gl@listquotfalse
39.175     \let\gl@quotpar\everypar
39.176     \let\everypar\gl@quottoks
39.177     \everypar\expandafter{\the\gl@quotpar}%
39.178     \gl@quotpar{\the\everypar
39.179       \ifgl@listquot\global\gl@listquotfalse\else\gl@quotcont\fi}%
39.180   \fi
39.181   \toks@\expandafter{\gl@quotcont}%
39.182   \edef\gl@quotcont{\the\toks@
39.183     \expandafter\noexpand
39.184     \csname rquot\romannumeral\gl@quotdepth\endcsname}}
39.185   {\csname rquot\romannumeral\gl@quotdepth\endcsname}
39.186
39.187   \def\lquoti{\guillemotleft{}}
39.188   \def\rquoti{\guillemotright{}}
39.189   \def\lquotii{'{'}
39.190   \def\rquotii{'{'}
39.191   \def\lquotiii{'{'}}
39.192   \def\rquotiii{'{'}}
39.193
39.194   \let\gl@quotcont\@empty

```

If there is a margin par inside quoting, we don't add the quotes. `\gl@listquot` stores the quotes to be used before item labels; otherwise they could appear after the labels.

```

39.195   \addto\@marginparreset{\let\gl@quotcont\@empty}
39.196
39.197   \def\gl@listquot{%
39.198     \csname rquot\romannumeral\gl@quotdepth\endcsname
39.199     \global\gl@listquottrue}
39.200 \fi      %<<<<<<

```

Now, the `\frenchspacing`, followed by `\dots` and `\%`. Instead of redefining `\ldots` and `\cdots`, we redefine `\ldotp` and `\cdotp`, so that this is compatible with `amsmath`. In `LaTeX` we also redefine `\textellipsis`, and for plain or other we redefine `\dots`.

```

39.201 \addto\textgalician{\bbl@frenchspacing}
39.202 \addto\notextgalician{\bbl@nonfrenchspacing}
39.203

```

```

39.204 \mathchardef\gl@cdotp="0201
39.205 \ifcs@LaTeXe %<<<<<<
39.206 \addto\textgalician{%
39.207 \babel@save\textellipsis
39.208 \babel@save\ldotp
39.209 \babel@save\cdotp%
39.210 \def\textellipsis{\hbox{...}\spacefactor\sfcode'.'}%
39.211 \mathchardef\ldotp="013A%
39.212 \mathchardef\cdotp="0201%
39.213 }
39.214 \else %<<<<<<
39.215 \addto\textgalician{%
39.216 \babel@save\dots
39.217 \babel@save\ldotp
39.218 \babel@save\cdotp
39.219 \mathchardef\ldotp="013A%
39.220 \mathchardef\cdotp="0201%
39.221 \def\dots{\ifmmode\ldots\else...\spacefactor\sfcode'.'}\fi}%
39.222 }
39.223 \fi %<<<<<<
39.224
39.225 \ifcs@LaTeXe %<<<<<<
39.226 \addto\textgalician{%
39.227 \let\percentsign\%%
39.228 \babel@save\%%
39.229 \def\%{\unskip\,\percentsign{}}}%
39.230 \else
39.231 \addto\textgalician{%
39.232 \let\percentsign\%%
39.233 \babel@save\%%
39.234 \def\%{\unskip\ifmmode\,\else$\m@th\,$\fi\percentsign{}}}%
39.235 \fi

```

We follow with the math group. It's not easy to add an accent in an operator. The difficulty is that we must avoid using text (that is, `\mbox`) because we have no control on font and size, and at time we should access `\i`, which is a text command forbidden in math mode. `\dotlessi` must be converted to uppercase if necessary in $\text{\LaTeX 2}_{\epsilon}$. There are two versions, depending on the format.

```

39.236 \addto\mathgalician{%
39.237 \babel@save\dotlessi
39.238 \def\dotlessi{\protect\gl@dotlessi}}
39.239
39.240 \let\nomathgalician\relax %% Unused, but called
39.241
39.242 \ifcs@LaTeXe %<<<<<<
39.243 \def\gl@texti{\i}
39.244 \addto\@uclclist{\dotlessi\gl@texti}
39.245 \fi %<<<<<<
39.246
39.247 \ifcs@LaTeXe %<<<<<<
39.248 \def\gl@dotlessi{%
39.249 \ifmmode
39.250 {\ifnum\mathgroup=\m@ne
39.251 \imath
39.252 \else
39.253 \count@\escapechar \escapechar=\m@ne
39.254 \expandafter\expandafter\expandafter
39.255 \split@name\expandafter\string\the\textfont\mathgroup\@nil
39.256 \escapechar=\count@
39.257 \@ifundefined{f@encoding\string\i}%
39.258 {\edef\f@encoding{\string?}}{}}%
39.259 \expandafter\count@\the\csname\f@encoding\string\i\endcsname
39.260 \advance\count@"7000

```

```

39.261      \mathchar\count@
39.262      \fi}%
39.263      \else
39.264      \i
39.265      \fi}
39.266 \else      %<<<<<<
39.267 \def\gl@dotlessi{%
39.268 \ifmmode
39.269 \mathchar"7010
39.270 \else
39.271 \i
39.272 \fi}
39.273 \fi      %<<<<<<

```

The switches for accents and spaces in math.

```

39.274 \def\accentedoperators{%
39.275 \def\gl@op@ac##1{\acute{##1}}%
39.276 \def\gl@op@i{\acute{\dotlessi}}}%
39.277 \def\unaccentedoperators{%
39.278 \def\gl@op@ac##1{##1}%
39.279 \def\gl@op@i{i}}
39.280 \accentedoperators
39.281
39.282 \def\spacedoperators{\let\gl@op@sp\,}
39.283 \def\unspacedoperators{\let\gl@op@sp@empty}
39.284 \unspacedoperators

```

The operators are stored in `\gl@operators`, which in turn is included in the math group. Since `\operator@font` is defined in L^AT_EX 2_ε only, we need to define them in the plain variant.

```

39.285 \addto\mathgalian{\%
39.286 \gl@operators}
39.287
39.288 \if@LaTeX\else %<<<<<<
39.289 \let\operator@font\rm
39.290 \def\@empty{}
39.291 \fi      %<<<<<<
39.292
39.293 \def\gl@operators{%
39.294 \babel@save\lim \def\lim{\mathop{\operator@font l\protect\gl@op@i m}}%
39.295 \babel@save\limsup \def\limsup{\mathop{\operator@font l\gl@op@i m\sup}}%
39.296 \babel@save\liminf \def\liminf{\mathop{\operator@font l\gl@op@i m\inf}}%
39.297 \babel@save\max \def\max{\mathop{\operator@font m\gl@op@ac ax}}%
39.298 \babel@save\inf \def\inf{\mathop{\operator@font \protect\gl@op@i nf}}%
39.299 \babel@save\min \def\min{\mathop{\operator@font m\protect\gl@op@i n}}%
39.300 \babel@save\bmod
39.301 \def\bmod{%
39.302 \nonscript\mskip-\medmuskip\mkern5mu%
39.303 \mathbin{\operator@font m\gl@op@ac od}\penalty900\mkern5mu%
39.304 \nonscript\mskip-\medmuskip}%
39.305 \babel@save\pmod
39.306 \def\pmod##1{%
39.307 \allowbreak\mkern18mu{\operator@font m\gl@op@ac od}\,\,\,##1}%
39.308 \def\gl@a##1 {%
39.309 \gl@delayed
39.310 \if^##1^then % is it empty? do nothing and continue
39.311 \gl@a
39.312 \else
39.313 \gl@delayed
39.314 \if&##1then % is it &? do nothing and finish
39.315 \else
39.316 \begingroup
39.317 \let\,\@empty % \, is ignored when def'ing the macro name
39.318 \let\acute\@firstofone % same

```

```

39.319 \edef\gl@b{\expandafter\noexpand\csname##1\endcsname}%
39.320 \def\,{\noexpand\gl@op@sp}%
39.321 \def\acute####1{%
39.322 \if i####1%
39.323 \noexpand\gl@op@i
39.324 \else
39.325 \noexpand\gl@op@ac####1%
39.326 \fi}%
39.327 \edef\gl@a{\endgroup
39.328 \noexpand\babel@save\expandafter\noexpand\gl@b
39.329 \def\expandafter\noexpand\gl@b{%
39.330 \mathop{\noexpand\operator@font##1}\nolimits}}%
39.331 \gl@a % It restores itself
39.332 \gl@a
39.333 \fi
39.334 \fi}%
39.335 \let\gl@b\galicianoperators
39.336 \addto\gl@b{ }%
39.337 \expandafter\gl@a\gl@b sen tx cosec arc\,sen arc\,cos arc\,tx senh & %\, will be set to \gl
39.338 %
39.339 \babel@save\sin \let\sin\sen
39.340 \babel@save\arcsin \let\arcsin\arcsen
39.341 \babel@save\sinh \let\sinh\senh
39.342 }
39.343
39.344 \def\galicianoperators{cotx txh}

```

Now comes the text shorthands. They are grouped in `\shorthandsgalician` and this style performs some operations before the babel shorthands are called. The goals are to allow expression like $a^{\{x\}}$ and to deactivate the shorthands making them of category ‘other’. After providing a `\’i` shorthand, the new macros are defined.

```

39.345 \DeclareTextCompositeCommand{\’}{OT1}{i}{\@tabacckludge\’{i}}
39.346
39.347 \def\gl@set@shorthand#1{%
39.348 \expandafter\edef\csname gl@savecat\string#1\endcsname
39.349 {\the\catcode\’#1}%
39.350 \initiate@active@char{#1}%
39.351 \catcode\’#1=\csname gl@savecat\string#1\endcsname\relax
39.352 \expandafter\let\csname gl@math\string#1\expandafter\endcsname
39.353 \csname normal@char\string#1\endcsname}
39.354
39.355 \def\gl@use@shorthand{%
39.356 \gl@delayed
39.357 \ifx\thepage\relax\then
39.358 \string
39.359 \else{%
39.360 \gl@delayed
39.361 \ifx\protect\@unexpandable@protect\then
39.362 \noexpand
39.363 \else
39.364 \gl@use@sh
39.365 \fi}%
39.366 \fi}
39.367
39.368 \def\gl@text@sh#1{\csname active@char\string#1\endcsname}
39.369 \def\gl@math@sh#1{\csname gl@math\string#1\endcsname}
39.370
39.371 \def\gl@use@sh{%
39.372 \gl@delayed
39.373 \if@safe@actives\then
39.374 \string
39.375 \else{%

```



```

39.376 \gl@delayed
39.377 \ifmode\then
39.378 \gl@math@sh
39.379 \else
39.380 \gl@text@sh
39.381 \fi}%
39.382 \fi}
39.383
39.384 \gdef\gl@activate#1{%
39.385 \begingroup
39.386 \lccode'\~='#1
39.387 \lowercase{%
39.388 \endgroup
39.389 \def~{\gl@use@shorthand~}}
39.390
39.391 \def\galiciandeactivate#1{%
39.392 \@tfor\@tempa:=#1\do{\expandafter\gl@spdeactivate\@tempa}}
39.393
39.394 \def\gl@spdeactivate#1{%
39.395 \if.#1%
39.396 \mathcode'\.\=\gl@period@code
39.397 \else
39.398 \begingroup
39.399 \lccode'\~='#1
39.400 \lowercase{%
39.401 \endgroup
39.402 \expandafter\let\expandafter~%
39.403 \csname normal@char\string#1\endcsname}%
39.404 \catcode'#1\csname gl@savecat\string#1\endcsname\relax
39.405 \fi}
39.406
39.407 \def\gl@reviveshorthands{%
39.408 \gl@restore{"}\gl@restore{~}%
39.409 \gl@restore{<}\gl@restore{>}%
39.410 \gl@quoting}
39.411
39.412 \def\gl@restore#1{%
39.413 \catcode'#1=\active
39.414 \begingroup
39.415 \lccode'\~='#1
39.416 \lowercase{%
39.417 \endgroup
39.418 \bbl@deactivate{~}}

```

But galician allows two category codes for ', so both should be taken into account in \bbl@pr@m@s.

```

39.419 \begingroup
39.420 \catcode'\'=12
39.421 \lccode'\~=' \lccode'\=' '
39.422 \lowercase{%
39.423 \gdef\bbl@pr@m@s{%
39.424 \gl@delayed
39.425 \ifx~\@let@token\then
39.426 \pr@@@s
39.427 \else
39.428 {\gl@delayed
39.429 \ifx'\@let@token\then
39.430 \pr@@@s
39.431 \else
39.432 {\gl@delayed
39.433 \ifx~\@let@token\then
39.434 \pr@@@t
39.435 \else

```

```

39.436         \egroup
39.437     \fi}%
39.438     \fi}%
39.439 \fi}}
39.440 \endgroup

39.441 \expandafter\ifx\csname @tabacckludge\endcsname\relax
39.442     \let\gl@tak\a
39.443 \else
39.444     \let\gl@tak\@tabacckludge
39.445 \fi
39.446
39.447 \ifcs@LaTeXe %<<<<<
39.448     \def\@tabacckludge#1{\expandafter\gl@tak\string#1}
39.449     \let\a\@tabacckludge
39.450 \else\ifcs@plain %<<<<<
39.451     \def\@tabacckludge#1{\csname\string#1\endcsname}
39.452 \else %<<<<<
39.453     \def\@tabacckludge#1{\csname a\string#1\endcsname}
39.454 \fi\fi %<<<<<
39.455
39.456 \expandafter\ifx\csname add@accent\endcsname\relax
39.457     \def\add@accent#1#2{\accent#1 #2}
39.458 \fi

```

Instead of redefining `\'`, we redefine the internal macro for the OT1 encoding.

```

39.459 \ifcs@LaTeXe %<<<<<
39.460     \def\gl@accent#1#2#3{%
39.461         \expandafter\@text@composite
39.462         \csname OT1\string#1\endcsname#3\@empty\@text@composite
39.463         {\bbl@allowhyphens\add@accent{#2}{#3}\bbl@allowhyphens
39.464         \setbox\@tempboxa\hbox{#3}%
39.465         \global\mathchardef\accent@spacefactor\spacefactor}%
39.466         \spacefactor\accent@spacefactor}}
39.467 \else %<<<<<
39.468     \def\gl@accent#1#2#3{%
39.469         \bbl@allowhyphens\add@accent{#2}{#3}\bbl@allowhyphens
39.470         \spacefactor\sffcode'#3 }
39.471 \fi %<<<<<

```

The shorthands are activated in the aux file. Now, we begin the shorthands group.

```

39.472 \addto\shorthandsgalician{\languageshorthands{galician}}
39.473 \let\noshorthandsgalician\relax

```

First, decimal comma.

```

39.474 \def\galiciandecimal#1{\def\gl@decimal{{#1}}}
39.475 \def\decimalcomma{\galiciandecimal{,}}
39.476 \def\decimalpoint{\galiciandecimal{.}}
39.477 \decimalcomma
39.478
39.479 \gl@set@shorthand{.}
39.480
39.481 \@namedef{gl@math\string.}{%
39.482     \@ifnextchar\egroup
39.483     {\mathchar\gl@period@code\relax}%
39.484     {\gl@text@sh.}}
39.485
39.486 \declare@shorthand{system}{.}{\mathchar\gl@period@code\relax}

39.487 \addto\shorthandsgalician{%
39.488     \mathchardef\gl@period@code\the\mathcode'\.%
39.489     \babel@savevariable{\mathcode'\.}%
39.490     \mathcode'\.= "8000 %
39.491     \gl@activate{.}}

```

```

39.492
39.493 \AtBeginDocument{%
39.494   \catcode'\.=12
39.495   \if@filesw
39.496     \immediate\write\@mainaux{%
39.497     \string\catcode'\string\.=12}%
39.498   \fi}
39.499
39.500 \declare@shorthand{galician}{.1}{\gl@decimal1}
39.501 \declare@shorthand{galician}{.2}{\gl@decimal2}
39.502 \declare@shorthand{galician}{.3}{\gl@decimal3}
39.503 \declare@shorthand{galician}{.4}{\gl@decimal4}
39.504 \declare@shorthand{galician}{.5}{\gl@decimal5}
39.505 \declare@shorthand{galician}{.6}{\gl@decimal6}
39.506 \declare@shorthand{galician}{.7}{\gl@decimal7}
39.507 \declare@shorthand{galician}{.8}{\gl@decimal8}
39.508 \declare@shorthand{galician}{.9}{\gl@decimal9}
39.509 \declare@shorthand{galician}{.0}{\gl@decimal0}

Now accents and tools
39.510 \gl@set@shorthand{"}
39.511 \def\gl@umlaut#1{%
39.512   \bbl@allowhyphens\add@accent{127}#1\bbl@allowhyphens
39.513   \spacefactor\sfcode'#1 }

We override the default " of babel, intended for german.
39.514 \ifes@LaTeXe %<<<<<
39.515   \addto\shorthandsgalician{%
39.516     \gl@activate{"}%
39.517     \gl@activate{~}%
39.518     \babel@save\bbl@umlauta
39.519     \let\bbl@umlauta\gl@umlaut
39.520     \expandafter\babel@save\csname OT1\string\~\endcsname
39.521     \expandafter\def\csname OT1\string\~\endcsname{\gl@accent\~{126}}%
39.522     \expandafter\babel@save\csname OT1\string\' \endcsname
39.523     \expandafter\def\csname OT1\string\' \endcsname{\gl@accent\'{19}}%
39.524 \else %<<<<<
39.525   \addto\shorthandsgalician{%
39.526     \gl@activate{"}%
39.527     \gl@activate{~}%
39.528     \babel@save\bbl@umlauta
39.529     \let\bbl@umlauta\gl@umlaut
39.530     \babel@save\~%
39.531     \def\~{\gl@accent\~{126}}%
39.532     \babel@save\'%
39.533     \def\'#1{\if#1i\gl@accent\'{19}\i\else\gl@accent\'{19}\fi}}
39.534 \fi %<<<<<

39.535 \declare@shorthand{galician}{a}{\protect\gl@sptext{a}}
39.536 \declare@shorthand{galician}{A}{\protect\gl@sptext{A}}
39.537 \declare@shorthand{galician}{o}{\protect\gl@sptext{o}}
39.538 \declare@shorthand{galician}{0}{\protect\gl@sptext{0}}
39.539
39.540 \declare@shorthand{galician}{u}{\u}
39.541 \declare@shorthand{galician}{U}{\U}
39.542 \declare@shorthand{galician}{i}{\i}
39.543 \declare@shorthand{galician}{I}{\I}
39.544
39.545 \declare@shorthand{galician}{<}{\begin{quoting}}
39.546 \declare@shorthand{galician}{>}{\end{quoting}}
39.547 \declare@shorthand{galician}{-}{\bbl@allowhyphens-\bbl@allowhyphens}
39.548 \declare@shorthand{galician}{=}{=}
39.549 {\bbl@allowhyphens\char\hyphenchar\font\hskip\z@skip}
39.550 \declare@shorthand{galician}{~}{~}

```

```

39.551 {\bbl@allowhyphens\discretionary{\char\hyphenchar\font}%
39.552 {\char\hyphenchar\font}{\char\hyphenchar\font}\bbl@allowhyphens}
39.553 \declare@shorthand{galician}{\r}
39.554 {\bbl@allowhyphens\discretionary{\char\hyphenchar\font}%
39.555 {}{r}\bbl@allowhyphens}
39.556 \declare@shorthand{galician}{\R}
39.557 {\bbl@allowhyphens\discretionary{\char\hyphenchar\font}%
39.558 {}{R}\bbl@allowhyphens}
39.559 \declare@shorthand{galician}{""}{\hskip\z@skip}
39.560 \declare@shorthand{galician}{"/}
39.561 {\setbox\z@\hbox{}}%
39.562 \dimen@ht\z@
39.563 \advance\dimen@-1ex
39.564 \advance\dimen@\dp\z@
39.565 \dimen@.31\dimen@
39.566 \advance\dimen@-\dp\z@
39.567 \ifdim\dimen@>0pt
39.568 \kern.01em\lower\dimen@\box\z@\kern.03em
39.569 \else
39.570 \box\z@
39.571 \fi}
39.572 \declare@shorthand{galician}{"?}
39.573 {\setbox\z@\hbox{?'}}%
39.574 \leavevmode\raise\dp\z@\box\z@}
39.575 \declare@shorthand{galician}{"!}
39.576 {\setbox\z@\hbox{!'}}%
39.577 \leavevmode\raise\dp\z@\box\z@}
39.578
39.579 \gl@set@shorthand{~}
39.580 \declare@shorthand{galician}{~n}{\~n}
39.581 \declare@shorthand{galician}{~N}{\~N}
39.582 \declare@shorthand{galician}{~-}{~}%
39.583 \leavevmode
39.584 \bgroup
39.585 \let\@sptoken\gl@dashes % This assignation changes the
39.586 \@ifnextchar-% \@ifnextchar behaviour
39.587 {\gl@dashes}%
39.588 {\hbox{\char\hyphenchar\font}\egroup}}
39.589 \def\gl@dashes-{%
39.590 \@ifnextchar-%
39.591 {\bbl@allowhyphens\hbox{---}\bbl@allowhyphens\egroup\@gobble}%
39.592 {\bbl@allowhyphens\hbox{--}\bbl@allowhyphens\egroup}}
39.593
39.594 \def\deactivatetilden{%
39.595 \expandafter\let\csname galician@sh@\string~@n\endcsname\relax
39.596 \expandafter\let\csname galician@sh@\string~@N\endcsname\relax}

```

The shorthands for quoting.

```

39.597 \expandafter\ifx\csname XML@catcodes\endcsname\relax
39.598 \addto\gl@select{%
39.599 \catcode'\<\active\catcode'\>=\active
39.600 \gl@quoting}
39.601
39.602 \gl@set@shorthand{<}
39.603 \gl@set@shorthand{>}
39.604
39.605 \declare@shorthand{system}{<}{\csname normal@char\string<\endcsname}
39.606 \declare@shorthand{system}{>}{\csname normal@char\string>\endcsname}
39.607
39.608 \addto\shorthandsgalician{%
39.609 \gl@activate{<}%
39.610 \gl@activate{>}}
39.611 \ifcs@LaTeXe %<<<<<<

```

```

39.612 \AtBeginDocument{%
39.613 \gl@quoting
39.614 \if@filesw
39.615 \immediate\write\@mainaux{\string\gl@quoting}%
39.616 \fi}%
39.617 \fi %<<<<<
39.618
39.619 \def\activatequoting{%
39.620 \catcode'\>=\active \catcode'\<=\active
39.621 \let\gl@quoting\activatequoting}
39.622 \def\deactivatequoting{%
39.623 \catcode'\>=12 \catcode'\<=12
39.624 \let\gl@quoting\deactivatequoting}
39.625
39.626 \declare@shorthand{galician}{<<}{\guillemotleft{}}
39.627 \declare@shorthand{galician}{>>}{\guillemotright{}}
39.628 \fi
39.629
39.630 \let\gl@quoting\relax
39.631 \let\deactivatequoting\relax
39.632 \let\activatequoting\relax

```

The acute accents are stored in a macro. If `activeacute` was set as an option it's executed. If not is not deleted for a possible later use in the `cfg` file. In non L^AT_EX 2_ε formats is always executed.

```

39.633 \def\gl@activeacute{%
39.634 \gl@set@shorthand{'}%
39.635 \addto\shorthandsgalician{\gl@activate{'}}%
39.636 \addto\gl@reviveshorthands{\gl@restore{'}}%
39.637 \addto\gl@select{\catcode'\>=\active}%
39.638 \declare@shorthand{galician}{'a}{\@tabacckludge'a}%
39.639 \declare@shorthand{galician}{'A}{\@tabacckludge'A}%
39.640 \declare@shorthand{galician}{'e}{\@tabacckludge'e}%
39.641 \declare@shorthand{galician}{'E}{\@tabacckludge'E}%
39.642 \declare@shorthand{galician}{'i}{\@tabacckludge'i}%
39.643 \declare@shorthand{galician}{'I}{\@tabacckludge'I}%
39.644 \declare@shorthand{galician}{'o}{\@tabacckludge'o}%
39.645 \declare@shorthand{galician}{'O}{\@tabacckludge'O}%
39.646 \declare@shorthand{galician}{'u}{\@tabacckludge'u}%
39.647 \declare@shorthand{galician}{'U}{\@tabacckludge'U}%
39.648 \declare@shorthand{galician}{'n}{\~n}%
39.649 \declare@shorthand{galician}{'N}{\~N}%
39.650 \declare@shorthand{galician}{' '}{\textquotedblright}%
39.651 \let\gl@activeacute\relax}
39.652
39.653 \ifes@LaTeXe %<<<<<
39.654 \@ifpackagewith{babel}{activeacute}{\gl@activeacute}{%
39.655 \else %<<<<<
39.656 \gl@activeacute
39.657 \fi %<<<<<%

```

And the customization. By default these macros only store the values and do nothing.

```

39.658 \def\gl@enumerate#1#2#3#4{%
39.659 \def\gl@enum{{#1}{#2}{#3}{#4}}}
39.660
39.661 \def\gl@itemize#1#2#3#4{%
39.662 \def\gl@item{{#1}{#2}{#3}{#4}}}

```

The part formerly in the `.lld` file comes here. It performs layout adaptation of L^AT_EX to “orthodox” Galician rules.

```

39.663 \ifes@LaTeXe %<<<<<
39.664
39.665 \gl@enumerate{1.}{a)}{1)}{a$'$}

```

```

39.666 \def\galiciandashitems{\gl@itemize{---}{---}{---}{---}}
39.667 \def\galiciansymbitems{%
39.668   \gl@itemize
39.669     {\leavevmode\hbox to 1.2ex
39.670       {\hss\vrule height .9ex width .7ex depth -.2ex\hss}}%
39.671     {\textbullet}%
39.672     {\$ \m@th \circ$}%
39.673     {\$ \m@th \diamond$}}
39.674 \def\galiciansignitems{%
39.675   \gl@itemize
39.676     {\textbullet}%
39.677     {\$ \m@th \circ$}%
39.678     {\$ \m@th \diamond$}%
39.679     {\$ \m@th \triangleright$}}
39.680 \galiciansymbitems
39.681
39.682 \def\gl@enumdef#1#2#3\@@{%
39.683   \if#21%
39.684     \@namedef{theenum#1}{\arabic{enum#1}}%
39.685   \else\if#2a%
39.686     \@namedef{theenum#1}{\emph{\alph{enum#1}}}%
39.687   \else\if#2A%
39.688     \@namedef{theenum#1}{\Alph{enum#1}}%
39.689   \else\if#2i%
39.690     \@namedef{theenum#1}{\roman{enum#1}}%
39.691   \else\if#2I%
39.692     \@namedef{theenum#1}{\Roman{enum#1}}%
39.693   \else\if#2o%
39.694     \@namedef{theenum#1}{\arabic{enum#1}\protect\gl@sptext{o}}%
39.695   \fi\fi\fi\fi\fi\fi
39.696   \toks@{\expandafter{\csname theenum#1\endcsname}}
39.697   \expandafter\edef\csname labelenum#1\endcsname
39.698     {\noexpand\gl@listquot\the\toks@#3}}
39.699
39.700 \addto\layoutgalician{%
39.701   \def\gl@enumerate##1##2##3##4{%
39.702     \gl@enumdef{i}##1\@empty\@empty\@@
39.703     \gl@enumdef{ii}##2\@empty\@empty\@@
39.704     \gl@enumdef{iii}##3\@empty\@empty\@@
39.705     \gl@enumdef{iv}##4\@empty\@empty\@@}%
39.706   \def\gl@itemize##1##2##3##4{%
39.707     \def\labelitemi{\gl@listquot##1}%
39.708     \def\labelitemii{\gl@listquot##2}%
39.709     \def\labelitemiii{\gl@listquot##3}%
39.710     \def\labelitemiv{\gl@listquot##4}}%
39.711   \def\p@enumii{\theenumi}%
39.712   \def\p@enumiii{\theenumi\theenumii}%
39.713   \def\p@enumiv{\p@enumiii\theenumiii}%
39.714   \expandafter\gl@enumerate\gl@enum
39.715   \expandafter\gl@itemize\gl@item
39.716   \DeclareTextCommand{\guillemotleft}{OT1}{%
39.717     \ifmmode\ll
39.718     \else
39.719       \save@sf@q{\penalty\@M
39.720         \leavevmode\hbox{\usefont{U}{lasy}{m}{n}%
39.721           \char40 \kern-0.19em\char40 }}%
39.722     \fi}%
39.723   \DeclareTextCommand{\guillemotright}{OT1}{%
39.724     \ifmmode\gg
39.725     \else
39.726       \save@sf@q{\penalty\@M
39.727         \leavevmode\hbox{\usefont{U}{lasy}{m}{n}%

```

```

39.728      \char41 \kern-0.19em\char41 }}%
39.729      \fi}%
39.730      \def\@fnsymbol##1%
39.731      {\ifcase##1\or*\or**\or***\or****\or
39.732      *****\or*****\else\@ctrerr\fi}%
39.733      \def\@alph##1%
39.734      {\ifcase##1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or
39.735      l\or m\or n\or \~n\or o\or p\or q\or r\or s\or t\or u\or v\or
39.736      x\or z\else\@ctrerr\fi}%
39.737      \def\@Alph##1%
39.738      {\ifcase##1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or
39.739      L\or M\or N\or \~N\or O\or P\or Q\or R\or S\or T\or U\or V\or
39.740      X\or Z\else\@ctrerr\fi}%
39.741      \let\@afterindentfalse\@afterindenttrue
39.742      \@afterindenttrue
39.743      \def\@secntformat##1{\csname the##1\endcsname.\quad}%
39.744      \def\@numberline##1{\hb@xt@{\@tempdima{##1}\if&##1&\else.\fi\hfil}}%
39.745      \def\@roman##1{\protect\gl@roman{\number##1}}%
39.746      \def\gl@roman##1{\protect\gl@msc{\romannumeral##1}}%
39.747      \def\gl@romanindex##1##2{##1{\protect\gl@msc{##2}}}}

```

We need to execute the following code when babel has been run, in order to see if galician is the main language.

```

39.748 \AtEndOfPackage{%
39.749   \let\gl@activeacute\@undefined
39.750   \def\bbl@tempa{galician}%
39.751   \ifx\bbl@main@language\bbl@tempa
39.752     \AtBeginDocument{\layoutgalician}%
39.753     \addto\gl@select{%
39.754       \@ifstar{\let\layoutgalician\relax}%
39.755       {\layoutgalician\let\layoutgalician\relax}}%
39.756   \fi
39.757   \selectgalician}
39.758
39.759 \fi          %<<<<<<

```

After restoring the catcode of ~ and setting the minimal values for hyphenation, the .ldf is finished.

```

39.760 \gl@saveditcatcodes
39.761
39.762 \providehyphenmins{\CurrentOption}{\tw@\tw@}
39.763
39.764 \ifes@LaTeXe %<<<<<<
39.765   \ldf@finish{galician}
39.766 \else %<<<<<<
39.767   \gl@select
39.768   \ldf@finish{galician}
39.769   \csname activatequoting\endcsname
39.770 \fi          %<<<<<<
39.771
39.772 \code>

```

That's all in the main file. Now the file with custom-bib macros.

```

39.773 (*bblbst)
39.774 \def\bbland{e}
39.775 \def\bbl@editores{directores}      \def\bbl@eds{dirs.\@}
39.776 \def\bbl@editor{director}         \def\bbl@ed{dir.\@}
39.777 \def\bbl@edby{dirixido por}
39.778 \def\bbl@edition{edici\~on}       \def\bbl@edn{ed.\@}
39.779 \def\bbl@etal{e outros}
39.780 \def\bbl@volume{volumen}          \def\bbl@vol{vol.\@}
39.781 \def\bbl@of{de}
39.782 \def\bbl@number{n\~umero}          \def\bbl@no{n\~umero}
39.783 \def\bbl@in{en}

```

```

39.784 \def\bbllpages{p\'axinas}      \def\bbllpp{p\'axs.\@}
39.785 \def\bbllpage{p\'axina}        \def\bbllp{p\'ax.\@}
39.786 \def\bbllchapter{cap\'itulo}   \def\bbllchap{cap.\@}
39.787 \def\bblltechreport{informe t\'ecnico}
39.788 \def\bblltechrep{inf.\@ t\'ec.\@}
39.789 \def\bbllmthesis{proxecto de fin de carreira}
39.790 \def\bbllphdthesis{tesis doutoral}
39.791 \def\bbllfirst {primeira}      \def\bbllfirsto {1\sptext{a}}
39.792 \def\bbllsecond{segunda}       \def\bbllsecondo {2\sptext{a}}
39.793 \def\bbllthird {terceira}      \def\bbllthirdo {3\sptext{a}}
39.794 \def\bbllfourth{cuarta}        \def\bbllfourtho {4\sptext{a}}
39.795 \def\bbllfifth {quinta}        \def\bbllfiftho {5\sptext{a}}
39.796 \def\bbllth{\sptext{a}}
39.797 \let\bbllst\bbllth \let\bbllnd\bbllth \let\bbllrd\bbllth
39.798 \def\bblljan{xaneiro} \def\bbllfeb{febreiro} \def\bbllmar{marzo}
39.799 \def\bbllapr{abril} \def\bbllmay{maio} \def\bblljun{xu~no}
39.800 \def\bblljul{xullo} \def\bbllaug{agosto} \def\bbllsep{setembro}
39.801 \def\bbloct{outubro}\def\bbllnov{novembro}\def\bblldec{decembro}
39.802 \let\bbllst\bbllth

```

The galician option writes a macro in the page field of *MakeIndex* in entries with medium caps number, and they are rejected. This program is a preprocessor which moves this macro to the entry field.

```

39.803 (*indexgl)
39.804 \makeatletter
39.805
39.806 \newcount\gl@converted
39.807 \newcount\gl@processed
39.808
39.809 \def\gl@encap{'\|}
39.810 \def\gl@openrange{'\({}
39.811 \def\gl@closrange{'\)}
39.812
39.813 \def\gl@split@file#1.#2\@{\#1}
39.814 \def\gl@split@ext#1.#2\@{\#2}
39.815
39.816 \typein[\answer]{^^JArchivo que convertir^^J%
39.817   (extension por omission .idx):}
39.818
39.819 \@expandtwoargs\in@{.}{\answer}
39.820 \ifin@
39.821   \edef\gl@input@file{\expandafter\gl@split@file\answer\@}
39.822   \edef\gl@input@ext{\expandafter\gl@split@ext\answer\@}
39.823 \else
39.824   \edef\gl@input@file{\answer}
39.825   \def\gl@input@ext{idx}
39.826 \fi
39.827
39.828 \typein[\answer]{^^JArquivo de destino^^J%
39.829   (arquivo por omission: \gl@input@file.eix,^^J%
39.830   extension por omission .eix):}
39.831 \ifx\answer\@empty
39.832   \edef\gl@output{\gl@input@file.eix}
39.833 \else
39.834   \@expandtwoargs\in@{.}{\answer}
39.835   \ifin@
39.836     \edef\gl@output{\answer}
39.837   \else
39.838     \edef\gl@output{\answer.eix}
39.839   \fi
39.840 \fi
39.841
39.842 \typein[\answer]{%

```



```

39.843 ^^J?Uouse algun esquema especial de controles^^J%
39.844 de MakeIndex para encap, open_range ou close_range?^^J%
39.845 [s/n] (n por omision)}
39.846
39.847 \if s\answer
39.848 \typein[\answer]{^^JCaracter para 'encap'^^J%
39.849 (\string| por omision)}
39.850 \ifx\answer@empty\else
39.851 \edef\gl@encap{%
39.852 '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
39.853 \fi
39.854 \typein[\answer]{^^JCaracter para 'open_range'^^J%
39.855 (\string( por omision)}
39.856 \ifx\answer@empty\else
39.857 \edef\gl@openrange{%
39.858 '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
39.859 \fi
39.860 \typein[\answer]{^^JCaracter para 'close_range'^^J%
39.861 (\string) por omision)}
39.862 \ifx\answer@empty\else
39.863 \edef\gl@closerange{%
39.864 '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
39.865 \fi
39.866 \fi
39.867
39.868 \newwrite\gl@indexfile
39.869 \immediate\openout\gl@indexfile=\gl@output
39.870
39.871 \newif\ifgl@encapsulated
39.872
39.873 \def\gl@roman#1{\romannumeral#1 }
39.874 \edef\gl@slash{\expandafter\@gobble\string\\}
39.875
39.876 \def\indexentry{%
39.877 \begingroup
39.878 \@sanitize
39.879 \gl@indexentry}
39.880
39.881 \begingroup
39.882
39.883 \catcode'\|=12 \lccode'\|=\gl@encap\relax
39.884 \catcode'\(=12 \lccode'\(=\gl@openrange\relax
39.885 \catcode'\)=12 \lccode'\)=\gl@closerange\relax
39.886
39.887 \lowercase{
39.888 \gdef\gl@indexentry#1{%
39.889 \endgroup
39.890 \advance\gl@processed\@ne
39.891 \gl@encapsulatedfalse
39.892 \gl@bar@idx#1|\@@
39.893 \gl@idxentry}%
39.894 }
39.895
39.896 \lowercase{
39.897 \gdef\gl@idxentry#1{%
39.898 \in@{\gl@roman}{#1}%
39.899 \ifin@
39.900 \advance\gl@converted\@ne
39.901 \immediate\write\gl@indexfile{%
39.902 \string\indexentry{\gl@b|\ifgl@encapsulated\gl@p\fi glromanindex%
39.903 {\ifx\gl@a@empty\else\gl@slash\gl@a\fi}}{#1}}%
39.904 \else

```

```

39.905 \immediate\write\gl@indexfile{%
39.906 \string\indexentry{\gl@b\ifgl@encapsulated|\gl@p\gl@a\fi}{#1}}%
39.907 \fi}
39.908 }
39.909
39.910 \lowercase{
39.911 \gdef\gl@bar@idx#1|#2\@{
39.912 \def\gl@b{#1}\def\gl@a{#2}%
39.913 \ifx\gl@a\@empty\else\gl@encapsulatedtrue\gl@bar@eat#2\fi}
39.914 }
39.915
39.916 \lowercase{
39.917 \gdef\gl@bar@eat#1#2|{\def\gl@p{#1}\def\gl@a{#2}%
39.918 \edef\gl@t{(\ifx\gl@t\gl@p
39.919 \else\edef\gl@t{)}\ifx\gl@t\gl@p
39.920 \else
39.921 \edef\gl@a{\gl@p\gl@a}\let\gl@p\@empty%
39.922 \fi\fi}
39.923 }
39.924
39.925 \endgroup
39.926
39.927 \input \gl@input@file.\gl@input@ext
39.928
39.929 \immediate\closeout\gl@indexfile
39.930
39.931 \typeout{*****}
39.932 \typeout{procesouse: \gl@input@file.\gl@input@ext }
39.933 \typeout{Li'nas lidas: \the\gl@processed}
39.934 \typeout{Li'nas convertidas: \the\gl@converted}
39.935 \typeout{Resultado en: \gl@output}
39.936 \ifnum\gl@converted>\z@
39.937 \typeout{Xenere o 'indice a partir deste arquivo}
39.938 \else
39.939 \typeout{Non se realizou ning'un tipo de conversi'on}
39.940 \typeout{P'odese xenerar o arquivo directamente~^J%
39.941 de \gl@input@file.\gl@input@ext}
39.942 \fi
39.943 \typeout{*****}
39.944 \@@end
39.945 </indexgl>

```

40 The Basque language

The file `basque.dtx`⁴⁵ defines all the language definition macro's for the Basque language.

For this language the characters `~` and `"` are made active. In table 15 an overview is given of their purpose. These active accent characters behave according

<code>" </code>	disable ligature at this position.
<code>"-</code>	an explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>\-</code>	like the old <code>\-</code> , but allowing hyphenation in the rest of the word.
<code>"<</code>	for French left double quotes (similar to <code><<</code>).
<code>"></code>	for French right double quotes (similar to <code>>></code>).
<code>~n</code>	a n with tilde. Works for uppercase too.

Table 15: The extra definitions made by `basque.ldf`

to their original definitions if not followed by one of the characters indicated in that table.

This option includes support for working with extended, 8-bit fonts, if available. Support is based on providing an appropriate definition for the accent macros on entry to the Basque language. This is automatically done by $\text{\LaTeX} 2_{\epsilon}$ or NFSS2. If T1 encoding is chosen, and provided that adequate hyphenation patterns⁴⁶ are available. The easiest way to use the new encoding with $\text{\LaTeX} 2_{\epsilon}$ is to load the package `tlenc` with `\usepackage`. This must be done before loading `babel`.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
40.1 (*code)
40.2 \LdfInit{basque}\captionsbasque
```

When this file is read as an option, i.e. by the `\usepackage` command, `basque` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@basque` to see whether we have to do something here.

```
40.3 \ifx\l@basque\@undefined
40.4 \@nopatterns{Basque}
40.5 \adddialect\l@basque0
40.6 \fi
```

The next step consists of defining commands to switch to (and from) the Basque language.

`\captionsbasque` The macro `\captionsbasque` defines all strings used in the four standard documentclasses provided with \LaTeX .

```
40.7 \addto\captionsbasque{%
40.8 \def\prefacename{Hitzaurrea}%
40.9 \def\refname{Erreferentziak}%
40.10 \def\abstractname{Laburpena}%
40.11 \def\bibname{Bibliografia}%
40.12 \def\chaptername{Kapitulua}%
40.13 \def\appendixname{Eranskina}%
40.14 \def\contentsname{Gaien Aurkibidea}%
40.15 \def\listfigurename{Irudien Zerrenda}%
40.16 \def\listtablename{Taulen Zerrenda}%
40.17 \def\indexname{Kontzeptuen Aurkibidea}%
40.18 \def\figurename{Irudia}%
```

⁴⁵The file described in this section has version number v1.0f and was last revised on 2005/03/29. The original author is Juan M. Aguirregabiria, (`wtpagagj@lg.ehu.es`) and is based on the Spanish file by Julio Sánchez, (`jsanchez@gmv.es`).

⁴⁶One source for such patterns is the archive at `tp.lc.ehu.es` that can be accessed by anonymous FTP or in `http://tp.lc.ehu.es/jma/basque.html`

```

40.19 \def\tablename{Taula}%
40.20 \def\partname{Atala}%
40.21 \def\enclname{Erantsia}%
40.22 \def\ccname{Kopia}%
40.23 \def\headtoname{Nori}%
40.24 \def\pagename{Orria}%
40.25 \def\seenname{Ikusi}%
40.26 \def\alsoname{Ikusi, halaber}%
40.27 \def\proofname{Frogapena}%
40.28 \def\glossaryname{Glosarioa}%
40.29 }%

```

`\datebasque` The macro `\datebasque` redefines the command `\today` to produce Basque

```

40.30 \def\datebasque{%
40.31 \def\today{\number\year.eko\space\ifcase\month\or
40.32 urtarrilaren\or otsailaren\or martxoaren\or apirilaren\or
40.33 maiatzaren\or ekainaren\or uztailaren\or abuztuaren\or
40.34 irailaren\or urriaren\or azaroaren\or
40.35 abenduaren\fi~\number\day}}

```

`\extrasbasque` The macro `\extrasbasque` will perform all the extra definitions needed for the Basque language. The macro `\noextrasbasque` is used to cancel the actions of `\extrasbasque`. For Basque, some characters are made active or are redefined. In particular, the " character and the ~ character receive new meanings. Therefore these characters have to be treated as 'special' characters.

```

40.36 \addto\extrasbasque{\languageshorthands{basque}}
40.37 \initiate@active@char{"}
40.38 \initiate@active@char{~}
40.39 \addto\extrasbasque{%
40.40 \bbl@activate{"}%
40.41 \bbl@activate{~}}

```

Don't forget to turn the shorthands off again.

```

40.42 \addto\noextrasbasque{
40.43 \bbl@deactivate{"}\bbl@deactivate{~}}

```

Apart from the active characters some other macros get a new definition.

Therefore we store the current one to be able to restore them later.

```

40.44 \addto\extrasbasque{%
40.45 \babel@save{"}%
40.46 \babel@save{~}%
40.47 \def{"{\protect\@umlaut}%
40.48 \def~{\protect\@tilde}}

```

`\basquehyphenmins` Basque hyphenation uses `\lefthyphenmin` and `\righthyphenmin` both set to 2.

```

40.49 \providehyphenmins{CurrentOption}{\tw@\tw@}

```

`\dieresia` The original definition of " is stored as `\dieresia`, because the we do not know what is its definition, since it depends on the encoding we are using or on special macros that the user might have loaded. The expansion of the macro might use the \TeX `\accent` primitive using some particular accent that the font provides or might check if a combined accent exists in the font. These two cases happen with respectively OT1 and T1 encodings. For this reason we save the definition of " and use that in the definition of other macros. We do likewise for ' and ~. The present coding of this option file is incorrect in that it can break when the encoding changes. We do not use `\tilde` as the macro name because it is already defined as `\mathaccent`.

```

40.50 \let\dieresia"
40.51 \let\texttilde~

```

`\@umlaut` We check the encoding and if not using T1, we make the accents expand but
`\@tilde` enabling hyphenation beyond the accent. If this is the case, not all break positions
will be found in words that contain accents, but this is a limitation in T_EX. An
unsolved problem here is that the encoding can change at any time. The definitions
below are made in such a way that a change between two 256-char encodings
are supported, but changes between a 128-char and a 256-char encoding are not
properly supported. We check if T1 is in use. If not, we will give a warning and
proceed redefining the accent macros so that T_EX at least finds the breaks that
are not too close to the accent. The warning will only be printed to the log file.

```

40.52 \ifx\DeclareFontShape\@undefined
40.53   \wlog{Warning: You are using an old LaTeX}
40.54   \wlog{Some word breaks will not be found.}
40.55   \def\@umlaut#1{\allowhyphens\dieresia{#1}\allowhyphens}
40.56   \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
40.57 \else
40.58   \edef\bbl@next{T1}
40.59   \ifx\f@encoding\bbl@next
40.60     \let\@umlaut\dieresia
40.61     \let\@tilde\texttilde
40.62   \else
40.63     \wlog{Warning: You are using encoding \f@encoding\space
40.64       instead of T1.}
40.65     \wlog{Some word breaks will not be found.}
40.66     \def\@umlaut#1{\allowhyphens\dieresia{#1}\allowhyphens}
40.67     \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
40.68   \fi
40.69 \fi

```

Now we can define our shorthands: the french quotes,

```

40.70 \declare@shorthand{basque}{<}{%
40.71   \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
40.72 \declare@shorthand{basque}{>}{%
40.73   \textormath{\guillemotright}{\mbox{\guillemotright}}}
ordinals47,
40.74 \declare@shorthand{basque}{''}{%
40.75   \textormath{\textquotedblright}{\sp\bgroup\prim@s'}}
tildes,
40.76 \declare@shorthand{basque}{~n}{\textormath{\~n}{\@tilde n}}
40.77 \declare@shorthand{basque}{~N}{\textormath{\~N}{\@tilde N}}

```

and some additional commands.

The shorthand “-” should be used in places where a word contains an explicit
hyphenation character. According to the Academy of the Basque language, when
a word break occurs at an explicit hyphen it must appear *both* at the end of the
first line *and* at the beginning of the second line.

```

40.78 \declare@shorthand{basque}{-}{%
40.79   \nobreak\discretionary{-}{-}{-}\bbl@allowhyphens}
40.80 \declare@shorthand{basque}{"|}{%
40.81   \textormath{\nobreak\discretionary{-}{-}{\kern.03em}%
40.82     \allowhyphens}}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting
the main language to be switched on at `\begin{document}` and resetting the
category code of `@` to its original value.

```

40.83 \ldf@finish{basque}
40.84 </code>

```

⁴⁷The code for the ordinals was taken from the answer provided by Raymond Chen
(raymond@math.berkeley.edu) to a question by Joseph Gil (yogi@cs.ubc.ca) in `comp.text.tex`.

41 The Romanian language

The file `romanian.dtx`⁴⁸ defines all the language-specific macros for the Romanian language.

For this language currently no special definitions are needed or available.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
41.1 (*code)
41.2 \LdfInit{romanian}\captionsromanian
```

When this file is read as an option, i.e. by the `\usepackage` command, `romanian` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@romanian` to see whether we have to do something here.

```
41.3 \ifx\l@romanian\undefined
41.4     \nopatterns{Romanian}
41.5     \adddialect\l@romanian0\fi
```

The next step consists of defining commands to switch to (and from) the Romanian language.

`\captionsromanian` The macro `\captionsromanian` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
41.6 \ls
41.7 \addto\captionsromanian{%
41.8     \def\prefacename{Prefa\c{t}\u{a}}%
41.9     \def\refname{Bibliografie}%
41.10    \def\abstractname{Rezumat}%
41.11    \def\bibname{Bibliografie}%
41.12    \def\chaptername{Capitolul}%
41.13    \def\appendixname{Anexa}%
41.14    \def\contentsname{Cuprins}%
41.15    \def\listfigurename{List\u{a} de figuri}%
41.16    \def\listtablename{List\u{a} de tabele}%
41.17    \def\indexname{Glosar}%
41.18    \def\figurename{Figura}%      % sau Plan\c{s}a
41.19    \def\tablename{Tabela}%
41.20    \def\partname{Partea}%
41.21    \def\enclname{Anex\u{a}}%    % sau Anexe
41.22    \def\ccname{Copie}%
41.23    \def\headtoname{Pentru}%
41.24    \def\pagename{Pagina}%
41.25    \def\seename{Vezi}%
41.26    \def\alsoname{Vezi de asemenea}%
41.27    \def\proofname{Demonstra\c{t}ie} %
41.28    \def\glossaryname{Glosar}%
41.29    }%
```

`\dateromanian` The macro `\dateromanian` redefines the command `\today` to produce Romanian dates.

```
41.30 \def\dateromanian{%
41.31     \def\today{\number\day~\ifcase\month\or
41.32         ianuarie\or februarie\or martie\or aprilie\or mai\or
41.33         iunie\or iulie\or august\or septembrie\or octombrie\or
41.34         noiembrie\or decembrie\fi
41.35     \space \number\year}}
```

`\extrasromanian` The macro `\extrasromanian` will perform all the extra definitions needed for the Romanian language. The macro `\noextrasromanian` is used to cancel the actions

⁴⁸The file described in this section has version number v1.2l and was last revised on 2005/03/31. A contribution was made by Umstatter Horst (hhu@cernvm.cern.ch).

of `\extrasromanian` For the moment these macros are empty but they are defined for compatibility with the other language definition files.

41.36 `\addto\extrasromanian{}`

41.37 `\addto\noextrasromanian{}`

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

41.38 `\ldf@finish{romanian}`

41.39 `\code`

42 The Danish language

The file `danish.dtx`⁴⁹ defines all the language definition macros for the Danish language.

For this language the character " is made active. In table 16 an overview is given of its purpose.

"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
""	like "-", but producing no hyphen sign (for words that should break at some sign such as "entrada/salida.")
"‘	lowered double left quotes (looks like „)
"’	normal double right quotes
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).
\-	like the old \-, but allowing hyphenation in the rest of the word.

Table 16: The extra definitions made by `danish.ldf`

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

42.1 `(*code)`

42.2 `\LdfInit{danish}\captionsdanish`

When this file is read as an option, i.e. by the `\usepackage` command, `danish` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@danish` to see whether we have to do something here.

42.3 `\ifx\l@danish\@undefined`

42.4 `\@nopatterns{Danish}`

42.5 `\adddialect\l@danish0\fi`

`\englishhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

42.6 `\providehyphenmins{\CurrentOption}{\tw@{tw@}}`

The next step consists of defining commands to switch to (and from) the Danish language.

`\captionsdanish` The macro `\captionsdanish` defines all strings used in the four standard documentclasses provided with L^AT_EX.

42.7 `\addto\captionsdanish{%`

42.8 `\def\prefacename{Forord}%`

42.9 `\def\refname{Litteratur}%`

42.10 `\def\abstractname{Resum\'e}%`

42.11 `\def\bibname{Litteratur}%`

42.12 `\def\chaptername{Kapitel}%`

42.13 `\def\appendixname{Bilag}%`

42.14 `\def\contentsname{Indhold}%`

42.15 `\def\listfigurename{Figurer}%`

42.16 `\def\listtablename{Tabeller}%`

42.17 `\def\indexname{Indeks}%`

42.18 `\def\figurename{Figur}%`

42.19 `\def\tablename{Tabel}%`

42.20 `\def\partname{Del}%`

42.21 `\def\enclname{Vedlagt}%`

42.22 `\def\ccname{Kopi til}% or Kopi sendt til`

⁴⁹The file described in this section has version number v1.3r and was last revised on 2009/09/19. A contribution was made by Henning Larsen (larsen@cernvm.cern.ch)


```

42.23 \def\headtoname{Til}% in letter
42.24 \def\pagename{Side}%
42.25 \def\seename{Se}%
42.26 \def\alsoname{Se ogs{\aa}}%
42.27 \def\proofname{Bevis}%
42.28 \def\glossaryname{Gloseliste}%
42.29 }%

```

`\datedanish` The macro `\datedanish` redefines the command `\today` to produce Danish dates.

```

42.30 \def\datedanish{%
42.31   \def\today{\number\day.\~\ifcase\month\or
42.32     januar\or februar\or marts\or april\or maj\or juni\or
42.33     juli\or august\or september\or oktober\or november\or december\fi
42.34     \space\number\year}}

```

`\extrasdanish` The macro `\extrasdanish` will perform all the extra definitions needed for the Danish language. The macro `\noextrasdanish` is used to cancel the actions of `\extrasdanish`.

Danish typesetting requires `\frenchspacing` to be in effect.

```

42.35 \addto\extrasdanish{\bbl@frenchspacing}
42.36 \addto\noextrasdanish{\bbl@nonfrenchspacing}

```

For Danish the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the danish group of shorthands should be used.

```

42.37 \initiate@active@char{"}
42.38 \addto\extrasdanish{\languageshorthands{danish}}
42.39 \addto\extrasdanish{\bbl@activate{}}

```

Don't forget to turn the shorthands off again.

```

42.40 \addto\noextrasdanish{\bbl@deactivate{}}

```

First we define access to the low opening double quote and guillemets for quotations,

```

42.41 \declare@shorthand{danish}{"'}{%
42.42   \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
42.43 \declare@shorthand{danish}{"'}{%
42.44   \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
42.45 \declare@shorthand{danish}{"<"}{%
42.46   \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
42.47 \declare@shorthand{danish}{">"}{%
42.48   \textormath{\guillemotright}{\mbox{\guillemotright}}}

```

then we define commands to be able to specify hyphenation breakpoints that behave a little different from `\-`.

```

42.49 \declare@shorthand{danish}{"-}{\nobreak-\bbl@allowhyphens}
42.50 \declare@shorthand{danish}{""}{\hskip\z@skip}
42.51 \declare@shorthand{danish}{""}{\textormath{\leavevmode\hbox{-}}{-}}
42.52 \declare@shorthand{danish}{"="}{\nobreak-\hskip\z@skip}

```

And we want to have a shorthand for disabling a ligature.

```

42.53 \declare@shorthand{danish}{"|"}{%
42.54   \textormath{\discretionary{-}{-}{\kern.03em}}{}}

```

To enable hyphenation in two words, written together but separated by a slash, as in 'uitdrukking/opmerking' we define the command `"/`.

```

42.55 \declare@shorthand{danish}{"/}{\textormath
42.56   {\bbl@allowhyphens\discretionary{/}{/}{\bbl@allowhyphens}}}

```

`\-` All that is left now is the redefinition of `\-`. The new version of `\-` should indicate an extra hyphenation position, while allowing other hyphenation positions to be

generated automatically. The standard behaviour of T_EX in this respect is very unfortunate for languages such as Dutch and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that T_EX can generate from the hyphenation patterns.

```
42.57 \expandafter\addto\csname extras\CurrentOption\endcsname{%
42.58   \babel@save\-\}
42.59 \expandafter\addto\csname extras\CurrentOption\endcsname{%
42.60   \def\-\{\bbl@allowhyphens\discretionary{-}{ }\bbl@allowhyphens}}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
42.61 \ldf@finish{danish}
42.62 \code>
```

43 The Icelandic language

43.1 Overview

The file `iceland.dtx`⁵⁰ defines all the language definition macros for the Icelandic language

Customization for the Icelandic language was made following several official and semiofficial publications [2, 3, 1, 6, 5]. These publications do not always agree and we indicate those instances.

For this language the character " is made active. In table 17 an overview is given of its purpose. The shorthands in table 17 can also be typeset by using the

"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
""	like "-", but producing no hyphen sign (for compound words with hyphen, e.g. x-""y).
"~	for a compound word mark without a breakpoint.
"=	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
"‘	for Icelandic left double quotes (looks like „).
"’	for Icelandic right double quotes.
">	for Icelandic ‘french’ left double quotes (similar to >>).
"<	for Icelandic ‘french’ right double quotes (similar to <<).
"o	for old Icelandic o
"O	for old Icelandic O
"ó	for old Icelandic ó
"Ó	for old Icelandic Ó
"e	for old Icelandic e
"E	for old Icelandic E
"é	for old Icelandic é
"É	for old Icelandic É
\tala	for typesetting numbers
\grada	for the ‘degree’ symbol
\gradur	for ‘degrees’, e.g. 5 °C
\upp	for textsuperscript

Table 17: The shorthands and extra definitions made by `icelandic.ldf`

commands in table 18.

43.2 References

- [1] Alþingi. *Reglur um frágang þingskjala og prentun umræðna*, 1988.
- [2] Auglýsing um greinarmerkjasetningu. Stj.tíð B, nr. 133/1974, 1974.
- [3] Auglýsing um breyting auglýsingu nr. 132/1974 um íslenska stafsetningu. Stj.tíð B, nr. 261/1977, 1977.
- [4] Einar Haugen, editor. *First Grammatical Treatise*. Longman, London, 2 edition, 1972.
- [5] Staðlaráð Íslands og Fagráð í upplýsingatækni, Reykjavík. *Forstaðall FS 130:1997*, 1997.
- [6] STRÍ Staðlaráð Íslands. *SI - kerfið*, 2 edition, 1994.

⁵⁰The file described in this section has version number ? and was last revised on ?.

<code>\ilqq</code>	for Icelandic left double quotes (looks like „).
<code>\irqq</code>	for Icelandic right double quotes (looks like “).
<code>\ilq</code>	for Icelandic left single quotes (looks like ,).
<code>\irq</code>	for Icelandic right single quotes (looks like ‘).
<code>\iflqq</code>	for Icelandic ‘french’ left double quotes (similar to >>).
<code>\ifrqq</code>	for Icelandic ‘french’ right double quotes (similar to <<).
<code>\ifrq</code>	for Icelandic ‘french’ right single quotes (similar to <).
<code>\iflq</code>	for Icelandic ‘french’ left single quotes (similar to >).
<code>\dq</code>	the original quotes character (“).
<code>\oob</code>	for old Icelandic o
<code>\Oob</code>	for old Icelandic O
<code>\ooob</code>	for old Icelandic ó
<code>\OOob</code>	for old Icelandic Ó
<code>\eob</code>	for old Icelandic e
<code>\Eob</code>	for old Icelandic E
<code>\eeob</code>	for old Icelandic é
<code>\EEob</code>	for old Icelandic É

Table 18: Commands which produce quotes and old Icelandic diacritics, defined by `icelandic.ldf`

43.3 T_EXnical details

When this file was read through the option `icelandic` we make it behave as if `icelandic` was specified.

```

43.1 \def\bbl@tempa{icelandic}
43.2 \ifx\CurrentOption\bbl@tempa
43.3   \def\CurrentOption{icelandic}
43.4 \fi

```

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```

43.5 \*code>
43.6 \LdfInit\CurrentOption{captions\CurrentOption}

```

When this file is read as an option, i.e., by the `\usepackage` command, `icelandic` will be an ‘unknown’ language, so we have to make it known. So we check for the existence of `\l@icelandic` to see whether we have to do something here.

```

43.7 \ifx\l@icelandic\@undefined
43.8   \nopatterns{Icelandic}
43.9   \adddialect\l@icelandic0
43.10 \fi

```

`\if@Two@E` We will need a new ‘if’: `\if@Two@E` is true if and only if L^AT_EX 2_ε is running *not* in compatibility mode. It is used in the definitions of the command `\tala` and `\upp`. The definition is somewhat complicated, due to the fact that `\if@compatibility` is not recognized as a `\if` in L^AT_EX-2.09 based formats.

```

43.11 \newif\if@Two@E \@Two@Etrue
43.12 \def\@FI@{\fi}
43.13 \ifx\@compatibilitytrue\@undefined
43.14   \@Two@Efalse \def\@FI@{\relax}
43.15 \else
43.16   \if@compatibility \@Two@Efalse \fi
43.17 \@FI@

```

`\extrasicelandic` The macro `\extrasicelandic` will perform all the extra definitions needed for the Icelandic language. The macro `\noextrasicelandic` is used to cancel the actions of `\extrasicelandic`.

For Icelandic the " character is made active. This is done once, later on its definition may vary.

```
43.18 \initiate@active@char{"}
43.19 \@namedef{extras\CurrentOption}{%
43.20   \languageshorthands{icelandic}}
43.21 \expandafter\addto\csname extras\CurrentOption\endcsname{%
43.22   \bbl@activate{"}}
```

Don't forget to turn the shorthands off again.

```
43.23 \addto\noextrasicelandic{\bbl@deactivate{"}}
```

The icelandic hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
43.24 \providehyphenmins{\CurrentOption}{\tw@}\tw@}
```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 18.

To be able to define the function of " , we first define a couple of 'support' macros.

43.4 Captionnames and date

The next step consists of defining the Icelandic equivalents for the L^AT_EX captionnames.

`\captionsicelandic` The macro `\captionsicelandic` will define all strings used in the four standard document classes provided with L^AT_EX.

```
43.25 \@namedef{captions\CurrentOption}{%
43.26   \def\prefacename{Form\'}{a}li}%
43.27   \def\refname{Heimildir}%
43.28   \def\abstractname{\'}{U}tdr\'}{a}ttur}%
43.29   \def\bibname{Heimildir}%
43.30   \def\chaptername{Kafli}%
43.31   \def\appendixname{Vi\'}{dh}auki}%
43.32   \def\contentsname{Efnisyfirlit}%
43.33   \def\listfigurename{Myndaskr\'}{a}}%
43.34   \def\listtablename{T\'}{o}fluskr\'}{a}}%
43.35   \def\indexname{Atri\'}{dh}isor\'}{dh}askr\'}{a}}%
43.36   \def\figurename{Mynd}%
43.37   \def\tablename{Tafla}%
43.38   \def\partname{Hluti}%
43.39   \def\enclname{Hj\'}{a}lagt}%
43.40   \def\ccname{Samrit}%
43.41   \def\headtoname{Til:}% in letter
43.42   \def\pagename{Bla\'}{dh}s\'}{i}{\'}{dh}a}%
43.43   \def\seename{Sj\'}{a}}%
43.44   \def\alsoname{Sj\'}{a} einnig}%
43.45   \def\proofname{S\'}{o}nnun}%
43.46   \def\glossaryname{Or\'}{dh}alisti}%
43.47 }
```

`\dateicelandic` The macro `\dateicelandic` redefines the command `\today` to produce Icelandic dates.

```
43.48 \def\dateicelandic{%
43.49   \def\today{\number\day.\~\ifcase\month\or
43.50     jan\'}{u}ar\or febr\'}{u}ar\or mars\or apr\'}{i}l\or ma\'}{i}\or
43.51     j\'}{u}n\'}{i}\or j\'}{u}l\'}{i}\or \'}{a}g\'}{u}st\or september\or
43.52     okt\'}{o}ber\or n\'}{o}vember\or desember\fi
43.53     \space\number\year}}
```

43.5 Icelandic quotation marks

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`.

```
43.54 \begingroup \catcode'\ "12
43.55 \def\x{\endgroup
43.56 \def\SS{\mathchar"7019 }
43.57 \def\dq{"}}
43.58 \x
```

Now we can define the icelandic and icelandic ‘french’ quotes. The icelandic ‘french’ guillemets are the reverse of french guillemets. We define single icelandic ‘french’ quotes for compatibility. Shorthands are provided for a number of different quotation marks, which make them useable both outside and inside mathmode.

```
43.59 \let\ilq\grq
43.60 \let\irq\grq
43.61 \let\iflq\frq
43.62 \let\ifrq\flq
43.63 \let\ilqq\glqq
43.64 \let\irqq\grqq
43.65 \let\iflqq\frqq
43.66 \let\ifrqq\flqq

43.67 \declare@shorthand{icelandic}{''}{\glqq}
43.68 \declare@shorthand{icelandic}{''}{\grqq}
43.69 \declare@shorthand{icelandic}{'>'}{\frqq}
43.70 \declare@shorthand{icelandic}{'<'}{\flqq}
```

and some additional commands:

```
43.71 \declare@shorthand{icelandic}{-}{\nobreak\-\bbl@allowhyphens}
43.72 \declare@shorthand{icelandic}{-|}{\%}
43.73 \textormath{\nobreak\discretionary{-}{-}{\kern.03em}%
43.74 \bbl@allowhyphens}{-}{-}
43.75 \declare@shorthand{icelandic}{-}{\hspace\z@skip}
43.76 \declare@shorthand{icelandic}{-}{\textormath{\leavevmode\hbox{-}{-}}{-}}
43.77 \declare@shorthand{icelandic}{-}{\nobreak-\hspace\z@skip}
```

43.6 Old Icelandic

In old Icelandic some letters have special diacritical marks, described for example in *First Grammatical Treatise* [4, 5]. We provide these in the T1 encoding with the ‘ogonek’. The ogonek is placed with the letters ‘o’, and ‘O’, ‘ó’ and ‘Ó’, ‘e’ and ‘E’, and ‘é’ and ‘É’. Shorthands are provided for these as well.

The following code by Leszek Holenderski lifted from `polish.dtx` is designed to position the diacritics correctly for every font in every size. These macros need a few extra dimension variables.

```
43.78 \newdimen\pl@left
43.79 \newdimen\pl@down
43.80 \newdimen\pl@right
43.81 \newdimen\pl@temp
```

`\sob` The macro `\sob` is used to put the ‘ogonek’ in the right place.

```
43.82 \def\sob#1#2#3#4#5{%parameters: letter and fractions hl,ho,vl,vo
43.83 \setbox0\hbox{#1}\setbox1\hbox{\k{}}\setbox2\hbox{p}%
43.84 \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
43.85 \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
43.86 \pl@left=\pl@right \advance\pl@left by\wd1
43.87 \pl@temp=-\pl@down \advance\pl@temp by\dp2 \dp1=\pl@temp
43.88 \leavevmode
43.89 \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
```

```

\oob
\Oob 43.90 \DeclareTextCommand{\oob}{T1}{\sob {o}{.85}{0}{.04}{0}}
\ooob 43.91 \DeclareTextCommand{\Oob}{T1}{\sob {O}{.7}{0}{0}{0}}
\OOob 43.92 \DeclareTextCommand{\ooob}{T1}{\sob {ó}{.85}{0}{.04}{0}}
\eob 43.93 \DeclareTextCommand{\Oob}{T1}{\sob {ó}{.7}{0}{0}{0}}
\Eob 43.94 \DeclareTextCommand{\eob}{T1}{\sob {e}{1}{0}{.04}{0}}
\eeob 43.95 \DeclareTextCommand{\Eob}{T1}{\sob {E}{1}{0}{.04}{0}}
\EEob 43.96 \DeclareTextCommand{\eeob}{T1}{\sob {é}{1}{0}{.04}{0}}
\EEob 43.97 \DeclareTextCommand{\EEob}{T1}{\sob {É}{1}{0}{.04}{0}}

43.98 \declare@shorthand{icelandic}{"}{\oob}
43.99 \declare@shorthand{icelandic}{"}{\Oob}
43.100 \declare@shorthand{icelandic}{"}{\ooob}
43.101 \declare@shorthand{icelandic}{"}{\OOob}
43.102 \declare@shorthand{icelandic}{"}{\eob}
43.103 \declare@shorthand{icelandic}{"}{\Eob}
43.104 \declare@shorthand{icelandic}{"}{\eeob}
43.105 \declare@shorthand{icelandic}{"}{\EEob}

```

43.7 Formatting numbers

This section is lifted from `frenchb.dtx` by D. Flipo. In English the decimal part starts with a point and thousands should be separated by a comma: an approximation of 1000π should be inputed as `$3{,}141.592{,}653$` in math-mode and as `3,141.592,653` in text.

In Icelandic the decimal part starts with a comma and thousands should be separated by a space [1] or by a period [5]; we have the space. The above approximation of 1000π should be inputed as `$3\;141{,}592\;653$` in math-mode and as something like `3~141,592~653` in text. Braces are mandatory around the comma in math-mode, the reason is mentioned in the `TEXbook` p. 134: the comma is of type `\mathpunct` (thus normally followed by a space) while the point is of type `\mathord` (no space added).

Thierry Bouche suggested that a second type of comma, of type `\mathord` would be useful in math-mode, and proposed to introduce a command (named `\decimalsep` in this package), the expansion of which would depend on the current language.

Vincent Jalby suggested a command `\nombre` to conveniently typeset numbers: inputting `\nombre{3141,592653}` either in text or in math-mode will format this number properly according to the current language (Icelandic or non-Icelandic). We use `\nombre` to define command `\tala` in Icelandic.

`\tala` accepts an optional argument which happens to be useful with the extension ‘`dcolumn`’, it specifies the decimal separator used in the *source code*: `\newcolumntype{d}{D{,}{\decimalsep}{-1}}`

```

\begin{tabular}{d}\hline
3,14 \\\
\tala[,]{123,4567} \\\
\tala[,]{9876,543}\hline
\end{tabular}

```

will print a column of numbers aligned on the decimal point (comma or point depending on the current language), each slice of 3 digits being separated by a space or a comma according to the current language.

`\decimalsep` We need a internal definition, valid in both text and math-mode, for the comma (`\@comma@`) and another one for the unbreakable fixed length space (no glue) used in Icelandic (`\f@thousandsep`).

The commands `\decimalsep` and `\thousandsep` get default definitions (for the English language) when `icelandic` is loaded; these definitions will be updated when the current language is switched to or from Icelandic.

```

43.106 \mathchardef\m@comma="013B \def\@comma@{\ifmmode\m@comma\else,\fi}

```

```

43.107 \def\f@thousandsep{\ifmode\mskip5.5mu\else\penalty\M\kern.3em\fi}
43.108 \newcommand{\decimalsep}{.} \newcommand{\thousandsep}{\@comma@}
43.109 \expandafter\addto\csname extras\CurrentOption\endcsname{%
43.110     \def\decimalsep{\@comma@}%
43.111     \def\thousandsep{\f@thousandsep}}
43.112 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
43.113     \def\decimalsep{.}%
43.114     \def\thousandsep{\@comma@}}

```

`\tala` The decimal separator used when *inputing* a number with `\tala` has to be a comma. `\tala` splits the inputted number into two parts: what comes before the first comma will be formatted by `\@integerpart` while the rest (if not empty) will be formatted by `\@decimalpart`. Both parts, once formatted separately will be merged together with between them, either the decimal separator `\decimalsep` or (in L^AT_EX_{2 ϵ} only) the optional argument of `\tala`.

```

43.115 \if@Two@E
43.116   \newcommand{\tala}[2][\decimalsep]{%
43.117     \def\@decimalsep{#1}\@tala#2\@empty,\@empty,\@nil}
43.118 \else
43.119   \newcommand{\tala}[1]{%
43.120     \def\@decimalsep{\decimalsep}\@tala#1\@empty,\@empty,\@nil}
43.121 \fi
43.122 \def\@tala#1,#2,#3\@nil{%
43.123   \ifx\@empty#2%
43.124     \@integerpart{#1}%
43.125   \else
43.126     \@integerpart{#1}\@decimalsep\@decimalpart{#2}%
43.127   \fi}

```

The easiest bit is the decimal part: We attempt to read the first four digits of the decimal part, if it has less than 4 digits, we just have to print them, otherwise `\thousandsep` has to be appended after the third digit, and the algorithm is applied recursively to the rest of the decimal part.

```

43.128 \def\@decimalpart#1{\@@decimalpart#1\@empty\@empty\@empty}
43.129 \def\@@decimalpart#1#2#3#4{#1#2#3%
43.130   \ifx\@empty#4%
43.131   \else
43.132     \thousandsep\expandafter\@@decimalpart\expandafter#4%
43.133   \fi}

```

Formatting the integer part is more difficult because the slices of 3 digits start from the *bottom* while the number is read from the top! This (tricky) code is borrowed from David Carlisle's comma.sty.

```

43.134 \def\@integerpart#1{\@integerpart{#1}\@empty\@empty\@empty}
43.135 \def\@integerpart#1#2#3#4{%
43.136   \ifx\@empty#2%
43.137     \@addthousandsep#1\relax
43.138   \else
43.139     \ifx\@empty#3%
43.140       \@addthousandsep\@empty\@empty#1#2\relax
43.141     \else
43.142       \ifx\@empty#4%
43.143         \@addthousandsep\@empty#1#2#3\relax
43.144       \else
43.145         \@integerpartafterfi{#1#2#3#4}%
43.146       \fi
43.147     \fi
43.148   \fi}
43.149 \def\@integerpartafterfi#1\fi\fi\fi{\fi\fi\fi\@integerpart{#1}}
43.150 \def\@addthousandsep#1#2#3#4{#1#2#3%
43.151   \if#4\relax
43.152   \else

```



```

43.153 \thousandsep\expandafter\@addthousandsep\expandafter#4%
43.154 \fi}

```

43.8 Extra utilities

We now provide the Icelandic user with some extra utilities.

`\upp` `\upp` is for typesetting superscripts. `\upp` relies on

`\upp@size` The internal macro `\upp@size` holds the size at which the superscript will be typeset. The reason for this is that we have to specify it differently for different formats.

```

43.155 \ifx\sevenrm\@undefined
43.156 \ifx\@ptsize\@undefined
43.157 \let\upp@size\small
43.158 \else
43.159 \ifx\selectfont\@undefined

```

In this case the format is the original L^AT_EX-2.09:

```

43.160 \ifcase\@ptsize
43.161 \let\upp@size\ixpt\or
43.162 \let\upp@size\xpt\or
43.163 \let\upp@size\xipt
43.164 \fi

```

When `\selectfont` is defined we probably have NFSS available:

```

43.165 \else
43.166 \ifcase\@ptsize
43.167 \def\upp@size{\fontsize\@ixpt{10pt}\selectfont}\or
43.168 \def\upp@size{\fontsize\@xpt{11pt}\selectfont}\or
43.169 \def\upp@size{\fontsize\@xipt{12pt}\selectfont}
43.170 \fi
43.171 \fi
43.172 \fi
43.173 \else

```

If we end up here it must be a plain based T_EX format, so:

```

43.174 \let\upp@size\sevenrm
43.175 \fi

```

Now we can define `\upp`. When L^AT_EX 2_ε runs in compatibility mode (L^AT_EX-2.09 emulation), `\textsuperscript` is also defined, but does no good job, so we give two different definitions for `\upp` using `\if@Two@E`.

```

43.176 \if@Two@E
43.177 \DeclareRobustCommand*\upp[1]{\textsuperscript{#1}}
43.178 \else
43.179 \DeclareRobustCommand*\upp[1]{%
43.180 \leavevmode\raise1ex\hbox{\upp@size#1}}
43.181 \fi

```

Some definitions for special characters. `\grada` needs a special treatment: it is `\char6` in T1-encoding and `\char23` in OT1-encoding.

```

43.182 \ifx\fmtname\LaTeXeFmtName
43.183 \DeclareTextSymbol{\grada}{T1}{6}
43.184 \DeclareTextSymbol{\grada}{OT1}{23}
43.185 \else
43.186 \def\T@one{T1}
43.187 \ifx\f@encoding\T@one
43.188 \newcommand{\grada}{\char6}
43.189 \else
43.190 \newcommand{\grada}{\char23}
43.191 \fi
43.192 \fi

```

`\gradur` Macro for typesetting the abbreviation for ‘degrees’ (as in ‘degrees Celsius’). As the bounding box of the character ‘degree’ has *very* different widths in CMR/DC and PostScript fonts, we fix the width of the bounding box of `\gradur` to 0.3em, this lets the symbol ‘degree’ stick to the preceding (e.g., `45\gradur`) or following character (e.g., `20~\gradur C`).

```
43.193 \DeclareRobustCommand*{\gradur}{%
43.194         \leavevmode\hbox to 0.3em{\hss\grada\hss}}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
43.195 \ldf@finish\CurrentOption
43.196 \code
```

44 The Norwegian language

The file `norsk.dtx`⁵¹ defines all the language definition macros for the Norwegian language as well as for an alternative variant ‘nynorsk’ of this language.

For this language the character " is made active. In table 19 an overview is given of its purpose.

"ff	for ff to be hyphenated as ff-f, this is also implemented for b, d, f, g, l, m, n, p, r, s, and t. (<code>o"ppussing</code>)
"ee	Hyphenate "ee as \'e-e. (<code>komit"een</code>)
"-	an explicit hyphen sign, allowing hyphenation in the composing words. Use this for compound words when the hyphenation patterns fail to hyphenate properly. (<code>alpin"-anlegg</code>)
"	Like "-", but inserts 0.03em space. Use it if the compound point is spanned by a ligature. (<code>hoff" intriger</code>)
""	Like "-", but producing no hyphen sign. (<code>i""g\aa{r}</code>)
"~	Like -, but allows no hyphenation at all. (<code>E"~cup</code>)
"=	Like -, but allowing hyphenation in the composing words. (<code>marksistisk"=leninistisk</code>)
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).

Table 19: The extra definitions made by `norsk.sty`

Rune Kleveland distributes a Norwegian dictionary for ispell (570000 words). It can be found at <http://www.uio.no/~runekl/dictionary.html>.

This dictionary supports the spellings `spi"sslede` for ‘spisslede’ (hyphenated spiss-slede) and other such words, and also suggest the spelling `spi"sslede` for ‘spisslede’ and ‘spisslede’.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
44.1 (*code)
44.2 \LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `norsk` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@norsk` to see whether we have to do something here.

```
44.3 \ifx\l@norsk\@undefined
44.4     \@nopatterns{Norsk}
44.5     \adddialect\l@norsk0\fi
```

`\norskhyphenmins` Some sets of Norwegian hyphenation patterns can be used with `\lefthyphenmin` set to 1 and `\righthyphenmin` set to 2, but the most common set `nohyph.tex` can’t. So we use `\lefthyphenmin=2` by default.

```
44.6 \providehyphenmins{\CurrentOption}{\tw@{tw@}}
```

Now we have to decide which version of the captions should be made available. This can be done by checking the contents of `\CurrentOption`.

```
44.7 \def\bbl@tempa{norsk}
44.8 \ifx\CurrentOption\bbl@tempa
```

The next step consists of defining commands to switch to (and from) the Norwegian language.

⁵¹The file described in this section has version number v2.0h and was last revised on 2005/03/30. Contributions were made by Haavard Helstrup (HAAVARD@CERNVM) and Alv Kjetil Holme (HOLMEA@CERNVM); the ‘nynorsk’ variant has been supplied by Per Steinar Iversen (iversen@vxcern.cern.ch) and Terje Engeset Petterst (TERJEEP@VSFYS1.FI.UIB.NO); the short-hand definitions were provided by Rune Kleveland (runekl@math.uio.no).

`\captionsnorsk` The macro `\captionsnorsk` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```

44.9  \def\captionsnorsk{%
44.10  \def\prefacename{Forord}%
44.11  \def\refname{Referanser}%
44.12  \def\abstractname{Sammendrag}%
44.13  \def\bibName{Bibliografi}%      or Litteraturoversikt
44.14  %                               or Litteratur or Referanser
44.15  \def\chaptername{Kapittel}%
44.16  \def\appendixname{Tillegg}%     or Appendiks
44.17  \def\contentsname{Innhold}%
44.18  \def\listfigurename{Figurer}%   or Figurliste
44.19  \def\listtablename{Tabeller}%   or Tabelliste
44.20  \def\indexname{Register}%
44.21  \def\figurename{Figur}%
44.22  \def\tablename{Tabell}%
44.23  \def\partname{Del}%
44.24  \def\enclname{Vedlegg}%
44.25  \def\ccname{Kopi sendt}%
44.26  \def\headtoname{Til}% in letter
44.27  \def\pagename{Side}%
44.28  \def\seename{Se}%
44.29  \def\alsoname{Se ogs\aa{}}%
44.30  \def\proofname{Bevis}%
44.31  \def\glossaryname{Ordliste}%
44.32  }
44.33 \else

```

For the ‘nynorsk’ version of these definitions we just add a “dialect”.

```

44.34  \adddialect\l@nynorsk\l@norsk

```

`\captionsnynorsk` The macro `\captionsnynorsk` defines all strings used in the four standard documentclasses provided with L^AT_EX, but using a different spelling than in the command `\captionsnorsk`.

```

44.35  \def\captionsnynorsk{%
44.36  \def\prefacename{Forord}%
44.37  \def\refname{Referansar}%
44.38  \def\abstractname{Samandrag}%
44.39  \def\bibName{Litteratur}%      or Litteraturoversyn
44.40  %                               or Referansar
44.41  \def\chaptername{Kapittel}%
44.42  \def\appendixname{Tillegg}%   or Appendiks
44.43  \def\contentsname{Innhald}%
44.44  \def\listfigurename{Figurar}% or Figurliste
44.45  \def\listtablename{Tabellar}% or Tabelliste
44.46  \def\indexname{Register}%
44.47  \def\figurename{Figur}%
44.48  \def\tablename{Tabell}%
44.49  \def\partname{Del}%
44.50  \def\enclname{Vedlegg}%
44.51  \def\ccname{Kopi til}%
44.52  \def\headtoname{Til}% in letter
44.53  \def\pagename{Side}%
44.54  \def\seename{Sj\aa{}}%
44.55  \def\alsoname{Sj\aa{} \‘{o}g}%
44.56  \def\proofname{Bevis}%
44.57  \def\glossaryname{Ordliste}%
44.58  }
44.59 \fi

```

`\datenorsk` The macro `\datenorsk` redefines the command `\today` to produce Norwegian dates.

```

44.60 \@namedef{date\CurrentOption}{%
44.61   \def\today{\number\day.\~\ifcase\month\or
44.62     januar\or februar\or mars\or april\or mai\or juni\or
44.63     juli\or august\or september\or oktober\or november\or desember
44.64     \fi
44.65     \space\number\year}}

```

`\extrasnorsk` The macro `\extrasnorsk` will perform all the extra definitions needed for the Norwegian language. The macro `\noextrasnorsk` is used to cancel the actions of `\extrasnorsk`.

Norwegian typesetting requires `\frenchspacing` to be in effect.

```

44.66 \@namedef{extras\CurrentOption}{\bbl@frenchspacing}
44.67 \@namedef{noextras\CurrentOption}{\bbl@nonfrenchspacing}

```

For Norsk the " character is made active. This is done once, later on its definition may vary.

```

44.68 \initiate@active@char{"}
44.69 \expandafter\addto\csname extras\CurrentOption\endcsname{%
44.70   \languageshorthands{norsk}}
44.71 \expandafter\addto\csname extras\CurrentOption\endcsname{%
44.72   \bbl@activate{"}}

```

Don't forget to turn the shorthands off again.

```

44.73 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
44.74   \bbl@deactivate{"}}

```

The code above is necessary because we need to define a number of shorthand commands. These shorthand commands are then used as indicated in table 19.

To be able to define the function of ", we first define a couple of 'support' macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`.

```

44.75 \begingroup \catcode'\12
44.76 \def\x{\endgroup
44.77   \def\@SS{\mathchar"7019 }
44.78   \def\dq{"}}
44.79 \x

```

Now we can define the discretionary shorthand commands. The number of words where such hyphenation is required is for each character

b	d	f	g	k	l	n	p	r	s	t
4	4	15	3	43	30	8	12	1	33	35

taken from a list of 83000 ispell-roots.

```

44.80 \declare@shorthand{norsk}{"b}{\textormath{\bbl@disc b{bb}}{b}}
44.81 \declare@shorthand{norsk}{"B}{\textormath{\bbl@disc B{BB}}{B}}
44.82 \declare@shorthand{norsk}{"d}{\textormath{\bbl@disc d{dd}}{d}}
44.83 \declare@shorthand{norsk}{"D}{\textormath{\bbl@disc D{DD}}{D}}
44.84 \declare@shorthand{norsk}{"e}{\textormath{\bbl@disc e{'e}}{}}
44.85 \declare@shorthand{norsk}{"E}{\textormath{\bbl@disc E{'E}}{}}
44.86 \declare@shorthand{norsk}{"F}{\textormath{\bbl@disc F{FF}}{F}}
44.87 \declare@shorthand{norsk}{"g}{\textormath{\bbl@disc g{gg}}{g}}
44.88 \declare@shorthand{norsk}{"G}{\textormath{\bbl@disc G{GG}}{G}}
44.89 \declare@shorthand{norsk}{"k}{\textormath{\bbl@disc k{kk}}{k}}
44.90 \declare@shorthand{norsk}{"K}{\textormath{\bbl@disc K{KK}}{K}}
44.91 \declare@shorthand{norsk}{"l}{\textormath{\bbl@disc l{ll}}{l}}
44.92 \declare@shorthand{norsk}{"L}{\textormath{\bbl@disc L{LL}}{L}}
44.93 \declare@shorthand{norsk}{"n}{\textormath{\bbl@disc n{nn}}{n}}
44.94 \declare@shorthand{norsk}{"N}{\textormath{\bbl@disc N{NN}}{N}}
44.95 \declare@shorthand{norsk}{"p}{\textormath{\bbl@disc p{pp}}{p}}
44.96 \declare@shorthand{norsk}{"P}{\textormath{\bbl@disc P{PP}}{P}}

```

```

44.97 \declare@shorthand{norsk}{\textmath{\bbl@disc r{rr}}{r}}
44.98 \declare@shorthand{norsk}{\textmath{\bbl@disc R{RR}}{R}}
44.99 \declare@shorthand{norsk}{\textmath{\bbl@disc s{ss}}{s}}
44.100 \declare@shorthand{norsk}{\textmath{\bbl@disc S{SS}}{S}}
44.101 \declare@shorthand{norsk}{\textmath{\bbl@disc t{tt}}{t}}
44.102 \declare@shorthand{norsk}{\textmath{\bbl@disc T{TT}}{T}}

```

We need to treat "f a bit differently in order to preserve the ff-ligature.

```

44.103 \declare@shorthand{norsk}{\textmath{\bbl@discff}{f}}
44.104 \def\bbl@discff{\penalty\M
44.105   \afterassignment\bbl@insertff \let\bbl@nextff= }
44.106 \def\bbl@insertff{%
44.107   \if f\bbl@nextff
44.108     \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi
44.109     {\relax\discretionary{ff-}{f}{ff}\allowhyphens}{f\bbl@nextff}}
44.110 \let\bbl@nextff=f

```

We now define the French double quotes and some commands concerning hyphenation:

```

44.111 \declare@shorthand{norsk}{"<"}{\flqq}
44.112 \declare@shorthand{norsk}{">"}{\frqq}
44.113 \declare@shorthand{norsk}{"-"}{\penalty\M-\bbl@allowhyphens}
44.114 \declare@shorthand{norsk}{"|"}{%
44.115   \textmath{\penalty\M\discretionary{-}{-}{\kern.03em}%
44.116     \allowhyphens}{}}
44.117 \declare@shorthand{norsk}{""}{\hskip\z@skip}
44.118 \declare@shorthand{norsk}{"~"}{\textmath{\leavevmode\hbox{-}{-}}}
44.119 \declare@shorthand{norsk}{"="}{\penalty\M-\hskip\z@skip}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

44.120 \ldf@finish\CurrentOption
44.121 \code>

```

45 The Swedish language

The file `swedish.dtx`⁵² defines all the language-specific macros for the Swedish language. This file has borrowed heavily from `finnish.dtx` and `germanb.dtx`.

For this language the character " is made active. In table 20 an overview is given of its purpose. The vertical placement of the "umlaut" in some letters can be controlled this way.

"a	Gives ä, also implemented for "A, "o and "O.
"w, "W	gives å and Å.
"ff	for ff to be hyphenated as ff-f. Used for compound words, such as <code>stra"ffånge</code> , which should be hyphenated as <code>straff-fånge</code> . This is also implemented for b, d, f, g, l, m, n, p, r, s, and t.
"	disable ligature at this position. This should be used for compound words, such as “ <code>stra"ffinrättning</code> ”, which should not have the ligature “ffi”.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word, such as e. g. in “ <code>x"-axeln</code> ”.
""	like "-", but producing no hyphen sign (for words that should break at some sign such as <code>och/"eller</code>).
"~	for an explicit hyphen without a breakpoint; useful for expressions such as “ <code>2"~3 veckor</code> ” where no line-break is desirable.
"=	an explicit hyphen sign allowing subsequent hyphenation, for expressions such as “ <code>studiebidrag och -lån</code> ”.
\-	like the old \-, but allowing hyphenation in the rest of the word.

Table 20: The extra definitions made by `swedish.sty`

Two variations for formatting of dates are added. `\datesymd` makes `\today` output dates formatted as YYYY-MM-DD, which is commonly used in Sweden today. `\datesdmy` formats the date as D/M YYYY, which is also very common in Sweden. These commands should be issued after `\begindocument`.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
45.1 (*code)
45.2 \LdfInit{swedish}\captionsswedish
```

When this file is read as an option, i.e. by the `\usepackage` command, `swedish` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@swedish` to see whether we have to do something here.

```
45.3 \ifx\l@swedish\@undefined
45.4     \@nopatterns{Swedish}
45.5     \adddialect\l@swedish0\fi
```

The next step consists of defining commands to switch to the Swedish language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsswedish` The macro `\captionsswedish` defines all strings used in the four standard documentclasses provided with L^AT_EX.

⁵²The file described in this section has version number v2.3d and was last revised on 2005/03/31. Contributions were made by Sten Hellman (HELLMAN@CERNVM.CERN.CH) and Erik Östhols (erik_osthols@yahoo.com).

```

45.6 \addto\captionsswedish{%
45.7   \def\prefacename{F\"orord}%
45.8   \def\refname{Referenser}%
45.9   \def\abstractname{Sammanfattning}%
45.10  \def\bibname{Litteraturf\"orteckning}%
45.11  \def\chaptername{Kapitel}%
45.12  \def\appendixname{Bilaga}%
45.13  \def\contentsname{Inneh\"oldsname aa\endcsname ll}%
45.14  \def\listfigurename{Figurer}%
45.15  \def\listtablename{Tabeller}%
45.16  \def\indexname{Sakregister}%
45.17  \def\figurename{Figur}%
45.18  \def\tablename{Tabell}%
45.19  \def\partname{Del}%

45.20  \def\enclname{Bil.}%
45.21  \def\ccname{Kopia f\"or k\"annedom}%
45.22  \def\headtoname{Till}% in letter
45.23  \def\pagename{Sida}%
45.24  \def\seename{se}%

45.25  \def\alsoname{se \"aven}%

45.26  \def\proofname{Bevis}%
45.27  \def\glossaryname{Ordlista}%
45.28  }%

```

`\dateswedish` The macro `\dateswedish` redefines the command `\today` to produce Swedish dates.

```

45.29 \def\dateswedish{%
45.30   \def\today{%
45.31     \number\day~\ifcase\month\or
45.32     januari\or februari\or mars\or april\or maj\or juni\or
45.33     juli\or augusti\or september\or oktober\or november\or
45.34     december\fi
45.35     \space\number\year}}

```

`\datesymd` The macro `\datesymd` redefines the command `\today` to produce dates in the format YYYY-MM-DD, common in Sweden.

```

45.36 \def\datesymd{%
45.37   \def\today{\number\year-\two@digits\month-\two@digits\day}%
45.38 }

```

`\datesdmy` The macro `\datesdmy` redefines the command `\today` to produce Swedish dates in the format DD/MM YYYY, also common in Sweden.

```

45.39 \def\datesdmy{%
45.40   \def\today{\number\day/\number\month\space\number\year}%
45.41 }

```

`\swedishhyphenmins` The swedish hyphenation patterns can be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 2.

```

45.42 \providehyphenmins{swedish}{\tw@\tw@}

```

`\extrasswedish` The macro `\extrasswedish` performs all the extra definitions needed for the Swedish language. The macro `\noextrasswedish` is used to cancel the actions of `\extrasswedish`.

For Swedish texts `\frenchspacing` should be in effect. We make sure this is the case and reset it if necessary.

```

45.43 \addto\extrasswedish{\bbl@frenchspacing}
45.44 \addto\noextrasswedish{\bbl@nonfrenchspacing}

```


For Swedish the " character is made active. This is done once, later on its definition may vary.

```
45.45 \initiate@active@char{"}
45.46 \addto\extrasswedish{\languageshorthands{swedish}}
45.47 \addto\extrasswedish{\bbl@activate{"}}
```

Don't forget to turn the shorthands off again.

```
45.48 \addto\noextrasswedish{\bbl@deactivate{"}}
```

The “umlaut” accent macro \ is changed to lower the “umlaut” dots. The redefinition is done with the help of \umlautlow.

```
45.49 \addto\extrasswedish{\babel@save"\umlautlow}
45.50 \addto\noextrasswedish{\umlauthigh}
```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 20.

To be able to define the function of ", we first define a couple of ‘support’ macros.

\dq We save the original double quote character in \dq to keep it available, the math accent \ can now be typed as ".

```
45.51 \beginingroup \catcode'\ "12
45.52 \def\x{\endgroup
45.53 \def\@SS{\mathchar"7019 }
45.54 \def\dq{"}}
45.55 \x
```

Now we can define the doublequote macros: the umlauts and å.

```
45.56 \declare@shorthand{swedish}{"w}{\textormath{\aa\allowhyphens}{\ddot w}}
45.57 \declare@shorthand{swedish}{"a}{\textormath{\ "a\allowhyphens}{\ddot a}}
45.58 \declare@shorthand{swedish}{"o}{\textormath{\ "o\allowhyphens}{\ddot o}}
45.59 \declare@shorthand{swedish}{"W}{\textormath{\AA\allowhyphens}{\ddot W}}
45.60 \declare@shorthand{swedish}{"A}{\textormath{\ "A\allowhyphens}{\ddot A}}
45.61 \declare@shorthand{swedish}{"O}{\textormath{\ "O\allowhyphens}{\ddot O}}
```

discretionary commands

```
45.62 \declare@shorthand{swedish}{"b}{\textormath{\bbl@disc b{bb}}{b}}
45.63 \declare@shorthand{swedish}{"B}{\textormath{\bbl@disc B{BB}}{B}}
45.64 \declare@shorthand{swedish}{"d}{\textormath{\bbl@disc d{dd}}{d}}
45.65 \declare@shorthand{swedish}{"D}{\textormath{\bbl@disc D{DD}}{D}}
45.66 \declare@shorthand{swedish}{"f}{\textormath{\bbl@disc f{ff}}{f}}
45.67 \declare@shorthand{swedish}{"F}{\textormath{\bbl@disc F{FF}}{F}}
45.68 \declare@shorthand{swedish}{"g}{\textormath{\bbl@disc g{gg}}{g}}
45.69 \declare@shorthand{swedish}{"G}{\textormath{\bbl@disc G{GG}}{G}}
45.70 \declare@shorthand{swedish}{"l}{\textormath{\bbl@disc l{ll}}{l}}
45.71 \declare@shorthand{swedish}{"L}{\textormath{\bbl@disc L{LL}}{L}}
45.72 \declare@shorthand{swedish}{"m}{\textormath{\bbl@disc m{mm}}{m}}
45.73 \declare@shorthand{swedish}{"M}{\textormath{\bbl@disc M{MM}}{M}}
45.74 \declare@shorthand{swedish}{"n}{\textormath{\bbl@disc n{nn}}{n}}
45.75 \declare@shorthand{swedish}{"N}{\textormath{\bbl@disc N{NN}}{N}}
45.76 \declare@shorthand{swedish}{"p}{\textormath{\bbl@disc p{pp}}{p}}
45.77 \declare@shorthand{swedish}{"P}{\textormath{\bbl@disc P{PP}}{P}}
45.78 \declare@shorthand{swedish}{"r}{\textormath{\bbl@disc r{rr}}{r}}
45.79 \declare@shorthand{swedish}{"R}{\textormath{\bbl@disc R{RR}}{R}}
45.80 \declare@shorthand{swedish}{"s}{\textormath{\bbl@disc s{ss}}{s}}
45.81 \declare@shorthand{swedish}{"S}{\textormath{\bbl@disc S{SS}}{S}}
45.82 \declare@shorthand{swedish}{"t}{\textormath{\bbl@disc t{tt}}{t}}
45.83 \declare@shorthand{swedish}{"T}{\textormath{\bbl@disc T{TT}}{T}}
```

and some additional commands:

```
45.84 \declare@shorthand{swedish}{"-}{\nobreak-\bbl@allowhyphens}
45.85 \declare@shorthand{swedish}{"|}{\%}
45.86 \textormath{\nobreak\discretionary{-}{\kern.03em}%}
```

```

45.87 \bbl@allowhyphens}{}}
45.88 \declare@shorthand{swedish}{""}{\hskip\z@skip}
45.89 \declare@shorthand{swedish}{ "~ }{%
45.90 \textormath{\leavevmode\hbox{-}\bbl@allowhyphens}{-}}
45.91 \declare@shorthand{swedish}{ "="}{\hbox{-}\allowhyphens}

```

- \- Redefinition of \-. The new version of \- should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of T_EX in this respect is very unfortunate for languages such as Dutch, Finnish, German and Swedish, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that T_EX can generate from the hyphenation patterns.

```

45.92 \addto\extrasswedish{\babel@save\-\}
45.93 \addto\extrasswedish{\def\-\{\allowhyphens
45.94 \discretionary{-}{ }\allowhyphens}}

```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at \begin{document} and resetting the category code of @ to its original value.

```

45.95 \ldf@finish{swedish}
45.96 </code>

```

46 The North Sami language

The file `samin.dtx`⁵³ defines all the language definition macros for the North Sami language.

Several Sami dialects/languages are spoken in Finland, Norway, Sweden and on the Kola Peninsula (Russia). The alphabets differ, so there will eventually be a need for more `.dtx` files for e.g. Lule and South Sami. Hence the name `samin.dtx` (and not `sami.dtx` or the like) in the North Sami case.

There are currently no hyphenation patterns available for the North Sami language, but you might consider using the patterns for Finnish (`fi8hyph.tex`), Norwegian (`nohyph.tex`) or Swedish (`sehyph.tex`). Add a line for the `samin` language to the `language.dat` file, and rebuild the \LaTeX format file. See the documentation for your \LaTeX distribution.

A note on writing North Sami in \LaTeX : The TI encoding and EC fonts do not include the T WITH STROKE letter, which you will need a workaround for. My suggestion is to place the lines

```
\newcommand{\tx}{\mbox{t\hspace{-.35em}-}}
\newcommand{\txx}{\mbox{T\hspace{-.5em}-}}
in the preamble of your documents. They define the commands
\txx{} for LATIN CAPITAL LETTER T WITH STROKE and
\tx{} for LATIN SMALL LETTER T WITH STROKE.
```

46.1 The code of `samin.dtx`

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
46.1 {\code}
46.2 \LdfInit{samin}{captionssamin}
```

When this file is read as an option, i.e. by the `\usepackage` command, `samin` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@samin` to see whether we have to do something here.

```
46.3 \ifx\undefined\l@samin
46.4 \nopatterns{Samin}
46.5 \adddialect\l@samin0\fi
```

The next step consists of defining commands to switch to (and from) the North Sami language.

`\saminhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
46.6 \providehyphenmins{samin}{\tw@{\tw@}}
```

`\captionssamin` The macro `\captionssamin` defines all strings used in the four standard documentclasses provided with \LaTeX .

```
46.7 \def\captionssamin{%
46.8 \def\prefacename{Ovdas\'atni}%
46.9 \def\refname{\v Cujhusat}%
46.10 \def\abstractname{\v Coahkk\'aigeassu}%
46.11 \def\bibname{Girjj\'ala\v svuohta}%
46.12 \def\chaptername{Kapihttal}%
46.13 \def\appendixname{\v Cuovus}%
46.14 \def\contentsname{Sisdoallu}%
46.15 \def\listfigurename{Govvosat}%
46.16 \def\listtablename{Tabeallat}%
46.17 \def\indexname{Registtar}%
46.18 \def\figurename{Govus}%
```

⁵³The file described in this section has version number v1.0c and was last revised on 2004/02/20. It was written by Regnor Jernsletten, (Regnor.Jernsletten@sami.uit.no) or (Regnor.Jernsletten@eunet.no).

```

46.19 \def\tablename{Tabealla}%
46.20 \def\partname{Oassi}%
46.21 \def\enclname{Mielddus}%
46.22 \def\ccname{Kopia s\'addejuvvon}%
46.23 \def\headtoname{Vuost\'aiv\'aldi}%
46.24 \def\pagename{Siidu}%
46.25 \def\seenname{geah\v ca}%
46.26 \def\alsoname{geah\v ca maidd\'ai}%
46.27 \def\proofname{Duo\dj{a\v stus}%
46.28 \def\glossaryname{S\'atnelistu}%
46.29 }%

```

`\datesamin` The macro `\datesamin` redefines the command `\today` to produce North Sami dates.

```

46.30 \def\datesamin{%
46.31   \def\today{\ifcase\month\or
46.32     o\dj{\dj{a}jagem\'anu\or
46.33     guovvam\'anu\or
46.34     njuk\v cam\'anu\or
46.35     cuo\ng{om\'anu\or
46.36     miessem\'anu\or
46.37     geassem\'anu\or
46.38     suoidnem\'anu\or
46.39     borgem\'anu\or
46.40     \v cak\v cam\'anu\or
46.41     golggotm\'anu\or
46.42     sk\'abmam\'anu\or
46.43     juovlam\'anu\fi
46.44     \space\number\day.\~b.\space\number\year}%
46.45 }%

```

`\extrassamin` The macro `\extrassamin` will perform all the extra definitions needed for the North Sami language. The macro `\noextrassamin` is used to cancel the actions of `\extrassamin`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```

46.46 \addto\extrassamin{}
46.47 \addto\noextrassamin{}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

46.48 \ldf@finish{samin}
46.49 \code

```

47 The Finnish language

The file `finnish.dtx`⁵⁴ defines all the language definition macros for the Finnish language.

For this language the character " is made active. In table 21 an overview is given of its purpose.

"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
"=	an explicit hyphen sign for expressions such as "pakastekaapit ja -arkut".
""	like "-", but producing no hyphen sign (for words that should break at some sign such as "entrada/salida."
"‘	lowered double left quotes (looks like „)
"’	normal double right quotes
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).
\-	like the old \-, but allowing hyphenation in the rest of the word.

Table 21: The extra definitions made by `finnish.ldf`

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
47.1 (*code)
47.2 \LdfInit{finnish}\captionsofinnish
```

When this file is read as an option, i.e. by the `\usepackage` command, `finnish` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@finnish` to see whether we have to do something here.

```
47.3 \ifx\l@finnish\@undefined
47.4     \@nopatterns{Finnish}
47.5     \adddialect\l@finnish0\fi
```

The next step consists of defining commands to switch to the Finnish language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsofinnish` The macro `\captionsofinnish` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
47.6 \addto\captionsofinnish{%
47.7   \def\prefacename{Esipuhe}%
47.8   \def\refname{Viitteet}%
47.9   \def\abstractname{Tiivistelmä}%
47.10  \def\bibname{Kirjallisuutta}%
47.11  \def\chaptername{Luku}%
47.12  \def\appendixname{Liite}%
47.13  \def\contentsname{Sisällys}% /* Could be "Sisällysluettelo" as well */
47.14  \def\listfigurename{Kuvat}%
47.15  \def\listtablename{Taulukot}%
47.16  \def\indexname{Hakemisto}%
47.17  \def\figurename{Kuva}%
47.18  \def\tablename{Taulukko}%
47.19  \def\partname{Osa}%
47.20  \def\enclname{Liitteet}%
```

⁵⁴The file described in this section has version number v1.3q and was last revised on 2007/10/20. A contribution was made by Mikko KANERVA (KANERVA@CERNVM) and Keranen Reino (KERANEN@CERNVM).

```

47.21 \def\ccname{Jakelu}%
47.22 \def\headtoname{Vastaanottaja}%
47.23 \def\pagename{Sivu}%
47.24 \def\seename{katso}%
47.25 \def\alsoname{katso my\"os}%
47.26 \def\proofname{Todistus}%
47.27 \def\glossaryname{Sanasto}%
47.28 }%

```

`\datefinnish` The macro `\datefinnish` redefines the command `\today` to produce Finnish dates.

```

47.29 \def\datefinnish{%
47.30   \def\today{\number\day.\~\ifcase\month\or
47.31     tammikuuta\or helmikuuta\or maaliskuuta\or huhtikuuta\or
47.32     toukokuuta\or kes\"akuuta\or hein\"akuuta\or elokuuta\or
47.33     syyskuuta\or lokakuuta\or marraskuuta\or joulukuuta\fi
47.34     \space\number\year}}

```

`\extrasfinnish` Finnish has many long words (some of them compound, some not). For this reason hyphenation is very often the only solution in line breaking. For this reason the values of `\hyphenpenalty`, `\exhyphenpenalty` and `\doublehyphendemerits` should be decreased. (In one of the manuals of style Matti Rintala noticed a paragraph with ten lines, eight of which ended in a hyphen!)

Matti Rintala noticed that with these changes $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ handles Finnish very well, although sometimes the values of `\tolerance` and `\emergencystretch` must be increased. However, I don't think changing these values in `finnish.ldf` is appropriate, as the looseness of the font (and the line width) affect the correct choice of these parameters.

```

47.35 \addto\extrasfinnish{%
47.36   \babel@savevariable\hyphenpenalty\hyphenpenalty=30%
47.37   \babel@savevariable\exhyphenpenalty\exhyphenpenalty=30%
47.38   \babel@savevariable\doublehyphendemerits\doublehyphendemerits=5000%
47.39   \babel@savevariable\finalhyphendemerits\finalhyphendemerits=5000%
47.40 }
47.41 \addto\noextrasfinnish{}

```

Another thing `\extrasfinnish` needs to do is to ensure that `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasfinnish` will switch it off again.

```

47.42 \addto\extrasfinnish{\bbl@frenchspacing}
47.43 \addto\noextrasfinnish{\bbl@nonfrenchspacing}

```

For Finnish the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the finnish group of shorthands should be used.

```

47.44 \initiate@active@char{"}
47.45 \addto\extrasfinnish{\languageshorthands{finnish}}

```

Don't forget to turn the shorthands off again.

```

47.46 \addto\extrasfinnish{\bbl@activate{}}
47.47 \addto\noextrasfinnish{\bbl@deactivate{}}

```

The 'umlaut' character should be positioned lower on *all* vowels in Finnish texts.

```

47.48 \addto\extrasfinnish{\umlautlow\umlautelaw}
47.49 \addto\noextrasfinnish{\umlauthigh}

```

First we define access to the low opening double quote and guillemets for quotations,

```

47.50 \declare@shorthand{finnish}{"'}{%
47.51   \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
47.52 \declare@shorthand{finnish}{"'}{%

```

```

47.53 \textormath{\textquotedblright}{\mbox{\textquotedblright}}
47.54 \declare@shorthand{finnish}{"<"}{%
47.55 \textormath{\guillemotleft}{\mbox{\guillemotleft}}
47.56 \declare@shorthand{finnish}{">"}{%
47.57 \textormath{\guillemotright}{\mbox{\guillemotright}}

```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```

47.58 \declare@shorthand{finnish}{"-"}{\nobreak-\bbl@allowhyphens}
47.59 \declare@shorthand{finnish}{""}{\hskip\z@skip}
47.60 \declare@shorthand{finnish}{ "="}{\hbox{-}\bbl@allowhyphens}

```

And we want to have a shorthand for disabling a ligature.

```

47.61 \declare@shorthand{finnish}{"|"}{%
47.62 \textormath{\discretionary{-}{-}{\kern.03em}}{}}

```

\- All that is left now is the redefinition of \-. The new version of \- should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of T_EX in this respect is very unfortunate for languages such as Dutch, Finnish and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that T_EX can generate from the hyphenation patterns.

```

47.63 \addto\extrasfinnish{\babel@save\-\}
47.64 \addto\extrasfinnish{\def\-\{\bbl@allowhyphens
47.65 \discretionary{-}{-}{\bbl@allowhyphens}}

```

\finishhyphenmins The finnish hyphenation patterns can be used with \lefthyphenmin set to 2 and \righthyphenmin set to 2.

```

47.66 \providehyphenmins{\CurrentOption}{\tw@ \tw@}

```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at \begin{document} and resetting the category code of @ to its original value.

```

47.67 \ldf@finish{finnish}
47.68 \code>

```

48 The Hungarian language

The file option `magyar.dtx` defines all the language definition macros for the Hungarian language.

The `babel` support for the Hungarian language until file version 1.3i was essentially changing the English document elements to Hungarian ones, but because of the differences between these two languages this was actually unusable (‘Part I’ was transferred to ‘Rész I’ which is not usable instead of ‘I. rész’). To enhance the typesetting facilities for Hungarian the following should be considered:

- In Hungarian documents there is a period after the part, section, subsection etc. numbers.
- In the part, chapter, appendix name the number (or letter) goes before the name, so ‘Part I’ translates to ‘I. rész’.
- The same is true with captions (‘Table 2.1’ goes to ‘2.1. táblázat’).
- There is a period after the caption name instead of a colon. (‘Table 2.1:’ goes to ‘2.1. táblázat.’)
- There is a period at the end of the title in a run-in head (when `afterskip<0` in `\@startsection`).
- Special hyphenation rules must be applied for the so-called long double consonants (`ccs`, `ssz`, ...).
- The opening quotation mark is like the German one (the closing is the same as in English).
- In Hungarian figure, table, etc. referencing a definite article is also incorporated. The Hungarian definite articles behave like the English indefinite ones (‘a/an’). ‘a’ is used for words beginning with a consonant and ‘az’ goes for a vowel. Since some numbers begin with a vowel some others with a consonant some commands should be provided for automatic definite article generation.

Until file version 1.3i⁵⁵ the special typesetting rules of the Hungarian language mentioned above were not taken into consideration. This version (v1.4j)⁵⁶ enables `babel` to typeset ‘good-looking’ Hungarian texts.

<code>\ontoday</code>	The <code>\ontoday</code> command works like <code>\today</code> but produces a slightly different date format used in expressions such as ‘on February 10th’.
<code>\Az</code>	The commands <code>\Az#1</code> and <code>\az#1</code> write the correct definite article for the argument and the argument itself (separated with a <code>~</code>). The star-forms (<code>\Az*</code> and <code>\az*</code>) produce the article only.
<code>\Azr</code>	<code>\Azr#1</code> and <code>\azr#1</code> treat the argument as a label so expand it then write the definite article for <code>\r@#1</code> , a non-breakable space then the label expansion. The star-forms do not print the label expansion. <code>\Azr(#1</code> and <code>\azr(#1</code> are used for equation referencing with the syntax <code>\azr{label}</code> .
<code>\Aref</code>	There are two aliases <code>\Aref</code> and <code>\aref</code> for <code>\Azr</code> and <code>\azr</code> , respectively. During the preparation of a document it is not known in general, if the code ‘ <code>a~\ref{label}</code> ’ or the code ‘ <code>az~\ref{label}</code> ’ is the grammatically correct one. Writing ‘ <code>\aref{label}</code> ’ instead of the previous ones solves the problem.
<code>\Azp</code>	<code>\Azp#1</code> and <code>\azp#1</code> also treat the argument as a label but use the label’s page for definite article determination. There are star-forms giving only the definite article without the page number.
<code>\Apageref</code>	There are aliases <code>\Apageref</code> and <code>\apageref</code> for <code>\Azp</code> and <code>\azp</code> , respec-

⁵⁵That file was last revised on 1996/12/23 with a contribution by the next authors: Attila Koppányi (attila@cernvm.cern.ch), Árpád Bíró (JZP1104@HUSZEG11.bitnet), István Hamecz (hami@ursus.bke.hu) and Dezső Horváth (horvath@pisa.infn.it).

⁵⁶It was written by József Bérces (jozsi@docs4.mht.bme.hu) with some help from Ferenc Wettl (wettl@math.bme.hu) and an idea from David Carlisle (david@dcarlisle.demon.co.uk).

shortcut	explanation	example
‘	same as <code>\glqq</code> in <code>babel</code> , or <code>\quotedblbase</code> in T1 (opening quotation mark, like „)	‘‘id\’ezet’’ \rightarrow „idézet”
‘c, ‘C	ccs is hyphenated as cs-cs	lo‘ccsan \rightarrow locs-csan
‘d, ‘D	ddz is hyphenated as dz-dz	e‘ddz\“unk \rightarrow edz-dzünk
‘g, ‘G	ggy is hyphenated as gy-gy	po‘ggy\’asz \rightarrow pogy-gyász
‘l, ‘L	lly is hyphenated as ly-ly	Kod\’a‘llyal \rightarrow Kodály-lyal
‘n, ‘N	nny is hyphenated as ny-ny	me‘nnyei \rightarrow meny-nyei
‘s, ‘S	ssz is hyphenated as sz-sz	vi‘ssza \rightarrow visz-sza
‘t, ‘T	tty is hyphenated as ty-ty	po‘ttyan \rightarrow poty-tyan
‘z, ‘Z	zsz is hyphenated as zs-zs	ri‘zzsel \rightarrow rizs-zsel

Table 22: The shortcuts defined in `magyar.ldf`

tively. The code `\apageref{label}` is equivalent either to `a~\pageref{label}` or to `az~\pageref{label}`.

`\Azc` `\Azc` and `\azc` work like the `\cite` command but (of course) they insert the definite article. There can be several comma separated cite labels and in that case the definite article is given for the first one. They accept `\cite`’s optional argument. There are star-forms giving the definite article only.

`\Acite` There are aliases `\Acite` and `\acite` for `\Azc` and `\azc`, respectively. For this language the character ‘ is made active. Table 22 shows the shortcuts. The main reason for the activation of the ‘ character is to handle the special hyphenation of the long double consonants.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
48.1 (*code)
48.2 \LdfInit{magyar}{caption\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `magyar` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@magyar` or `\l@hungarian` to see whether we have to do something here.

```
48.3 \ifx\l@magyar\@undefined
48.4   \ifx\l@hungarian\@undefined
48.5     \nopatterns{Magyar}
48.6     \adddialect\l@magyar0
48.7   \else
48.8     \let\l@magyar\l@hungarian
48.9   \fi
48.10 \fi
```

The statement above makes sure that `\l@magyar` is always defined; if `\l@hungarian` is still undefined we make it equal to `\l@magyar`.

```
48.11 \ifx\l@hungarian\@undefined
48.12   \let\l@hungarian\l@magyar
48.13 \fi
```

The next step consists of defining commands to switch to (and from) the Hungarian language.

`\captionsmagyar` The macro `\captionsmagyar` defines all strings used in the four standard document classes provided with L^AT_EX.

```
48.14 \@namedef{captions\CurrentOption}{%
48.15   \def\prefacename{El\H osz\’o}%
```

For the list of references at the end of an article we have a choice between two words, ‘Referenciák’ (a Hungarian version of the English word) and ‘Hivatkozások’. The latter seems to be in more widespread use.

48.16 \def\refname{Hivatkoz\'}asok}%

If you have a document with a summary instead of an abstract you might want to replace the word 'Kivonat' with 'Összefoglaló'.

48.17 \def\abstractname{Kivonat}%

The Hungarian version of 'Bibliography' is 'Bibliográfia', but a more natural word to use is 'Irodalomjegyzék'.

48.18 \def\bibname{Irodalomjegyz\'}ek}%

48.19 \def\chaptername{fejezet}%

48.20 \def\appendixname{F\'}uggel\'}ek}%

48.21 \def\contentsname{Tartalomjegyz\'}ek}%

48.22 \def\listfigurename{\'}Abr\'}ak jegyz\'}eke}%

48.23 \def\listtablename{T\'}abl\'}azatok jegyz\'}eke}%

48.24 \def\indexname{T\'}argymutat\'}o}%

48.25 \def\figurename{\'}abra}%

48.26 \def\tablename{t\'}abl\'}azat}%

48.27 \def\partname{r\'}esz}%

48.28 \def\enclname{Mell\'}eklet}%

48.29 \def\ccname{K\'}orlev\'}el--c\'}i mzetek}%

48.30 \def\headtoname{C\'}i mzett}%

48.31 \def\pagename{oldal}%

48.32 \def\seenname{l\'}asd}%

48.33 \def\alsiname{l\'}asd m\'}eg}%

Besides the Hungarian word for Proof, 'Bizonyítás' we can also name Corollary (Következmény), Theorem (Tétel) and Lemma (Lemma).

48.34 \def\proofname{Bizony\'}i t\'}as}%

48.35 \def\glossaryname{Sz\'}ojegyz\'}ek}%

48.36 }%

\datemagyar The macro \datemagyar redefines the command \today to produce Hungarian dates.

48.37 \@namedef{date\CurrentOption}{%

48.38 \def\today{%

48.39 \number\year.\nobreakspace\ifcase\month\or

48.40 janu\'}ar\'}or febru\'}ar\'}or m\'}arcius\'}or

48.41 \'}aprilis\'}or m\'}ajus\'}or j\'}unius\'}or

48.42 j\'}ulius\'}or augusztus\'}or szeptember\'}or

48.43 okt\'}ober\'}or november\'}or december\'}fi

48.44 \space\number\day.}}

\ondatemagyar The macro \ondatemagyar produces Hungarian dates which have the meaning 'on this day'. It does not redefine the command \today.

48.45 \@namedef{ondate\CurrentOption}{%

48.46 \number\year.\nobreakspace\ifcase\month\or

48.47 janu\'}ar\'}or febru\'}ar\'}or m\'}arcius\'}or

48.48 \'}aprilis\'}or m\'}ajus\'}or j\'}unius\'}or

48.49 j\'}ulius\'}or augusztus\'}or szeptember\'}or

48.50 okt\'}ober\'}or november\'}or december\'}fi

48.51 \space\ifcase\day\or

48.52 1-j\'}en\'}or 2-\'}an\'}or 3-\'}an\'}or 4-\'}en\'}or 5-\'}en\'}or

48.53 6-\'}an\'}or 7-\'}en\'}or 8-\'}an\'}or 9-\'}en\'}or 10-\'}en\'}or

48.54 11-\'}en\'}or 12-\'}en\'}or 13-\'}an\'}or 14-\'}en\'}or 15-\'}en\'}or

48.55 16-\'}an\'}or 17-\'}en\'}or 18-\'}an\'}or 19-\'}en\'}or 20-\'}an\'}or

48.56 21-\'}en\'}or 22-\'}en\'}or 23-\'}an\'}or 24-\'}en\'}or 25-\'}en\'}or

48.57 26-\'}an\'}or 27-\'}en\'}or 28-\'}an\'}or 29-\'}en\'}or 30-\'}an\'}or

48.58 31-\'}en\'}fi}

\extrasmagyar The macro \extrasmagyar will perform all the extra definitions needed for the Hungarian language. The macro \noextrasmagyar is used to cancel the actions of \extrasmagyar.

48.59 \@namedef{extras\CurrentOption}{%

```

48.60 \expandafter\let\expandafter\ontoday
48.61 \csname ondate\CurrentOption\endcsname}
48.62 \@namedef{noextras\CurrentOption}{\let\ontoday\@undefined}

```

Now we redefine some commands included into `latex.ltx`. The original form of a command is always saved with `\babel@save` and the changes are added to `\extrasmagyar`. This ensures that the Hungarian version of a macro is alive *only* if the Hungarian language is active.

`\fnum@figure` In figure and table captions the order of the figure/table number and `\figurename`
`\fnum@table` `/\tablename` must be changed. To achieve this `\fnum@figure` and `\fnum@table` are redefined and added to `\extrasmagyar`.

```

48.63 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.64 \babel@save\fnum@figure
48.65 \def\fnum@figure{\thefigure.\nobreakspace\figurename}}
48.66 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.67 \babel@save\fnum@table
48.68 \def\fnum@table{\thetable.\nobreakspace\tablename}}

```

`\@makecaption` The colon in a figure/table caption must be replaced by a dot by redefining `\@makecaption`.

```

48.69 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.70 \babel@save\@makecaption
48.71 \def\@makecaption#1#2{%
48.72 \vskip\abovcaptionskip
48.73 \sbox\@tempboxa{#1. #2}%
48.74 \ifdim \wd\@tempboxa >\hsize
48.75 {#1. #2\csname par\endcsname}
48.76 \else
48.77 \global \@minipagefalse
48.78 \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
48.79 \fi
48.80 \vskip\belowcaptionskip}}

```

`\@caption` There should be a dot after the figure/table number in `lof/lot`, so `\@caption` is redefined.

```

48.81 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.82 \babel@save\@caption
48.83 \long\def\@caption#1[#2]#3{%
48.84 \csname par\endcsname
48.85 \addcontentsline{\csname ext@#1\endcsname}{#1}%
48.86 {\protect\numberline{\csname the#1\endcsname.}\ignorespaces #2}}%
48.87 \begingroup
48.88 \@parboxrestore
48.89 \if@minipage
48.90 \setminipage
48.91 \fi
48.92 \normalsize
48.93 \@makecaption{\csname fnum@#1\endcsname}%
48.94 {\ignorespaces #3}\csname par\endcsname
48.95 \endgroup}}

```

`\@secntformat` In order to have a dot after the section number `\@secntformat` is redefined.

```

48.96 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.97 \babel@save\@secntformat
48.98 \def\@secntformat#1{\csname the#1\endcsname.\quad}}

```

`\@sect` Alas, `\@sect` must also be redefined to have that dot in `toc` too. On the other hand, we include a dot after a run-in head.

```

48.99 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.100 \babel@save\@sect

```

```

48.101 \def\@sect#1#2#3#4#5#6[#7]#8{%
48.102   \ifnum #2>\c@secnumdepth
48.103     \let\@svsec\@empty
48.104   \else
48.105     \refstepcounter{#1}%
48.106     \protected@edef\@svsec{\@secntformat{#1}\relax}%
48.107   \fi
48.108   \@tempskipa #5\relax
48.109   \ifdim \@tempskipa>\z@
48.110     \begingroup
48.111       #6{%
48.112         \@hangfrom{\hskip #3\relax\@svsec}%
48.113         \interlinepenalty \@M #8\@@par}%
48.114     \endgroup
48.115     \csname #1mark\endcsname{#7}%
48.116     \addcontentsline{toc}{#1}{%
48.117       \ifnum #2>\c@secnumdepth \else
48.118         \protect\numberline{\csname the#1\endcsname.}%
48.119       \fi
48.120       #7}%
48.121   \else
48.122     \def\@svsechd{%
48.123       #6{\hskip #3\relax
48.124         \@svsec #8.}%
48.125       \csname #1mark\endcsname{#7}%
48.126       \addcontentsline{toc}{#1}{%
48.127         \ifnum #2>\c@secnumdepth \else
48.128           \protect\numberline{\csname the#1\endcsname.}%
48.129         \fi
48.130         #7}}%
48.131   \fi
48.132   \@xsect{#5}}

```

`\@ssect` In order to have that dot after a run-in head when the star form of the sectioning commands is used, we have to redefine `\@ssect`.

```

48.133 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.134   \babel@save\@ssect
48.135   \def\@ssect#1#2#3#4#5{%
48.136     \@tempskipa #3\relax
48.137     \ifdim \@tempskipa>\z@
48.138       \begingroup
48.139         #4{%
48.140           \@hangfrom{\hskip #1}%
48.141           \interlinepenalty \@M #5\@@par}%
48.142       \endgroup
48.143     \else
48.144       \def\@svsechd{#4{\hskip #1\relax #5.}}%
48.145     \fi
48.146     \@xsect{#3}}

```

`\@begintheorem` Order changing and dot insertion in theorem by redefining `\@begintheorem` and `\@opargbegintheorem`

```

48.147 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.148   \babel@save\@begintheorem
48.149   \def\@begintheorem#1#2{\trivlist
48.150     \item[\hskip \labelsep{\bfseries #2.\ #1.}]{\itshape}%
48.151   \babel@save\@opargbegintheorem
48.152   \def\@opargbegintheorem#1#2#3{\trivlist
48.153     \item[\hskip \labelsep{\bfseries #2.\ #1\ (#3).}]{\itshape}}

```

The next step is to redefine some macros included into the class files. It is determined which class file is loaded then the original form of the macro is saved and the changes are added to `\extrasmagyar`.

First we check if the book.cls is loaded.

```
48.154 \ifclassloaded{book}{%
```

`\ps@headings` The look of the headings is changed: we have to insert some dots and change the order of chapter number and `\chaptername`.

```
48.155 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.156 \babel@save\ps@headings}
48.157 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.158 \if@twoside
48.159 \def\ps@headings{%
48.160 \let\@oddfoot\@empty\let\@evenfoot\@empty
48.161 \def\@evenhead{\thepage\hfil\slshape\leftmark}%
48.162 \def\@oddhead{\slshape\rightmark}\hfil\thepage}%
48.163 \let\@mkboth\markboth
48.164 \def\chaptermark##1{%
48.165 \markboth {\MakeUppercase{%
48.166 \ifnum \c@secnumdepth >\m@ne
48.167 \if@mainmatter
48.168 \thechapter. \@chapapp. \ %
48.169 \fi
48.170 \fi
48.171 ##1}}{}}%
48.172 \def\sectionmark##1{%
48.173 \markright {\MakeUppercase{%
48.174 \ifnum \c@secnumdepth >\z@
48.175 \thesection. \ %
48.176 \fi
48.177 ##1}}}%
48.178 \else
48.179 \def\ps@headings{%
48.180 \let\@oddfoot\@empty
48.181 \def\@oddhead{\slshape\rightmark}\hfil\thepage}%
48.182 \let\@mkboth\markboth
48.183 \def\chaptermark##1{%
48.184 \markright {\MakeUppercase{%
48.185 \ifnum \c@secnumdepth >\m@ne
48.186 \if@mainmatter
48.187 \thechapter. \@chapapp. \ %
48.188 \fi
48.189 \fi
48.190 ##1}}}%
48.191 \fi}
```

`\@part` At the beginning of a part we need eg. ‘I. rész’ instead of ‘Part I’ (in toc too). To achieve this `\@part` is redefined.

```
48.192 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.193 \babel@save\@part
48.194 \def\@part[#1]#2{%
48.195 \ifnum \c@secnumdepth >-2\relax
48.196 \refstepcounter{part}%
48.197 \addcontentsline{toc}{part}{\thepart.\hspace{1em}#1}%
48.198 \else
48.199 \addcontentsline{toc}{part}{#1}%
48.200 \fi
48.201 \markboth{}{}%
48.202 {\centering
48.203 \interlinepenalty \@M
48.204 \normalfont
48.205 \ifnum \c@secnumdepth >-2\relax
48.206 \huge\bfseries \thepart.\nobreakspace\partname
48.207 \csname par\endcsname
48.208 \vskip 20\p@}
```

```

48.209         \fi
48.210         \Huge \bfseries #2\csname par\endcsname}%
48.211     \@endpart}}

```

`\@chapter` The same changes are made to chapter. First the screen typeout and the toc are changed by redefining `\@chapter`.

```

48.212 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.213     \babel@save\@chapter
48.214     \def\@chapter[#1]#2{\ifnum \c@secnumdepth >\m@ne
48.215         \if@mainmatter
48.216             \refstepcounter{chapter}%
48.217             \typeout{\thechapter.\space\@chapapp.}%
48.218             \addcontentsline{toc}{chapter}%
48.219                 {\protect\numberline{\thechapter.}#1}%
48.220         \else
48.221             \addcontentsline{toc}{chapter}{#1}%
48.222         \fi
48.223     \else
48.224         \addcontentsline{toc}{chapter}{#1}%
48.225     \fi
48.226     \chaptermark{#1}%
48.227     \addtocontents{lof}{\protect\addvspace{10\p@}}%
48.228     \addtocontents{lot}{\protect\addvspace{10\p@}}%
48.229     \if@twocolumn
48.230         \topnewpage[\@makechapterhead{#2}]%
48.231     \else
48.232         \@makechapterhead{#2}%
48.233         \@afterheading
48.234     \fi}}

```

`\@makechapterhead` Then the look of the chapter-start is modified by redefining `\@makechapterhead`.

```

48.235 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.236     \babel@save\@makechapterhead
48.237     \def\@makechapterhead#1{%
48.238         \vspace*{50\p@}%
48.239         {\parindent \z@ \raggedright \normalfont
48.240         \ifnum \c@secnumdepth >\m@ne
48.241             \if@mainmatter
48.242                 \huge\bfseries \thechapter.\nobreakspace\@chapapp{}
48.243                 \csname par\endcsname\nobreak
48.244                 \vskip 20\p@
48.245             \fi
48.246         \fi
48.247         \interlinepenalty\@M
48.248         \Huge \bfseries #1\csname par\endcsname\nobreak
48.249         \vskip 40\p@
48.250     }}}}

```

This the end of the book class modification.

```
48.251 }{}
```

Now we check if `report.cls` is loaded.

```
48.252 \@ifclassloaded{report}{%
```

`\ps@headings` First the headings are modified just in case of the book class.

```

48.253 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.254     \babel@save\ps@headings}
48.255 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.256     \if@twoside
48.257         \def\ps@headings{%
48.258             \let\@oddfoot\@empty\let\@evenfoot\@empty
48.259             \def\@evenhead{\thepage\hfil\slshape\leftmark}%

```

```

48.260         \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
48.261         \let\@mkboth\markboth
48.262     \def\chaptermark##1{%
48.263         \markboth {\MakeUppercase{%
48.264             \ifnum \c@secnumdepth >\m@ne
48.265                 \thechapter. \@chapapp. \ %
48.266             \fi
48.267             ##1}}{}}%
48.268     \def\sectionmark##1{%
48.269         \markright {\MakeUppercase{%
48.270             \ifnum \c@secnumdepth >\z@
48.271                 \thesection. \ %
48.272             \fi
48.273             ##1}}}%
48.274     \else
48.275         \def\ps@headings{%
48.276             \let\@oddfoot\@empty
48.277             \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
48.278             \let\@mkboth\markboth
48.279             \def\chaptermark##1{%
48.280                 \markright {\MakeUppercase{%
48.281                     \ifnum \c@secnumdepth >\m@ne
48.282                         \thechapter. \@chapapp. \ %
48.283                     \fi
48.284                     ##1}}}%
48.285             \fi}

```

`\@chapter` Chapter-start modification with `\@chapter`

```

48.286     \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.287         \babel@save\@chapter
48.288         \def\@chapter[#1]#2{\ifnum \c@secnumdepth >\m@ne
48.289             \refstepcounter{chapter}%
48.290             \typeout{\thechapter.\space\@chapapp.}%
48.291             \addcontentsline{toc}{chapter}%
48.292                 {\protect\numberline{\thechapter.}#1}%
48.293             \else
48.294                 \addcontentsline{toc}{chapter}{#1}%
48.295             \fi
48.296             \chaptermark{#1}%
48.297             \addtocontents{lof}{\protect\addvspace{10\p@}}%
48.298             \addtocontents{lot}{\protect\addvspace{10\p@}}%
48.299             \if@twocolumn
48.300                 \@topnewpage[\@makechapterhead{#2}]%
48.301             \else
48.302                 \@makechapterhead{#2}%
48.303                 \@afterheading
48.304             \fi}}

```

`\@makechapterhead` and `\@makechapterhead.`

```

48.305     \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.306         \babel@save\@makechapterhead
48.307         \def\@makechapterhead#1{%
48.308             \vspace*{50\p@}%
48.309             {\parindent \z@ \raggedright \normalfont
48.310                 \ifnum \c@secnumdepth >\m@ne
48.311                     \huge\bfseries \thechapter.\nobreakspace\@chapapp{}
48.312                     \csname par\endcsname\nobreak
48.313                     \vskip 20\p@
48.314                 \fi
48.315                 \interlinepenalty\@M
48.316                 \Huge \bfseries #1\csname par\endcsname\nobreak
48.317                 \vskip 40\p@
48.318             }}}}

```

End of report class modification.

48.319 }{}

Checking if article.cls is loaded.

48.320 \@ifclassloaded{article}{%

\ps@headings Changing headings by redefining \ps@headings.

```
48.321 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.322 \babel@save\ps@headings}
48.323 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.324 \if@twoside
48.325 \def\ps@headings{%
48.326 \let\@oddfoot\@empty\let\@evenfoot\@empty
48.327 \def\@evenhead{\thepage\hfil\slshape\leftmark}%
48.328 \def\@oddhead{\slshape\rightmark}\hfil\thepage}%
48.329 \let\@mkboth\markboth
48.330 \def\sectionmark##1{%
48.331 \markboth {\MakeUppercase{%
48.332 \ifnum \c@secnumdepth >\z@
48.333 \thesection.\quad
48.334 \fi
48.335 ##1}}{}}%
48.336 \def\subsectionmark##1{%
48.337 \markright {%
48.338 \ifnum \c@secnumdepth >\@ne
48.339 \thesubsection.\quad
48.340 \fi
48.341 ##1}}}%
48.342 \else
48.343 \def\ps@headings{%
48.344 \let\@oddfoot\@empty
48.345 \def\@oddhead{\slshape\rightmark}\hfil\thepage}%
48.346 \let\@mkboth\markboth
48.347 \def\sectionmark##1{%
48.348 \markright {\MakeUppercase{%
48.349 \ifnum \c@secnumdepth >\m@ne
48.350 \thesection.\quad
48.351 \fi
48.352 ##1}}}%
48.353 \fi}%
```

No more necessary changes specific to the article class.

48.354 }{}

And now this is the turn of letter.cls.

48.355 \@ifclassloaded{letter}{%

\ps@headings In the headings the page number must be followed by a dot and then \pagename.

```
48.356 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.357 \babel@save\ps@headings}
48.358 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.359 \if@twoside
48.360 \def\ps@headings{%
48.361 \let\@oddfoot\@empty\let\@evenfoot\@empty
48.362 \def\@oddhead{\slshape\headtoname{:} \ignorespaces\toname
48.363 \hfil \@date
48.364 \hfil \thepage.\nobreakspace\pagename}%
48.365 \let\@evenhead\@oddhead}
48.366 \else
48.367 \def\ps@headings{%
48.368 \let\@oddfoot\@empty
48.369 \def\@oddhead{\slshape\headtoname{:} \ignorespaces\toname
48.370 \hfil \@date
```



```

48.371 \hfil \thepage.\nobreakspace\pagename}}
48.372 \fi}%

```

End of letter class.

```

48.373 }{}

```

After making the changes to the L^AT_EX macros we define some new ones to handle the problem with definite articles.

`\az` `\az` is a user-level command which decides if the next character is a star. `\@az` is called for `\az*` and `\az@` for `\az`.

```

48.374 \def\az{a\@ifstar{\@az}{\az@}}

```

`\Az` `\Az` is used at the beginning of a sentence. Otherwise it behaves the same as `\az`.

```

48.375 \def\Az{A\@ifstar{\@az}{\az@}}

```

`\az@` `\az@` is called if there is no star after `\az` or `\Az`. It calls `\@az` and writes #1 separating with a non-breakable space.

```

48.376 \def\az@#1{\@az{#1}\nobreakspace#1}

```

`\@az` This macro calls `\hun@tempdef` to remove the accents from the argument then calls `\@@az` that determines if a ‘z’ should be written after a/A (written by `\az/\Az`).

```

48.377 \def\@az#1{%
48.378   \hun@tempdef{relax}{relax}{#1}%
48.379   \edef\@tempb{\noexpand\@@az\@tempa\hbox!}%
48.380   \@tempb}

```

`\hun@tempdef` The macro `\hun@tempdef` has three tasks:

- Accent removal. Accented letters confuse `\@@az`, the main definite article determinator macro, so they must be changed to their non-accented counterparts. Special letters must also be changed, eg. $\text{œ} \rightarrow \text{o}$.
- Labels must be expanded.
- To handle Roman numerals correctly, commands starting with `\hun@` are defined for labels containing Roman numbers with the Roman numerals replaced by their Arabic representation. This macro can check if there is a `\hun@` command.

There are three arguments:

1. The primary command that should be expanded if it exists. This is usually the `\hun@` command for a label.
2. The secondary command which is used if the first one is `\relax`. This is usually the original L^AT_EX command for a label.
3. This is used if the first two is `\relax`. For this one no expansion is carried out but the accents are still removed and special letters are changed.

```

48.381 \def\hun@tempdef#1#2#3{%
48.382   \begingroup
48.383   \def\@safe@activesfalse{}%
48.384   \def\setbox #1{}% to get rid of accents and special letters
48.385   \def\hbox #1{}%
48.386   \def\accent ##1 ##2{##2}%
48.387   \def\add@accent ##1##2{##2}%
48.388   \def\@text@composite@x ##1##2{##2}%
48.389   \def\i{i}\def\j{j}%
48.390   \def\ae{a}\def\AE{A}\def\oe{o}\def\OE{O}%
48.391   \def\ss{s}\def\L{L}%

```

```

48.392 \def\d{}\def\b{}\def\c{}\def\t{}%
48.393 \expandafter\ifx\csname #1\endcsname\relax
48.394 \expandafter\ifx\csname #2\endcsname\relax
48.395 \xdef\@tempa{#3}%
48.396 \else
48.397 \xdef\@tempa{\csname #2\endcsname}%
48.398 \fi
48.399 \else
48.400 \xdef\@tempa{\csname #1\endcsname}%
48.401 \fi
48.402 \endgroup}

```

The following macros are used to determine the definite article for a label's expansion.

`\aref` `\aref` is an alias for `\azr`.

```
48.403 \def\aref{\azr}
```

`\Aref` `\Aref` is an alias for `\Azr`.

```
48.404 \def\Aref{\Azr}
```

`\azr` `\azr` calls `\@azr` if the next character is a star, otherwise it calls `\azr@`.

```
48.405 \def\azr{a\@ifstar{\@azr}{\azr@}}
```

`\Azr` `\Azr` is the same as `\azr` except that it writes 'A' instead of 'a'.

```
48.406 \def\Azr{A\@ifstar{\@azr}{\azr@}}
```

`\azr@` `\azr@` decides if the next character is (and in that case it calls `\azr@@@` which writes an extra (for equation referencing. Otherwise `\azr@@` is called.

```
48.407 \def\azr@{\@ifnextchar ({\azr@@@}{\azr@@}}
```

`\azr@@` Calls `\@azr` then writes the label's expansion preceded by a non-breakable space.

```
48.408 \def\azr@@#1{\@azr{#1}\nobreakspace\ref{#1}}
```

`\azr@@@` Same as `\azr@@` but inserts a (between the non-breakable space and the label expansion.

```
48.409 \def\azr@@@(#1{\@azr{#1}\nobreakspace(\ref{#1})}
```

`\@azr` Calls `\hun@tempadef` to choose between the label's `\hun@` or original L^AT_EX command and to expand it with accent removal and special letter substitution. Then calls `\@@az`, the core macro of definite article handling.

```

48.410 \def\@azr#1{%
48.411 \hun@tempadef{hun@r@#1}{r@#1}{}%
48.412 \ifx\@tempa\empty
48.413 \else
48.414 \edef\@tempb{\noexpand\@@az\expandafter\@firstoftwo\@tempa\hbox!}%
48.415 \@tempb
48.416 \fi
48.417 }

```

The following commands are used to generate the definite article for the page number of a label.

`\apageref` `\apageref` is an alias for `\azp`.

```
48.418 \def\apageref{\azp}
```

`\Apageref` `\Apageref` is an alias for `\Azp`.

```
48.419 \def\Apageref{\Azp}
```

`\azp` Checks if the next character is * and calls `\@azp` or `\azp@`.

```
48.420 \def\azp{a\@ifstar{\@azp}{\azp@}}
```

`\Azp` Same as `\azp` except that it writes ‘A’ instead of ‘a’.

```
48.421 \def\Azp{A@ifstar{\@azp}{\azp@}}
```

`\azp@` Calls `\@azp` then writes the label’s page preceded by a non-breakable space.

```
48.422 \def\azp@#1{\@azp{#1}\nobreakspace\pageref{#1}}
```

`\@azp` Calls `\hun@tempadef` then takes the label’s page and passes it to `\@@az`.

```
48.423 \def\@azp#1{%
48.424   \hun@tempadef{hun@r@#1}{r@#1}{}%
48.425   \ifx\@tempa\empty
48.426   \else
48.427     \edef\@tempb{\noexpand\@@az\expandafter\@secondoftwo\@tempa\hbox!}%
48.428     \@tempb
48.429   \fi
48.430 }
```

The following macros are used to give the definite article to citations.

`\acite` This is an alias for `\azc`.

```
48.431 \def\acite{\azc}
```

`\Acite` This is an alias for `\Azc`.

```
48.432 \def\Acite{\Azc}
```

`\azc` Checks if the next character is a star and calls `\@azc` or `\azc@`.

```
48.433 \def\azc{a@ifstar{\@azc}{\azc@}}
```

`\Azc` Same as `\azc` but used at the beginning of sentences.

```
48.434 \def\Azc{A@ifstar{\@azc}{\azc@}}
```

`\azc@` If there is no star we accept an optional argument, just like the `\cite` command.

```
48.435 \def\azc@{\@ifnextchar [{\azc@@}{\azc@[]}}
```

`\azc@@` First calls `\@azc` then `\cite`.

```
48.436 \def\azc@@[#1]#2{%
48.437   \@azc{#2}\nobreakspace\def\@tempa{#1}%
48.438   \ifx\@tempa\@empty\cite{#2}\else\cite{#1}{#2}\fi}
```

`\@azc` This is an auxiliary macro to get the first cite label from a comma-separated list.

```
48.439 \def\@azc#1{\@azc#1,\hbox!}
```

`\@@azc` This one uses only the first argument, that is the first element of the comma-separated list of cite labels. Calls `\hun@tempadef` to expand the cite label with accent removal and special letter replacement. Then `\@@az`, the core macro, is called.

```
48.440 \def\@@azc#1,#2\hbox#3!{%
48.441   \hun@tempadef{hun@b@#1}{b@#1}{}%
48.442   \ifx\@tempa\empty
48.443   \else
48.444     \edef\@tempb{\noexpand\@@az\@tempa\hbox!}%
48.445     \@tempb
48.446   \fi}
```

`\hun@number@leghth` This macro is used to count the number of digits in its argument until a non-digit character is found or the end of the argument is reached. It must be called as `\hun@number@leghtharg\hbox\hbox!` and `\count@` must be zeroed. It is called by `\@@az`.

```
48.447 \def\hun@number@leghth#1#2\hbox#3!{%
48.448   \ifcat\noexpand#1%
48.449     \ifnum\expandafter'\csname#1\endcsname>47
48.450     \ifnum\expandafter'\csname#1\endcsname<58
48.451       \advance\count@ by \@ne
48.452       \hun@number@leghth#2\hbox\hbox!\fi\fi\fi}
```

`\hun@alph@lehgth` This is used to count the number of letters until a non-letter is found or the end of the argument is reached. It must be called as `\hun@alph@lehgtharg\hbox\hbox!` and `\count@` must be set to zero. It is called by `\@@az@string`.

```
48.453 \def\hun@alph@lehgth#1#2\hbox#3!{%
48.454   \ifcat\noexpand#1A%
48.455     \advance\count@ by \@ne
48.456   \hun@alph@lehgth#2\hbox\hbox!\fi}
```

`\@@az@string` This macro is called by `\@@az` if the argument begins with a letter. The task of `\@@az@string` is to determine if the argument starts with a vowel and in that case `\let\@tempa\@tempb`. After checking if the first letter is A, E, I, O, or U, `\hun@alph@lehgth` is called to determine the length of the argument. If it gives 1 (that is the argument is a single-letter one or the second character is not letter) then the letters L, M, N, R, S, X, and Y are also considered as a vowel since their Hungarian pronounced name starts with a vowel.

```
48.457 \def\@@az@string#1#2{%
48.458   \ifx#1A%
48.459     \let\@tempa\@tempb
48.460   \else\ifx#1E%
48.461     \let\@tempa\@tempb
48.462   \else\ifx#1I%
48.463     \let\@tempa\@tempb
48.464   \else\ifx#1O%
48.465     \let\@tempa\@tempb
48.466   \else\ifx#1U%
48.467     \let\@tempa\@tempb
48.468   \fi\fi\fi\fi\fi
48.469   \ifx\@tempa\@tempb
48.470   \else
48.471     \count@\z@
48.472     \hun@alph@lehgth#1#2\hbox\hbox!%
48.473     \ifnum\count@=\@ne
48.474       \ifx#1F%
48.475         \let\@tempa\@tempb
48.476       \else\ifx#1L%
48.477         \let\@tempa\@tempb
48.478       \else\ifx#1M%
48.479         \let\@tempa\@tempb
48.480       \else\ifx#1N%
48.481         \let\@tempa\@tempb
48.482       \else\ifx#1R%
48.483         \let\@tempa\@tempb
48.484       \else\ifx#1S%
48.485         \let\@tempa\@tempb
48.486       \else\ifx#1X%
48.487         \let\@tempa\@tempb
48.488       \else\ifx#1Y%
48.489         \let\@tempa\@tempb
48.490       \fi\fi\fi\fi\fi\fi\fi\fi\fi
48.491     \fi
48.492   \fi}
```

`\@@az` This macro is the core of definite article handling. It determines if the argument needs ‘az’ or ‘a’ definite article by setting `\@tempa` to ‘z’ or `\@empty`. It sets `\@tempa` to ‘z’ if

- the first character of the argument is 5; or
- the first character of the argument is 1 and the *length of the number* (mod 3) = 1 (one–egy, thousand–ezer, million–egymillió,...); or
- the first character of the argument is a, A, e, E, i, I, o, O, u, or U; or

- the first character of the argument is l, L, m, M, n, N, r, R, s, S, x, X, y, or Y and the length of the argument is 1 or the second character is a non-letter.

At the end it calls `\@tempa`, that is, it either typesets a ‘z’ or nothing.

```

48.493 \def\@az#1#2\hbox#3!{%
48.494   \let\@tempa\@empty
48.495   \def\@tempb{z}%
48.496   \uppercase{%
48.497     \ifx5#1%
48.498       \let\@tempa\@tempb
48.499     \else\ifx1#1%
48.500       \count@\@ne
48.501       \hun@number@length#2\hbox\hbox!%
48.502       \loop
48.503       \ifnum\count@>\thr@@
48.504         \advance\count@-\thr@@
48.505       \repeat
48.506       \ifnum\count@=\@ne
48.507         \let\@tempa\@tempb
48.508       \fi
48.509     \else
48.510       \@az@string{#1}{#2}%
48.511     \fi\fi
48.512   }%
48.513   \@tempa}

```

`\refstepcounter` `\refstepcounter` must be redefined in order to keep `\@currentlabel` unexpanded. This is necessary to enable the `\label` command to write a `\hunnewlabel` command to the aux file with the Roman numerals substituted by their Arabic representations. Of course, the original definition of `\refstepcounter` is saved and restored if the Hungarian language is switched off.

```

48.514 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.515   \babel@save\refstepcounter
48.516   \def\refstepcounter#1{\stepcounter{#1}%
48.517     \def\@currentlabel{\csname p@#1\endcsname\csname the#1\endcsname}}%
48.518 }

```

`\label` `\label` is redefined to write another line into the aux file: `\hunnewlabel{ }{ }` where the Roman numerals are replaced their Arabic representations. The original definition of `\label` is saved into `\old@label` and it is also called by `\label`. On leaving the Hungarian typesetting mode `\label`’s original is restored since it is added to `\noextrasmagyar`.

```

48.519 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.520   \let\old@label\label
48.521   \def\label#1{\@bsphack
48.522     \old@label{#1}%
48.523     \begingroup
48.524       \let\romannumeral\number
48.525       \def\@roman##1{\number ##1}%
48.526       \def\@Roman##1{\number ##1}%
48.527       {\toks0={\noexpand\noexpand\noexpand\number}%
48.528        \def\number##1{\the\toks0 ##1}\xdef\tempb{\thepage}}%
48.529       \edef\@tempa##1{\noexpand\protected@write\@auxout{}%
48.530        {\noexpand\string\noexpand\hunnewlabel
48.531         {##1}{\@currentlabel}{\tempb}}}%
48.532       \@tempa{#1}%
48.533     \endgroup
48.534   \@esphack}%
48.535 }
48.536 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
48.537   \let\label\old@label
48.538 }

```

`\hunnewlabel` Finally, `\hunnewlabel` is defined. It checks if the label's expansion (#2) differs from that one given in the `\newlabel` command. If yes (that is, the label contains some Roman numerals), it defines the macro `\hun@r@label`, otherwise it does nothing.

```

48.539 \def\hunnewlabel#1#2{%
48.540   \def\@tempa{#2}%
48.541   \expandafter\ifx\csname r@#1\endcsname\@tempa
48.542     \relax% \message{No need for def: #1}%
48.543   \else
48.544     \global\expandafter\let\csname hun@r@#1\endcsname\@tempa%
48.545   \fi
48.546 }
```

For Hungarian the ‘ character is made active.

```

48.547 \AtBeginDocument{%
48.548   \if@files\immediate\write\@auxout{\catcode096=12}\fi}
48.549 \initiate@active@char{'}
48.550 \expandafter\addto\csname extras\CurrentOption\endcsname{%
48.551   \languageshorthands{magyar}%
48.552   \bbl@activate{'}}
48.553 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
48.554   \bbl@deactivate{'}}
```

The character sequence ‘‘ is declared as a shorthand in order to produce the opening quotation sign appropriate for Hungarian.

```

48.555 \declare@shorthand{magyar}{‘}{\glqq}
```

In Hungarian there are some long double consonants which must be hyphenated specially. For all these long double consonants (except dzzs, that is extremely very-very rare) a shortcut is defined.

```

48.556 \declare@shorthand{magyar}{‘c}{\textormath{\bbl@disc{c}{cs}}{c}}
48.557 \declare@shorthand{magyar}{‘C}{\textormath{\bbl@disc{C}{CS}}{C}}
48.558 \declare@shorthand{magyar}{‘d}{\textormath{\bbl@disc{d}{dz}}{d}}
48.559 \declare@shorthand{magyar}{‘D}{\textormath{\bbl@disc{D}{DZ}}{D}}
48.560 \declare@shorthand{magyar}{‘g}{\textormath{\bbl@disc{g}{gy}}{g}}
48.561 \declare@shorthand{magyar}{‘G}{\textormath{\bbl@disc{G}{GY}}{G}}
48.562 \declare@shorthand{magyar}{‘l}{\textormath{\bbl@disc{l}{ly}}{l}}
48.563 \declare@shorthand{magyar}{‘L}{\textormath{\bbl@disc{L}{LY}}{L}}
48.564 \declare@shorthand{magyar}{‘n}{\textormath{\bbl@disc{n}{ny}}{n}}
48.565 \declare@shorthand{magyar}{‘N}{\textormath{\bbl@disc{N}{NY}}{N}}
48.566 \declare@shorthand{magyar}{‘s}{\textormath{\bbl@disc{s}{sz}}{s}}
48.567 \declare@shorthand{magyar}{‘S}{\textormath{\bbl@disc{S}{SZ}}{S}}
48.568 \declare@shorthand{magyar}{‘t}{\textormath{\bbl@disc{t}{ty}}{t}}
48.569 \declare@shorthand{magyar}{‘T}{\textormath{\bbl@disc{T}{TY}}{T}}
48.570 \declare@shorthand{magyar}{‘z}{\textormath{\bbl@disc{z}{zs}}{z}}
48.571 \declare@shorthand{magyar}{‘Z}{\textormath{\bbl@disc{Z}{ZS}}{Z}}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

48.572 \ldf@finish\CurrentOption
48.573 \code>
```

49 The Estonian language

The file `estonian.dtx`⁵⁷ defines the language definition macro's for the Estonian language.

This file was written as part of the TWGML project, and borrows heavily from the `babel` German and Spanish language files `germanb.ldf` and `spanish.ldf`.

Estonian has the same umlauts as German (ä, ö, ü), but in addition to this, we have also õ, and two recent characters š and ž, so we need at least two active characters. We shall use " and ~ to type Estonian accents on ASCII keyboards (in the 7-bit character world). Their use is given in table 23. These active accent

~o	\~o, (and uppercase);
"a	\"a, (and uppercase);
"o	\"o, (and uppercase);
"u	\"u, (and uppercase);
~s	\v s, (and uppercase);
~z	\v z, (and uppercase);
"	disable ligature at this position;
"-	like \-, but allowing hyphenation in the rest of the word;
"‘	for Estonian low left double quotes (same as German);
"’	for Estonian right double quotes;
"<	for French left double quotes (also rather popular)
">	for French right double quotes.

Table 23: The extra definitions made by `estonian.ldf`

characters behave according to their original definitions if not followed by one of the characters indicated in that table; the original quote character can be typed using the macro `\dq`.

We support also the T1 output encoding (and Cork-encoded text input). You can choose the T1 encoding by the command `\usepackage[T1]{fontenc}`. This package must be loaded before `babel`. As the standard Estonian hyphenation file `eehyph.tex` is in the Cork encoding, choosing this encoding will give you better hyphenation.

As mentioned in the Spanish style file, it may happen that some packages fail (usually in a `\message`). In this case you should change the order of the `\usepackage` declarations or the order of the style options in `\documentclass`.

49.1 Implementation

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
49.1 \langle*code\rangle
49.2 \LdfInit{estonian}\captionsestonian

49.3 \ifx\l@estonian\@undefined
49.4 \@nopatterns{Estonian}
49.5 \adddialect\l@estonian0
49.6 \fi
```

Now come the commands to switch to (and from) Estonian.

⁵⁷The file described in this section has version number v1.0k and was last revised on 2009/03/08. The original author is Enn Saar, (saar@aai.ee).

`\captionsestonian` The macro `\captionsestonian` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```

49.7 \addto\captionsestonian{%
49.8   \def\prefacename{Sissejuhatus}%
49.9   \def\refname{Viited}%
49.10  \def\bibname{Kirjandus}%
49.11  \def\appendixname{Lisa}%
49.12  \def\contentsname{Sisukord}%
49.13  \def\listfigurename{Joonised}%
49.14  \def\listtablename{Tabelid}%
49.15  \def\indexname{Indeks}%
49.16  \def\figurename{Joonis}%
49.17  \def\tablename{Tabel}%
49.18  \def\partname{Osa}%
49.19  \def\enclname{Lisa(d)}%
49.20  \def\ccname{Koopia(d)}%
49.21  \def\headtoname{}%
49.22  \def\pagename{Lk.}%
49.23  \def\seename{vt.}%
49.24  \def\alsoname{vt. ka}%
49.25  }

```

These captions contain accented characters.

```

49.26 \begingroup \catcode'\active
49.27 \def\x{\endgroup
49.28 \addto\captionsestonian{%
49.29   \def\abstractname{Kokkuvõte}%
49.30   \def\chaptername{Peatükk}%
49.31   \def\proofname{Tõestus}%
49.32   \def\glossaryname{Sõnastik}%
49.33 }}\x

```

`\dateestonian` The macro `\dateestonian` redefines the command `\today` to produce Estonian dates.

```

49.34 \begingroup \catcode'\active
49.35 \def\x{\endgroup
49.36   \def\month@estonian{\ifcase\month\or
49.37     jaanuar\or veebruar\or marts\or aprill\or mai\or juuni\or
49.38     juuli\or august\or september\or oktoober\or november\or
49.39     detsember\fi}}
49.40 \x
49.41 \def\dateestonian{%
49.42   \def\today{\number\day.\space\month@estonian
49.43     \space\number\year.\space a.}}

```

Some useful macros, copied from the spanish package (and renamed `es@...` to `et@...`).

```

49.44 \def\et@sdef#1{\babel@save#1\def#1}
49.45
49.46 \@ifundefined{documentclass}
49.47 {\let\ifet@latex\iffalse}
49.48 {\let\ifet@latex\iftrue}

```

`\extrasestonian` The macro `\extrasestonian` will perform all the extra definitions needed for Estonian. The macro `\noextrasestonian` is used to cancel the actions of `\extrasestonian`. For Estonian, " is made active and has to be treated as 'special' (~ is active already).

```

49.49 \initiate@active@char{"}
49.50 \initiate@active@char{~}
49.51 \addto\extrasestonian{\languageshorthands{estonian}}
49.52 \addto\extrasestonian{\bbl@activate{"}\bbl@activate{~}}

```


Estonian does not use extra spaces after sentences.

```
49.53 \addto\extrasestonian{\bbl@frenchspacing}
49.54 \addto\noextrasestonian{\bbl@nonfrenchspacing}
```

`\estonianhyphenmins` For Estonian, `\lefthyphenmin` and `\righthyphenmin` are both 2.

```
49.55 \providehyphenmins{\CurrentOption}{\tw@}{\tw@}
```

The standard T_EX accents are too high for Estonian typography, we have to lower them (following the babel German style). For umlauts, we can use `\umlautlow` in `babel.ldf`.

```
49.56 \addto\extrasestonian{\umlautlow}
49.57 \addto\noextrasestonian{\umlauthigh}
```

Redefine tilde (as in `spanish.ldf`). In case of L^AT_EX, we redefine the internal macro for the OT1 encoding because in case of T1, the display and hyphenation of words containing `\~o` works better without redefining it (e. g. words containing `\et@gentilde` are not hyphenated unless `\allowhyphens` is used; when copied from Acrobat Reader, pasting an ò generated using `\et@gentilde{o}` gives `\~o` rather than ò; when the times package is used with T1 encoding, `\et@gentilde` places the tilde through the letter o). In plain T_EX there is no encoding infrastructure, so we just redefine `\~`.

```
49.58 \ifet@latex
49.59   \addto\extrasestonian{%
49.60     \expandafter\et@sdef\csname OT1\string\~\endcsname{\et@gentilde}}
49.61 \else
49.62   \addto\extrasestonian{\et@sdef\~{\et@gentilde}}
49.63 \fi
```

`\et@gentilde`

```
49.64 \def\et@gentilde#1{%
49.65   \if#1s\v{#1}\else\if#1S\v{#1}\else%
49.66   \if#1z\v{#1}\else\if#1Z\v{#1}\else%
49.67   \et@newtilde{#1}%
49.68   \fi\fi\fi\fi}
```

`\et@newtilde` For a detailed explanation of the following code see the definition of `\lower@umlaut` in `babel.dtx`.

```
49.69 \def\et@newtilde#1{%
49.70   \leavevmode\bgroup\U@D 1ex%
49.71   {\setbox\z@\hbox{\char126}\dimen@ -.45ex\advance\dimen@ \ht\z@
49.72     \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
49.73   \accent126\fontdimen5\font\U@D #1%
49.74   \egroup}
```

We save the double quote character in `\dq`, and tilde in `\til`.

```
49.75 \begingroup \catcode'\ "12
49.76 \edef\x{\endgroup
49.77   \def\noexpand\dq{"}
49.78   \def\noexpand\til{~}}
49.79 \x
```

If the encoding is T1, we have to tell T_EX about our redefined accents.

```
49.80 \ifx\fontencoding\bbl@t@one
49.81   \DeclareTextComposite{\~}{T1}{s}{178}
49.82   \DeclareTextComposite{\~}{T1}{S}{146}
49.83   \DeclareTextComposite{\~}{T1}{z}{186}
49.84   \DeclareTextComposite{\~}{T1}{Z}{154}
49.85   \DeclareTextComposite{\~}{T1}{'}{17}
49.86   \DeclareTextComposite{\~}{T1}{'}{18}
49.87   \DeclareTextComposite{\~}{T1}{<}{19}
49.88   \DeclareTextComposite{\~}{T1}{>}{20}
```

If the encoding differs from T1, we expand the accents, enabling hyphenation beyond the accent. In this case T_EX will not find all possible breaks, and we have to warn people.

```
49.89 \else
49.90   \wlog{Warning: Hyphenation would work better for the T1 encoding.}
49.91 \fi
```

Now we define the shorthands: umlauts,

```
49.92 \declare@shorthand{estonian}{a}{\textormath{"{a}\allowhyphens}{\ddot a}}
49.93 \declare@shorthand{estonian}{A}{\textormath{"{A}\allowhyphens}{\ddot A}}
49.94 \declare@shorthand{estonian}{o}{\textormath{"{o}\allowhyphens}{\ddot o}}
49.95 \declare@shorthand{estonian}{O}{\textormath{"{O}\allowhyphens}{\ddot O}}
49.96 \declare@shorthand{estonian}{u}{\textormath{"{u}\allowhyphens}{\ddot u}}
49.97 \declare@shorthand{estonian}{U}{\textormath{"{U}\allowhyphens}{\ddot U}}
```

German and French quotes,

```
49.98 \declare@shorthand{estonian}{"'}{%
49.99   \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
49.100 \declare@shorthand{estonian}{"'}{%
49.101   \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
49.102 \declare@shorthand{estonian}{"<"}{%
49.103   \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
49.104 \declare@shorthand{estonian}{">"}{%
49.105   \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

tildes and carons

```
49.106 \declare@shorthand{estonian}{~o}{\textormath{~{o}\allowhyphens}{\tilde o}}
49.107 \declare@shorthand{estonian}{~O}{\textormath{~{O}\allowhyphens}{\tilde O}}
49.108 \declare@shorthand{estonian}{~s}{\textormath{\v{s}\allowhyphens}{\check s}}
49.109 \declare@shorthand{estonian}{~S}{\textormath{\v{S}\allowhyphens}{\check S}}
49.110 \declare@shorthand{estonian}{~z}{\textormath{\v{z}\allowhyphens}{\check z}}
49.111 \declare@shorthand{estonian}{~Z}{\textormath{\v{Z}\allowhyphens}{\check Z}}
```

and some additional commands:

```
49.112 \declare@shorthand{estonian}{"-}{\nobreak\-\bbl@allowhyphens}
49.113 \declare@shorthand{estonian}{"|}{%
49.114   \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
49.115     \allowhyphens}{}}
49.116 \declare@shorthand{estonian}{""}{\dq}
49.117 \declare@shorthand{estonian}{~}{\til}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
49.118 \ldf@finish{estonian}
49.119 \code}
```

50 The Albanian language

The file `albanian.dtx`⁵⁸ defines all the language definition macros for the Albanian language.

Albanian is written in a latin script, but it has 36 letters, 9 which are diletters (dh, gj, ll, nj, rr, sh, th, xh, zh), and two extra special characters.

For this language the character " is made active. In table 24 an overview is given of its purpose.

"c	\c, also implemented for the uppercase
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
"	disable ligature at this position
" "	like "-", but producing no hyphen sign (for compound words with hyphen, e.g. x-"y).
"‘	for Albanian left double quotes (looks like „).
"’	for Albanian right double quotes.
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).

Table 24: The extra definitions made by `albanian.ldf`

Apart from defining shorthands we need to make sure that the first paragraph of each section is intended. Furthermore the following new math operators are defined (`\tg`, `\ctg`, `\arctg`, `\arcctg`, `\sh`, `\ch`, `\th`, `\cth`, `\arsh`, `\arch`, `\arth`, `\arcth`, `\Prob`, `\Expect`, `\Variance`).

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
50.1 (*code)
50.2 \LdfInit{albanian}\captionسالbanian
```

When this file is read as an option, i.e. by the `\usepackage` command, `albanian` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@albanian` to see whether we have to do something here.

```
50.3 \ifx\l@albanian\@undefined
50.4     \@nopatterns{Albanian}
50.5     \adddialect\l@albanian0\fi
```

The next step consists of defining commands to switch to (and from) the Albanian language.

`\captionسالbanian` The macro `\captionسالbanian` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
50.6 \addto\captionسالbanian{%
50.7     \def\prefacename{Parathenia}%
50.8     \def\refname{Referencat}%
50.9     \def\abstractname{P\ "ermbledhja}%
50.10    \def\bibname{Bibliografia}%
50.11    \def\chaptername{K kapitulli}%
50.12    \def\appendixname{Shtesa}%
50.13    \def\contentsname{P\ "ermbajta}%
50.14    \def\listfigurename{Figurat}%
50.15    \def\listtablename{Tabelat}%
50.16    \def\indexname{Indeksi}%
50.17    \def\figurename{Figura}%
50.18    \def\tablename{Tabela}%
50.19    \def\partname{Pjesa}%
```

⁵⁸The file described in this section has version number v1.0c and was last revised on 2007/10/20

```

50.20 \def\enclname{Lidhja}%
50.21 \def\ccname{Kopja}%
50.22 \def\headtoname{P\ "er}%
50.23 \def\pagename{Faqe}%
50.24 \def\seename{shiko}%
50.25 \def\alsoname{shiko dhe}%
50.26 \def\proofname{V\ "ertetim}%
50.27 \def\glossaryname{P\ "erhasja e Fjal\ "eve}%
50.28 }%

```

`\datealbanian` The macro `\datealbanian` redefines the command `\today` to produce Albanian dates.

```

50.29 \def\datealbanian{%
50.30   \def\today{\number\day~\ifcase\month\or
50.31     Janar\or Shkurt\or Mars\or Prill\or Maj\or
50.32     Qershor\or Korrik\or Gusht\or Shtator\or Tetor\or N\ "entor\or
50.33     Dhjetor\fi \space \number\year}}

```

`\extrasalbanian` The macro `\extrasalbanian` will perform all the extra definitions needed for the Albanian language. The macro `\noextrasalbanian` is used to cancel the actions of `\extrasalbanian`.

For Albanian the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the albanian group of shorthands should be used.

```

50.34 \initiate@active@char{"}
50.35 \addto\extrasalbanian{\languageshorthands{albanian}}
50.36 \addto\extrasalbanian{\bbl@activate{}}

```

Don't forget to turn the shorthands off again.

```

50.37 \addto\noextrasalbanian{\bbl@deactivate{}}

```

First we define shorthands to facilitate the occurrence of letters such as ç.

```

50.38 \declare@shorthand{albanian}{\c}{\textormath{\v c}{\check c}}
50.39 \declare@shorthand{albanian}{\e}{\textormath{\v e}{\check e}}
50.40 \declare@shorthand{albanian}{\C}{\textormath{\v C}{\check C}}
50.41 \declare@shorthand{albanian}{\E}{\textormath{\v E}{\check E}}

```

Then we define access to two forms of quotation marks, similar to the german and french quotation marks.

```

50.42 \declare@shorthand{albanian}{\`}{\%
50.43   \textormath{\quotedblbase}}{\mbox{\quotedblbase}}
50.44 \declare@shorthand{albanian}{\'}{\%
50.45   \textormath{\textquotedblleft}}{\mbox{\textquotedblleft}}
50.46 \declare@shorthand{albanian}{\<}{\%
50.47   \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
50.48 \declare@shorthand{albanian}{\>}{\%
50.49   \textormath{\guillemotright}}{\mbox{\guillemotright}}

```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```

50.50 \declare@shorthand{albanian}{\~}{\nobreak-\bbl@allowhyphens}
50.51 \declare@shorthand{albanian}{\_}{\hskip\z@skip}

```

And we want to have a shorthand for disabling a ligature.

```

50.52 \declare@shorthand{albanian}{\|}{\%
50.53   \textormath{\discretionary{-}{-}{\kern.03em}}{}}

```

`\bbl@frenchindent` In albanian the first paragraph of each section should be indented. Add this code only in L^AT_EX.

```

50.54 \ifx\fmtname plain \else
50.55   \let\@aifORI\@afterindentfalse
50.56   \def\bbl@frenchindent{\let\@afterindentfalse\@afterindenttrue

```

```

50.57 \afterindenttrue}
50.58 \def\bbl@nonfrenchindent{\let\@afterindentfalse\@aifORI
50.59 \afterindentfalse}
50.60 \addto\extrasalbanian{\bbl@frenchindent}
50.61 \addto\noextrasalbanian{\bbl@nonfrenchindent}
50.62 \fi

```

`\mathalbanian` Some math functions in Albanian math books have other names: e.g. `\sinh` in Albanian is written as `\sh` etc. So we define a number of new math operators.

```

50.63 \def\sh{\mathop{\operator@font sh}\nolimits} % same as \sinh
50.64 \def\ch{\mathop{\operator@font ch}\nolimits} % same as \cosh
50.65 \def\th{\mathop{\operator@font th}\nolimits} % same as \tanh
50.66 \def\cth{\mathop{\operator@font cth}\nolimits} % same as \coth
50.67 \def\arsh{\mathop{\operator@font arsh}\nolimits}
50.68 \def\arch{\mathop{\operator@font arch}\nolimits}
50.69 \def\arth{\mathop{\operator@font arth}\nolimits}
50.70 \def\arcth{\mathop{\operator@font arcth}\nolimits}
50.71 \def\tg{\mathop{\operator@font tg}\nolimits} % same as \tan
50.72 \def\ctg{\mathop{\operator@font ctg}\nolimits} % same as \cot
50.73 \def\arctg{\mathop{\operator@font arctg}\nolimits} % same as \arctan
50.74 \def\arcctg{\mathop{\operator@font arcctg}\nolimits}
50.75 \def\Prob{\mathop{\operator@font P}\nolimits}
50.76 \def\Expect{\mathop{\operator@font E}\nolimits}
50.77 \def\Variance{\mathop{\operator@font D}\nolimits}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

50.78 \ldf@finish{albanian}
50.79 \end{code}

```

51 The Croatian language

The file `croatian.dtx`⁵⁹ defines all the language definition macros for the Croatian language.

For this language currently no special definitions are needed or available.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
51.1 (*code)
51.2 \LdfInit{croatian}\captionscroatian
```

When this file is read as an option, i.e. by the `\usepackage` command, `croatian` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@croatian` to see whether we have to do something here.

```
51.3 \ifx\l@croatian\@undefined
51.4     \@nopatterns{Croatian}
51.5     \adddialect\l@croatian0\fi
```

The next step consists of defining commands to switch to (and from) the Croatian language.

`\captionscroatian` The macro `\captionscroatian` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
51.6 \addto\captionscroatian{%
51.7   \def\prefacename{Predgovor}%
51.8   \def\refname{Literatura}%
51.9   \def\abstractname{Sa\v{z}etak}%
51.10  \def\bibname{Bibliografija}%
51.11  \def\chaptername{Poglavlje}%
51.12  \def\appendixname{Dodatak}%
51.13  \def\contentsname{Sadr\v{z}aj}%
51.14  \def\listfigurename{Popis slika}%
51.15  \def\listtablename{Popis tablica}%
51.16  \def\indexname{Indeks}%
51.17  \def\figurename{Slika}%
51.18  \def\tablename{Tablica}%
51.19  \def\partname{Dio}%
51.20  \def\enclname{Prilozi}%
51.21  \def\ccname{Kopije}%
51.22  \def\headtoname{Prima}%
51.23  \def\pagename{Stranica}%
51.24  \def\seename{Vidjeti}%
51.25  \def\alsoname{Vidjeti i}%
51.26  \def\proofname{Dokaz}%
51.27  \def\glossaryname{Kazalo}%
51.28  }%
```

`\datecroatian` The macro `\datecroatian` redefines the command `\today` to produce Croatian dates.

```
51.29 \def\datecroatian{%
51.30   \def\today{\number\day.\~\ifcase\month\or
51.31     sije\v{c}nja\or velja\v{c}e\or o\v{z}ujka\or travnja\or svibnja\or
51.32     lipnja\or srpnja\or kolovoza\or rujna\or listopada\or studenog\or
51.33     prosinca\fi \space \number\year.}}
```

`\extrascroatian` The macro `\extrascroatian` will perform all the extra definitions needed for the Croatian language. The macro `\noextrascroatian` is used to cancel the actions of `\extrascroatian`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

⁵⁹The file described in this section has version number v1.3l and was last revised on 2005/03/29. A contribution was made by Alan Pać (paica@cernvm.cern.ch).

51.34 `\addto\extrascroatian{}`
51.35 `\addto\noextrascroatian{}`

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

51.36 `\ldf@finish{croatian}`
51.37 `\code`

52 The Czech Language

The file `czech.dtx`⁶⁰ defines all the language definition macros for the Czech language. It is meant as a replacement of $\mathcal{CS}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}$, the most-widely used standard for typesetting Czech documents in $\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}$.

52.1 Usage

For this language `\frenchspacing` is set.

Additionally, two macros are defined `\q` and `\w` for easy access to two accents are defined.

The command `\q` is used with the letters (`t`, `d`, `l`, and `L`) and adds a `'` to them to simulate a ‘hook’ that should be there. The result looks like `ť`. The command `\w` is used to put the ring-accent which appears in `ångström` over the letters `u` and `U`.

52.2 Compatibility

Great care has been taken to ensure backward compatibility with $\mathcal{CS}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}$. In particular, documents which load this file with `\usepackage{czech}` should produce identical output with no modifications to the source. Additionally, all the $\mathcal{CS}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}$ options are recognized:

l2, T1, OT1

These options set the default font encoding. Please note that their use is deprecated. You should use the `fontenc` package to select font encoding.

split, nosplit

These options control whether hyphenated words are automatically split according to Czech typesetting rules. With the `split` option “`je-li`” is hyphenated as “`je-/li`”. The `nosplit` option disables this behavior.

The use of this option is strongly discouraged, as it breaks too many common things—hyphens cannot be used in labels, negative arguments to $\mathrm{T}\mathrm{E}\mathrm{X}$ primitives will not work in horizontal mode (use `\minus` as a workaround), and there are a few other peculiarities with using this mode.

nocaptions

This option was used in $\mathcal{CS}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}$ to set up Czech/Slovak typesetting rules, but leave the original captions and dates. The recommended way to achieve this is to use English as the main language of the document and use the environment `otherlanguage*` for Czech text.

olduv There are two version of `\uv`. The older one allows the use of `\verb` inside the quotes but breaks any respective kerning with the quotes (like that in \mathcal{CS} fonts). The newer one honors the kerning in the font but does not allow `\verb` inside the quotes.

The new version is used by default in $\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\,2_{\epsilon}$ and the old version is used with plain $\mathrm{T}\mathrm{E}\mathrm{X}$. You may use `olduv` to override the default in $\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\,2_{\epsilon}$.

cstex This option was used to include the commands `\csprimeson` and `\csprimesoff`. Since these commands are always included now, it has been removed and the empty definition lasts for compatibility.

⁶⁰The file described in this section has version number v3.1a and was last revised on 2008/07/06. It was rewritten by Petr Tesařík (`babel@tesarici.cz`).

52.3 Implementation

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
52.1 {*code}
52.2 \LdfInit\CurrentOption{date\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `czech` might be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@czech` to see whether we have to do something here.

```
52.3 \ifx\l@czech\@undefined
52.4 \@nopatterns{Czech}
52.5 \adddialect\l@czech0\fi
```

We need to define these macros early in the process.

```
52.6 \def\cs@iltw@{IL2}
52.7 \newif\ifcs@splithyphens
52.8 \cs@splithyphensfalse
```

If Babel is not loaded, we provide compatibility with \LaTeX . However, if macro `\@ifpackageloaded` is not defined, we assume to be loaded from plain and provide compatibility with `csplain`. Of course, this does not work well with \LaTeX 2.09, but I doubt anyone will ever want to use this file with \LaTeX 2.09.

```
52.9 \ifx\@ifpackageloaded\@undefined
52.10 \let\cs@compat@plain\relax
52.11 \message{csplain compatibility mode}
52.12 \else
52.13 \@ifpackageloaded{babel}{\%
52.14 \let\cs@compat@latex\relax
52.15 \message{cslatex compatibility mode}}
52.16 \fi
52.17 \ifx\cs@compat@latex\relax
52.18 \ProvidesPackage{czech}[2008/07/06 v3.1a C\TeX Czech style]
```

Declare \LaTeX options (see also the descriptions on page 256).

```
52.19 \DeclareOption{IL2}{\def\encodingdefault{IL2}}
52.20 \DeclareOption{T1}{\def\encodingdefault{T1}}
52.21 \DeclareOption{OT1}{\def\encodingdefault{OT1}}
52.22 \DeclareOption{nosplit}{\cs@splithyphensfalse}
52.23 \DeclareOption{split}{\cs@splithyphenstrue}
52.24 \DeclareOption{nocaptions}{\let\cs@nocaptions=\relax}
52.25 \DeclareOption{olduv}{\let\cs@olduv=\relax}
52.26 \DeclareOption{cstex}{\relax}
```

Make IL2 encoding the default. This can be overridden with the other font encoding options.

```
52.27 \ExecuteOptions{\cs@iltw@}
```

Now, process the user-supplied options.

```
52.28 \ProcessOptions
```

Standard \LaTeX 2_ε does not include the IL2 encoding in the format. The encoding can be loaded later using the `fontenc` package, but \LaTeX included IL2 by default. This means existing documents for \LaTeX do not load that package, so load the encoding ourselves in compatibility mode.

```
52.29 \ifx\encodingdefault\cs@iltw@
52.30 \input il2enc.def
52.31 \fi
```

Restore the definition of `\CurrentOption`, clobbered by processing the options.

```
52.32 \def\CurrentOption{czech}
52.33 \fi
```

The next step consists of defining commands to switch to (and from) the Czech language.

`\captionsczech` The macro `\captionsczech` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```

52.34 \@namedef{captions\CurrentOption}{%
52.35   \def\prefacename{P\v{r}edmluva}%
52.36   \def\refname{Reference}%
52.37   \def\abstractname{Abstrakt}%
52.38   \def\bibname{Literatura}%
52.39   \def\chaptername{Kapitola}%
52.40   \def\appendixname{P\v{r}\'\{i\}loha}%
52.41   \def\contentsname{Obsah}%
52.42   \def\listfigurename{Seznam obr\'azk\v{r}{u}}%
52.43   \def\listtablename{Seznam tabulek}%
52.44   \def\indexname{Rejst\v{r}\'\{i\}k}%
52.45   \def\figurename{Obr\'azek}%
52.46   \def\tablename{Tabulka}%
52.47   \def\partname{\v{C}\'ast}%
52.48   \def\enclname{P\v{r}\'\{i\}loha}%
52.49   \def\ccname{Na v\v{e}dom\'\{i\}}%
52.50   \def\headtoname{Komu}%
52.51   \def\pagename{Strana}%
52.52   \def\seename{viz}%
52.53   \def\alsoname{viz tak\'e}%
52.54   \def\proofname{D\v{r}{u}kaz}%
52.55   \def\glossaryname{Slovn\'\{i\}k}%
52.56   }%

```

`\dateczech` The macro `\dateczech` redefines the command `\today` to produce Czech dates.

C_SL_AT_EX allows line break between the day and the month. However, this behavior has been agreed upon to be a bad thing by the c_ST_EX mailing list in December 2005 and has not been adopted.

```

52.57 \@namedef{date\CurrentOption}{%
52.58   \def\today{\number\day.\~ifcase\month\or ledna\or \'unora\or
52.59     b\v{r}{ezna}\or dubna\or kv\v{e}tna\or \v{c}{ervna}\or \v{c}{ervence}\or
52.60     srpna\or z\'a\v{r}\'\{i\}\or \v{r}\'\{i\}jna\or listopadu\or
52.61     prosince\fi \space\number\year}}

```

`\extrasczech` The macro `\extrasczech` will perform all the extra definitions needed for the Czech language. The macro `\noextrasczech` is used to cancel the actions of `\extrasczech`. This means saving the meaning of two one-letter control sequences before defining them.

For Czech texts `\frenchspacing` should be in effect. Language group for shorthands is also set here.

```

52.62 \expandafter\addto\csname extras\CurrentOption\endcsname{%
52.63   \bbl@frenchspacing
52.64   \languageshorthands{czech}}
52.65 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
52.66   \bbl@nonfrenchspacing}

52.67 \expandafter\addto\csname extras\CurrentOption\endcsname{%
52.68   \babel@save\q\let\q\v
52.69   \babel@save\w\let\w\r}

```

`\sq` We save the original single and double quote characters in `\sq` and `\dq` to make `\dq` them available later.

```

52.70 \begingroup\catcode\'"=12\catcode\''=12
52.71 \def\x{\endgroup
52.72   \def\sq{'}
52.73   \def\dq{"}}
52.74 \x

```

This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
52.75 \providehyphenmins{\CurrentOption}{\tw@\thr@@}
```

`\v` L^AT_EX's normal `\v` accent places a caron over the letter that follows it (ö). This is not what we want for the letters d, t, l and L; for those the accent should change shape. This is achieved by the following.

```
52.76 \AtBeginDocument{%
52.77   \DeclareTextCompositeCommand{\v}{OT1}{t}{t}{%
52.78     t\kern-.23em\raise.24ex\hbox{'t}}
52.79   \DeclareTextCompositeCommand{\v}{OT1}{d}{d}{%
52.80     d\kern-.13em\raise.24ex\hbox{'d}}
52.81   \DeclareTextCompositeCommand{\v}{OT1}{l}{l}{\lcaron{}}
52.82   \DeclareTextCompositeCommand{\v}{OT1}{L}{L}{\Lcaron{}}}
```

`\lcaron` For the letters l and L we want to distinguish between normal fonts and monospaced fonts.

```
\Lcaron
52.83 \def\lcaron{%
52.84   \setbox0\hbox{M}\setbox\tw@\hbox{i}%
52.85   \ifdim\wd0>\wd\tw@\relax
52.86     l\kern-.13em\raise.24ex\hbox{'l'}\kern-.11em%
52.87   \else
52.88     l\raise.45ex\hbox to\z@{\kern-.35em '\hss}%
52.89   \fi}
52.90 \def\Lcaron{%
52.91   \setbox0\hbox{M}\setbox\tw@\hbox{i}%
52.92   \ifdim\wd0>\wd\tw@\relax
52.93     L\raise.24ex\hbox to\z@{\kern-.28em'\hss}%
52.94   \else
52.95     L\raise.45ex\hbox to\z@{\kern-.40em '\hss}%
52.96   \fi}
```

Initialize active quotes. C_SL_AT_EX provides a way of converting English-style quotes into Czech-style ones. Both single and double quotes are affected, i.e. ‘`text`’ is converted to something like `,,text‘` and ‘`text`’ is converted to `,text‘`. This conversion can be switched on and off with `\csprimeson` and `\csprimesoff`.⁶¹

These quotes present various troubles, e.g. the kerning is broken, apostrophes are converted to closing single quote, some primitives are broken (most notably the `\catcode‘\⟨char⟩` syntax will not work any more), and writing them to `.aux` files cannot be handled correctly. For these reasons, these commands are only available in C_SL_AT_EX compatibility mode.

```
52.97 \ifx\cs@compat@latex\relax
52.98   \let\cs@ltxprim@s\prim@s
52.99   \def\csprimeson{%
52.100     \catcode‘‘\active \catcode‘\’\active \let\prim@s\bbl@prim@s}
52.101   \def\csprimesoff{%
52.102     \catcode‘‘12 \catcode‘\’12 \let\prim@s\cs@ltxprim@s}
52.103   \begingroup\catcode‘‘\active
52.104   \def\x{\endgroup
52.105     \def‘{\futurelet\cs@next\cs@openquote}
52.106     \def\cs@openquote{%
52.107       \ifx\cs@next \expandafter\cs@opendq
52.108       \else \expandafter\clq
52.109       \fi}%
52.110   }\x
52.111   \begingroup\catcode‘\’\active
52.112   \def\x{\endgroup}
```

⁶¹By the way, the names of these macros are misleading, because the handling of primes in math mode is rather marginal, the most important thing being the handling of quotes...

```

52.113 \def'\textormath{\futurelet\cs@next\cs@closequote}
52.114         {\bgroup\prim@s}}
52.115 \def\cs@closequote{%
52.116     \ifx'\cs@next \expandafter\cs@closedq
52.117     \else \expandafter\crq
52.118     \fi}%
52.119 } \x
52.120 \def\cs@opendq{\clqq\let\cs@next= }
52.121 \def\cs@closedq{\crqq\let\cs@next= }

```

The way I recommend for typesetting quotes in Czech documents is to use shorthands similar to those used in German.

```

52.122 \else
52.123 \initiate@active@char{"}
52.124 \expandafter\addto\csname extras\CurrentOption\endcsname{%
52.125     \bbl@activate{"}}
52.126 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
52.127     \bbl@deactivate{"}}
52.128 \declare@shorthand{czech}{"'}{\clqq}
52.129 \declare@shorthand{czech}{"'}{\crqq}
52.130 \declare@shorthand{czech}{"<}{\flqq}
52.131 \declare@shorthand{czech}{">}{\frqq}
52.132 \declare@shorthand{czech}{"=}{\cs@splithyphen}
52.133 \fi

```

`\clqq` This is the CS opening quote, which is similar to the German quote (`\glqq`) but the kerning is different.

For the OT1 encoding, the quote is constructed from the right double quote (i.e. the “Opening quotes” character) by moving it down to the baseline and shifting it to the right, or to the left if italic correction is positive.

For T1, the “German Opening quotes” is used. It is moved to the right and the total width is enlarged. This is done in an attempt to minimize the difference between the OT1 and T1 versions.

```

52.134 \ProvideTextCommand{\clqq}{OT1}{%
52.135     \set@low@box{\textquotedblright}%
52.136     \setbox\@ne=\hbox{1/}\dimen\@ne=\wd\@ne
52.137     \setbox\@ne=\hbox{1}\advance\dimen\@ne-\wd\@ne
52.138     \leavevmode
52.139     \ifdim\dimen\@ne>z@\kern-.1em\box\z@\kern.1em
52.140     \else\kern.1em\box\z@\kern-.1em\fi\allowhyphens}
52.141 \ProvideTextCommand{\clqq}{T1}
52.142     {\kern.1em\quotedblbase\kern-.0158em\relax}
52.143 \ProvideTextCommandDefault{\clqq}{\UseTextSymbol{OT1}\clqq}

```

`\crqq` For OT1, the CS closing quote is basically the same as `\grqq`, only the `\textormath` macro is not used, because as far as I know, `\grqq` does not work in math mode anyway.

For T1, the character is slightly wider and shifted to the right to match its OT1 counterpart.

```

52.144 \ProvideTextCommand{\crqq}{OT1}
52.145     {\save@sf@q{\nobreak\kern-.07em\textquotedblleft\kern.07em}}
52.146 \ProvideTextCommand{\crqq}{T1}
52.147     {\save@sf@q{\nobreak\kern.06em\textquotedblleft\kern.024em}}
52.148 \ProvideTextCommandDefault{\crqq}{\UseTextSymbol{OT1}\crqq}

```

`\clq` Single CS quotes are similar to double quotes (see the discussion above).

```

\crq52.149 \ProvideTextCommand{\clq}{OT1}
52.150     {\set@low@box{\textquoteright}\box\z@\kern.04em\allowhyphens}
52.151 \ProvideTextCommand{\clq}{T1}
52.152     {\quotesinglbase\kern-.0428em\relax}
52.153 \ProvideTextCommandDefault{\clq}{\UseTextSymbol{OT1}\clq}

```

```

52.154 \ProvideTextCommand{\crq}{OT1}
52.155   {\save@sf@q{\nobreak\textquoteleft\kern.17em}}
52.156 \ProvideTextCommand{\crq}{T1}
52.157   {\save@sf@q{\nobreak\textquoteleft\kern.17em}}
52.158 \ProvideTextCommandDefault{\crq}{\UseTextSymbol{OT1}\crq}

```

`\uv` There are two versions of `\uv`. The older one opens a group and uses `\aftergroup` to typeset the closing quotes. This version allows using `\verb` inside the quotes, because the enclosed text is not passed as an argument, but unfortunately it breaks any kerning with the quotes. Although the kerning with the opening quote could be fixed, the kerning with the closing quote cannot.

The newer version is defined as a command with one parameter. It preserves kerning but since the quoted text is passed as an argument, it cannot contain `\verb`.

Decide which version of `\uv` should be used. For sake of compatibility, we use the older version with plain \TeX and the newer version with $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2_{\varepsilon}$.

```

52.159 \ifx\cs@compat@plain\undefined\else\let\cs@olduv=\relax\fi
52.160 \ifx\cs@olduv\undefined
52.161   \DeclareRobustCommand\uv[1]{\leavevmode\clqq#1\crqq}
52.162 \else
52.163   \DeclareRobustCommand\uv{\bgroup\aftergroup\closequotes
52.164     \leavevmode\clqq\let\cs@next=}
52.165   \def\closequotes{\unskip\crqq\relax}
52.166 \fi

```

`\cs@wordlen` Declare a counter to hold the length of the word after the hyphen.

```

52.167 \newcount\cs@wordlen

```

`\cs@hyphen` Store the original hyphen in a macro. Ditto for the ligatures.

```

\cs@endash52.168 \begingroup\catcode'\-12
\cs@emdash52.169 \def\x{\endgroup
52.170   \def\cs@hyphen{-}
52.171   \def\cs@endash{--}
52.172   \def\cs@emdash{---}

```

`\cs@boxhyphen` Provide a non-breakable hyphen to be used when a compound word is too short to be split, i.e. the second part is shorter than `\righthyphenmin`.

```

52.173 \def\cs@boxhyphen{\hbox{-}}

```

`\cs@splithyphen` The macro `\cs@splithyphen` inserts a split hyphen, while allowing both parts of the compound word to be hyphenated at other places too.

```

52.174 \def\cs@splithyphen{\kern\z@
52.175   \discretionary{-}{\char\hyphenchar\the\font}{-}\nobreak\hskip\z@}
52.176 } \x

```

- To minimize the effects of activating the hyphen character, the active definition expands to the non-active character in all cases where hyphenation cannot occur, i.e. if not typesetting (check `\protect`), not in horizontal mode, or in inner horizontal mode.

```

52.177 \initiate@active@char{-}
52.178 \declare@shorthand{czech}{-}{%
52.179   \ifx\protect\@typeset@protect
52.180     \ifhmode
52.181       \ifinner
52.182         \bbl@afterelse\bbl@afterelse\bbl@afterelse\cs@hyphen
52.183       \else
52.184         \bbl@afterfi\bbl@afterelse\bbl@afterelse\cs@firsthyphen
52.185       \fi
52.186     \else
52.187       \bbl@afterfi\bbl@afterelse\cs@hyphen

```

```

52.188 \fi
52.189 \else
52.190 \bbl@afterfi\cs@hyphen
52.191 \fi}

```

`\cs@firsthyphen` If we encounter a hyphen, check whether it is followed by a second or a third hyphen and if so, insert the corresponding ligature.
`\cs@firsthyph@n`
`\cs@secondhyphen` If we don't find a hyphen, the token found will be placed in `\cs@token` for further analysis, and it will also stay in the input.
`\cs@secondhyph@n`

```

52.192 \begingroup\catcode'\- \active
52.193 \def\x{\endgroup
52.194 \def\cs@firsthyphen{\futurelet\cs@token\cs@firsthyph@n}
52.195 \def\cs@firsthyph@n{%
52.196 \ifx -\cs@token
52.197 \bbl@afterelse\cs@secondhyphen
52.198 \else
52.199 \bbl@afterfi\cs@checkhyphen
52.200 \fi}
52.201 \def\cs@secondhyphen##1{%
52.202 \futurelet\cs@token\cs@secondhyph@n}
52.203 \def\cs@secondhyph@n{%
52.204 \ifx -\cs@token
52.205 \bbl@afterelse\cs@emdash@gobble
52.206 \else
52.207 \bbl@afterfi\cs@endash
52.208 \fi}
52.209 } \x

```

`\cs@checkhyphen` Check that hyphenation is enabled, and if so, start analyzing the rest of the word, i.e. initialize `\cs@word` and `\cs@wordlen` and start processing input with `\cs@scanword`.

```

52.210 \def\cs@checkhyphen{%
52.211 \ifnum\expandafter\hyphenchar\the\font='-
52.212 \def\cs@word{}\cs@wordlen\z@
52.213 \bbl@afterelse\cs@scanword
52.214 \else
52.215 \cs@hyphen
52.216 \fi}

```

`\cs@scanword` Each token is first analyzed with `\cs@scanword`, which expands the token and passes the first token of the result to `\cs@gett@ken`. If the expanded token is not identical to the unexpanded one, presume that it might be expanded further and pass it back to `\cs@scanword` until you get an unexpandable token. Then analyze it in `\cs@examinetoken`.
`\cs@continuescan`
`\cs@gett@ken`
`\cs@gett@ken`

The `\cs@continuescan` macro does the same thing as `\cs@scanword`, but it does not require the first token to be in `\cs@token` already.

```

52.217 \def\cs@scanword{\let\cs@lasttoken=\cs@token\expandafter\cs@gett@ken}
52.218 \def\cs@continuescan{\let\cs@lasttoken\@undefined\expandafter\cs@gett@ken}
52.219 \def\cs@gett@ken{\futurelet\cs@token\cs@gett@ken}
52.220 \def\cs@gett@ken{%
52.221 \ifx\cs@token\cs@lasttoken \def\cs@next{\cs@examinetoken}%
52.222 \else \def\cs@next{\cs@scanword}%
52.223 \fi \cs@next}

```

`\cs@examinetoken` Examine the token in `\cs@token`:

- If it is a letter (catcode 11) or other (catcode 12), add it to `\cs@word` with `\cs@addparam`.
- If it is the `\char` primitive, add it with `\cs@expandchar`.
- If the token starts or ends a group, ignore it with `\cs@ignoretoken`.

- Otherwise analyze the meaning of the token with `\cs@checkchardef` to detect primitives defined with `\chardef`.

```

52.224 \def\cs@examinetoken{%
52.225   \ifcat A\cs@token
52.226     \def\cs@next{\cs@addparam}%
52.227   \else\ifcat 0\cs@token
52.228     \def\cs@next{\cs@addparam}%
52.229   \else\ifx\char\cs@token
52.230     \def\cs@next{\afterassignment\cs@expandchar\let\cs@token= }%
52.231   \else\ifx\bgroup\cs@token
52.232     \def\cs@next{\cs@ignoretoken\bgroup}%
52.233   \else\ifx\egroup\cs@token
52.234     \def\cs@next{\cs@ignoretoken\egroup}%
52.235   \else\ifx\begin\cs@token
52.236     \def\cs@next{\cs@ignoretoken\begin}%
52.237   \else\ifx\end\cs@token
52.238     \def\cs@next{\cs@ignoretoken\end}%
52.239   \else
52.240     \def\cs@next{\expandafter\expandafter\expandafter\cs@checkchardef
52.241       \expandafter\meaning\expandafter\cs@token\string\char\end}%
52.242   \fi\fi\fi\fi\fi\fi\cs@next}

```

`\cs@checkchardef` Check the meaning of a token and if it is a primitive defined with `\chardef`, pass it to `\\@examinechar` as if it were a `\char` sequence. Otherwise, there are no more word characters, so do the final actions in `\cs@nosplit`.

```

52.243 \expandafter\def\expandafter\cs@checkchardef
52.244   \expandafter#\expandafter1\string\char#2\end{%
52.245   \def\cs@token{#1}%
52.246   \ifx\cs@token@empty
52.247     \def\cs@next{\afterassignment\cs@examinechar\let\cs@token= }%
52.248   \else
52.249     \def\cs@next{\cs@nosplit}%
52.250   \fi \cs@next}

```

`\cs@ignoretoken` Add a token to `\cs@word` but do not update the `\cs@wordlen` counter. This is mainly useful for group starting and ending primitives, which need to be preserved, but do not affect the word boundary.

```

52.251 \def\cs@ignoretoken#1{%
52.252   \edef\cs@word{\cs@word#1}%
52.253   \afterassignment\cs@continuescan\let\cs@token= }

```

`cs@addparam` Add a token to `\cs@word` and check its lcode. Note that this macro can only be used for tokens which can be passed as a parameter.

```

52.254 \def\cs@addparam#1{%
52.255   \edef\cs@word{\cs@word#1}%
52.256   \cs@checkcode{\lcode'#1}}

```

`\cs@expandchar` Add a `\char` sequence to `\cs@word` and check its lcode. The charcode is first parsed in `\cs@expandchar` and then the resulting `\chardef`-defined sequence is analyzed in `\cs@examinechar`.

```

52.257 \def\cs@expandchar{\afterassignment\cs@examinechar\chardef\cs@token=}
52.258 \def\cs@examinechar{%
52.259   \edef\cs@word{\cs@word\char\the\cs@token\space}%
52.260   \cs@checkcode{\lcode\cs@token}}

```

`\cs@checkcode` Check the lcode of a character. If it is zero, it does not count to the current word, so finish it with `\cs@nosplit`. Otherwise update the `\cs@wordlen` counter and go on scanning the word with `\cs@continuescan`. When enough characters are gathered in `\cs@word` to allow word break, insert the split hyphen and finish.

```

52.261 \def\cs@checkcode#1{%
52.262   \ifnum0=#1
52.263     \def\cs@next{\cs@nosplit}%
52.264   \else
52.265     \advance\cs@wordlen\@ne
52.266     \ifnum\righthyphenmin>\the\cs@wordlen
52.267       \def\cs@next{\cs@continuescan}%
52.268     \else
52.269       \cs@splithyphen
52.270       \def\cs@next{\cs@word}%
52.271     \fi
52.272   \fi \cs@next}

```

\cs@nosplit Insert a non-breakable hyphen followed by the saved word.

```

52.273 \def\cs@nosplit{\cs@boxhyphen\cs@word}

```

\cs@hyphen The `\minus` sequence can be used where the active hyphen does not work, e.g. in arguments to \TeX primitives in outer horizontal mode.

```

52.274 \let\minus\cs@hyphen

```

\standardhyphens These macros control whether split hyphens are allowed in Czech and/or Slovak texts. You may use them in any language, but the split hyphen is only activated for Czech and Slovak.

```

52.275 \def\standardhyphens{\cs@splithyphensfalse\cs@deactivatehyphens}
52.276 \def\splithyphens{\cs@splithyphenstrue\cs@activatehyphens}

```

\cs@splitattr Now we declare the `split` language attribute. This is similar to the `split` package option of `cslatex`, but it only affects Czech, not Slovak.

```

52.277 \def\cs@splitattr{\babel@save\ifcs@splithyphens\splithyphens}
52.278 \bbl@declare@ttribute{czech}{split}{%
52.279   \addto\extrasczech{\cs@splitattr}}

```

\cs@activatehyphens These macros are defined as `\relax` by default to prevent activating/deactivating the hyphen character. They are redefined when the language is switched to Czech/Slovak. At that moment the hyphen is also activated if split hyphens were requested with `\splithyphens`.

When the language is de-activated, de-activate the hyphen and restore the bogus definitions of these macros.

```

52.280 \let\cs@activatehyphens\relax
52.281 \let\cs@deactivatehyphens\relax
52.282 \expandafter\addto\csname extras\CurrentOption\endcsname{%
52.283   \def\cs@activatehyphens{\bbl@activate{-}}%
52.284   \def\cs@deactivatehyphens{\bbl@deactivate{-}}%
52.285   \ifcs@splithyphens\cs@activatehyphens\fi}
52.286 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
52.287   \cs@deactivatehyphens
52.288   \let\cs@activatehyphens\relax
52.289   \let\cs@deactivatehyphens\relax}

```

\cs@looseness One of the most common situations where an active hyphen will not work properly is the `\looseness` primitive. Change its definition so that it deactivates the hyphen if needed.

```

52.290 \let\cs@looseness\looseness
52.291 \def\looseness{%
52.292   \ifcs@splithyphens
52.293     \cs@deactivatehyphens\afterassignment\cs@activatehyphens \fi
52.294   \cs@looseness}

```


`\cs@selectlanguage` Specifying the `nocaptions` option means that captions and dates are not redefined by default, but they can be switched on later with `\captionsczech` and/or `\dateczech`.

We mimic this behavior by redefining `\selectlanguage`. This macro is called once at the beginning of the document to set the main language of the document. If this is `\cs@main@language`, it disables the macros for setting captions and date. In any case, it restores the original definition of `\selectlanguage` and expands it.

The definition of `\selectlanguage` can be shared between Czech and Slovak; the actual language is stored in `\cs@main@language`.

```

52.295 \ifx\cs@nocaptions\@undefined\else
52.296   \edef\cs@main@language{\CurrentOption}
52.297   \ifx\cs@origselect\@undefined
52.298     \let\cs@origselect=\selectlanguage
52.299     \def\selectlanguage{%
52.300       \let\selectlanguage\cs@origselect
52.301       \ifx\bbl@main@language\cs@main@language
52.302         \expandafter\cs@selectlanguage
52.303       \else
52.304         \expandafter\selectlanguage
52.305       \fi}
52.306   \def\cs@selectlanguage{%
52.307     \cs@tempdisable{captions}%
52.308     \cs@tempdisable{date}%
52.309     \selectlanguage}

```

`\cs@tempdisable` `\cs@tempdisable` disables a language setup macro temporarily, i.e. the macro with the name of `(#1)\bbl@main@language` just restores the original definition and purges the saved macro from memory.

```

52.310   \def\cs@tempdisable#1{%
52.311     \def\@tempa{cs@#1}%
52.312     \def\@tempb{#1\bbl@main@language}%
52.313     \expandafter\expandafter\expandafter\let
52.314       \expandafter \csname\expandafter \@tempa \expandafter\endcsname
52.315       \csname \@tempb \endcsname
52.316     \expandafter\edef\csname \@tempb \endcsname{%
52.317       \let \expandafter\noexpand \csname \@tempb \endcsname
52.318       \expandafter\noexpand \csname \@tempa \endcsname
52.319       \let \expandafter\noexpand\csname \@tempa \endcsname
52.320       \noexpand\@undefined}}

```

These macros are not needed, once the initialization is over.

```

52.321   \@onlypreamble\cs@main@language
52.322   \@onlypreamble\cs@origselect
52.323   \@onlypreamble\cs@selectlanguage
52.324   \@onlypreamble\cs@tempdisable
52.325 \fi
52.326 \fi

```

The encoding of mathematical fonts should be changed to IL2. This allows to use accented letter in some font families. Besides, documents do not use CM fonts if there are equivalents in CS-fonts, so there is no need to have both bitmaps of CM-font and CS-font.

`\@font@warning` and `\@font@info` are temporarily redefined to avoid annoying font warnings.

```

52.327 \ifx\cs@compat@plain\@undefined
52.328 \ifx\cs@check@enc\@undefined\else
52.329   \def\cs@check@enc{
52.330     \ifx\encodingdefault\cs@iltw@
52.331       \let\cs@warn\@font@warning \let\@font@warning\@gobble

```

```

52.332 \let\cs@info\@font@info \let\@font@info\@gobble
52.333 \SetSymbolFont{operators}{normal}{\cs@iltw@}{cmr}{m}{n}
52.334 \SetSymbolFont{operators}{bold}{\cs@iltw@}{cmr}{bx}{n}
52.335 \SetMathAlphabet\mathbf{normal}{\cs@iltw@}{cmr}{bx}{n}
52.336 \SetMathAlphabet\mathit{normal}{\cs@iltw@}{cmr}{m}{it}
52.337 \SetMathAlphabet\mathrm{normal}{\cs@iltw@}{cmr}{m}{n}
52.338 \SetMathAlphabet\mathsf{normal}{\cs@iltw@}{cmss}{m}{n}
52.339 \SetMathAlphabet\mathtt{normal}{\cs@iltw@}{cmtt}{m}{n}
52.340 \SetMathAlphabet\mathbf{bold}{\cs@iltw@}{cmr}{bx}{n}
52.341 \SetMathAlphabet\mathit{bold}{\cs@iltw@}{cmr}{m}{it}
52.342 \SetMathAlphabet\mathrm{bold}{\cs@iltw@}{cmr}{bx}{n}
52.343 \SetMathAlphabet\mathsf{bold}{\cs@iltw@}{cmss}{bx}{n}
52.344 \SetMathAlphabet\mathtt{bold}{\cs@iltw@}{cmtt}{m}{n}
52.345 \let\@font@warning\cs@warn \let\cs@warn\@undefined
52.346 \let\@font@info\cs@info \let\cs@info\@undefined
52.347 \fi
52.348 \let\cs@check@enc\@undefined}
52.349 \AtBeginDocument{\cs@check@enc}
52.350 \fi
52.351 \fi

```

`cs@undoiltw@` The thing is that L^AT_EX 2_ε core only supports the T1 encoding and does not bother changing the uc/lc/sf codes when encoding is switched. :(However, the IL2 encoding *does* change these codes, so if encoding is switched back from IL2, we must also undo the effect of this change to be compatible with L^AT_EX 2_ε. OK, this is not the rightTM solution but it works. Cheers to Petr Olšák.

```

52.352 \def\cs@undoiltw@{%
52.353 \uccode158=208 \lccode158=158 \sfcode158=1000
52.354 \sfcode159=1000
52.355 \uccode165=133 \lccode165=165 \sfcode165=1000
52.356 \uccode169=137 \lccode169=169 \sfcode169=1000
52.357 \uccode171=139 \lccode171=171 \sfcode171=1000
52.358 \uccode174=142 \lccode174=174 \sfcode174=1000
52.359 \uccode181=149
52.360 \uccode185=153
52.361 \uccode187=155
52.362 \uccode190=0 \lccode190=0
52.363 \uccode254=222 \lccode254=254 \sfcode254=1000
52.364 \uccode255=223 \lccode255=255 \sfcode255=1000}

```

`@@enc@update` Redefine the L^AT_EX 2_ε internal function `\@@enc@update` to change the encodings correctly.

```

52.365 \ifx\cs@enc@update\@undefined
52.366 \ifx\@@enc@update\@undefined\else
52.367 \let\cs@enc@update\@@enc@update
52.368 \def\@@enc@update{\ifx\cf@encoding\cs@iltw@\cs@undoiltw@\fi
52.369 \cs@enc@update
52.370 \expandafter\ifnum\csname l@\language\endcsname=\the\language
52.371 \expandafter\ifx
52.372 \csname l@\language:\f@encoding\endcsname\relax
52.373 \else
52.374 \expandafter\expandafter\expandafter\let
52.375 \expandafter\csname
52.376 \expandafter l\expandafter @\expandafter\language
52.377 \expandafter\endcsname\csname l@\language:\f@encoding\endcsname
52.378 \fi
52.379 \language=\csname l@\language\endcsname\relax
52.380 \fi}
52.381 \fi\fi

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the

category code of @ to its original value.

52.382 \ldf@finish\CurrentOption

52.383 </code>

53 The Polish language

The file `polish.dtx`⁶² defines all the language-specific macros for the Polish language.

For this language the character `"` is made active. In table 25 an overview is given of its purpose.

<code>"a</code>	or <code>\aob</code> , for tailed-a (like <i>ą</i>)
<code>"A</code>	or <code>\Aob</code> , for tailed-A (like <i>Ą</i>)
<code>"e</code>	or <code>\eob</code> , for tailed-e (like <i>ę</i>)
<code>"E</code>	or <code>\Eob</code> , for tailed-E (like <i>Ę</i>)
<code>"c</code>	or <code>\'c</code> , for accented c (like <i>ć</i>), same with uppercase letters and n,o,s
<code>"l</code>	or <code>\lpb{}</code> , for l with stroke (like <i>ł</i>)
<code>"L</code>	or <code>\Lpb{}</code> , for L with stroke (like <i>Ł</i>)
<code>"r</code>	or <code>\zkb{}</code> , for pointed z (like <i>ż</i>), cf. pronunciation
<code>"R</code>	or <code>\Zkb{}</code> , for pointed Z (like <i>Ż</i>)
<code>"z</code>	or <code>\'z</code> , for accented z
<code>"Z</code>	or <code>\'Z</code> , for accented Z
<code>" </code>	disable ligature at this position.
<code>"-</code>	an explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>" "</code>	like <code>"-</code> , but producing no hyphen sign (for compound words with hyphen, e.g. <i>x-"y</i>).
<code>"‘</code>	for German left double quotes (looks like <i>„</i>).
<code>"’</code>	for German right double quotes.
<code>"<</code>	for French left double quotes (similar to <i><<</i>).
<code>"></code>	for French right double quotes (similar to <i>>></i>).

Table 25: The extra definitions made by `polish.sty`

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
53.1 (*code)
53.2 \LdfInit{polish}\captionspolish
```

When this file is read as an option, i.e. by the `\usepackage` command, `polish` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@polish` to see whether we have to do something here.

```
53.3 \ifx\l@polish\@undefined
53.4 \@nopatterns{Polish}
53.5 \adddialect\l@polish0\fi
```

The next step consists of defining commands to switch to (and from) the Polish language.

`\captionspolish` The macro `\captionspolish` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
53.6 \addto\captionspolish{%
53.7   \def\prefacename{Przedmowa}%
53.8   \def\refname{Literatura}%
53.9   \def\abstractname{Streszczenie}%
53.10  \def\bibname{Bibliografia}%
53.11  \def\chaptername{Rozdział}%
53.12  \def\appendixname{Dodatek}%
53.13  \def\contentsname{Spis treści}%
53.14  \def\listfigurename{Spis rysunków}%
53.15  \def\listtablename{Spis tabel}%
```

⁶²The file described in this section has version number v1.2l and was last revised on 2005/03/31.

```

53.16 \def\indexname{Indeks}%
53.17 \def\figurename{Rysunek}%
53.18 \def\tablename{Tablica}%
53.19 \def\partname{Cz\eob{}}\s{c}%
53.20 \def\enclname{Za\l\ aob{}}cznik}%
53.21 \def\ccname{Kopie:}%
53.22 \def\headtoname{Do}%
53.23 \def\pagename{Strona}%
53.24 \def\seename{Por\ownaj}%
53.25 \def\alsoname{Por\ownaj tak.ze}%
53.26 \def\proofname{Dow\od}%
53.27 \def\glossaryname{Glossary}% <-- Needs translation
53.28 }

```

`\datepolish` The macro `\datepolish` redefines the command `\today` to produce Polish dates.

```

53.29 \def\datepolish{%
53.30 \def\today{\number\day~\ifcase\month\or
53.31 stycznia\or lutego\or marca\or kwietnia\or maja\or czerwca\or lipca\or
53.32 sierpnia\or wrze\snia\or pa\zdziernika\or listopada\or grudnia\fi
53.33 \space\number\year}%
53.34 }

```

`\extrapolish` The macro `\extrapolish` will perform all the extra definitions needed for the Polish language. The macro `\noextrapolish` is used to cancel the actions of `\extrapolish`.

For Polish the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the polish group of shorthands should be used.

```

53.35 \initiate@active@char{"}
53.36 \addto\extrapolish{\languageshorthands{polish}}
53.37 \addto\extrapolish{\bbl@activate{}}

```

Don't forget to turn the shorthands off again.

```

53.38 \addto\noextrapolish{\bbl@deactivate{}}

```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 25.

If you have problems at the end of a word with a linebreak, use the other version without hyphenation tricks. Some TeX wizard may produce a better solution with forecasting another token to decide whether the character after the double quote is the last in a word. Do it and let us know.

In Polish texts some letters get special diacritical marks. Leszek Holenderski designed the following code to position the diacritics correctly for every font in every size. These macros need a few extra dimension variables.

```

53.39 \newdimen\pl@left
53.40 \newdimen\pl@down
53.41 \newdimen\pl@right
53.42 \newdimen\pl@temp

```

`\sob` The macro `\sob` is used to put the 'ogonek' in the right place.

```

53.43 \def\sob#1#2#3#4#5{%parameters: letter and fractions hl,ho,vl,vo
53.44 \setbox0\hbox{#1}\setbox1\hbox{$\_mathchar'454$}\setbox2\hbox{p}%
53.45 \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
53.46 \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
53.47 \pl@left=\pl@right \advance\pl@left by\wd1
53.48 \pl@temp=-\pl@down \advance\pl@temp by\dp2 \dp1=\pl@temp
53.49 \leavevmode
53.50 \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}

```

`\aob` The ogonek is placed with the letters ‘a’, ‘A’, ‘e’, and ‘E’.

```
\Aob 53.51 \DeclareTextCommand{\aob}{OT1}{\sob a{.66}{.20}{0}{.90}}
\eob 53.52 \DeclareTextCommand{\Aob}{OT1}{\sob A{.80}{.50}{0}{.90}}
\Eob 53.53 \DeclareTextCommand{\eob}{OT1}{\sob e{.50}{.35}{0}{.93}}
53.54 \DeclareTextCommand{\Eob}{OT1}{\sob E{.60}{.35}{0}{.90}}
```

For the ‘new’ T1 encoding we can provide simpler definitions.

```
53.55 \DeclareTextCommand{\aob}{T1}{\k a}
53.56 \DeclareTextCommand{\Aob}{T1}{\k A}
53.57 \DeclareTextCommand{\eob}{T1}{\k e}
53.58 \DeclareTextCommand{\Eob}{T1}{\k E}
```

Construct the characters by default from the OT1 encoding.

```
53.59 \ProvideTextCommandDefault{\aob}{\UseTextSymbol{OT1}{\aob}}
53.60 \ProvideTextCommandDefault{\Aob}{\UseTextSymbol{OT1}{\Aob}}
53.61 \ProvideTextCommandDefault{\eob}{\UseTextSymbol{OT1}{\eob}}
53.62 \ProvideTextCommandDefault{\Eob}{\UseTextSymbol{OT1}{\Eob}}
```

`\spb` The macro `\spb` is used to put the ‘poprzeczka’ in the right place.

```
53.63 \def\spb#1#2#3#4#5{%
53.64   \setbox0\hbox{#1}\setbox1\hbox{\char'023}%
53.65   \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
53.66   \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
53.67   \pl@left=\pl@right \advance\pl@left by\wd1
53.68   \ht1=\pl@down \dp1=-\pl@down
53.69   \leavevmode
53.70   \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
```

`\skb` The macro `\skb` is used to put the ‘kropka’ in the right place.

```
53.71 \def\skb#1#2#3#4#5{%
53.72   \setbox0\hbox{#1}\setbox1\hbox{\char'056}%
53.73   \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
53.74   \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
53.75   \pl@left=\pl@right \advance\pl@left by\wd1
53.76   \leavevmode
53.77   \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
```

`\textpl` For the ‘poprzeczka’ and the ‘kropka’ in text fonts we don’t need any special coding, but we can (almost) use what is already available.

```
53.78 \def\textpl{%
53.79   \def\lpb{\p111}%
53.80   \def\Lpb{\pLLL}%
53.81   \def\zkb{\.z}%
53.82   \def\Zkb{\.Z}}
```

Initially we assume that typesetting is done with text fonts.

```
53.83 \textpl
53.84 \let\l11=\l1 \let\LLL=\L
53.85 \def\p111{\l11}
53.86 \def\pLLL{\LLL}
```

`\telepl` But for the ‘teletype’ font in ‘OT1’ encoding we have to take some special actions, involving the macros defined above.

```
53.87 \def\telepl{%
53.88   \def\lpb{\spb l{.45}{.5}{.4}{.8}}%
53.89   \def\Lpb{\spb L{.23}{.5}{.4}{.8}}%
53.90   \def\zkb{\skb z{.5}{.5}{1.2}{0}}%
53.91   \def\Zkb{\skb Z{.5}{.5}{1.1}{0}}}
```

To activate these codes the font changing commands as they are defined in L^AT_EX are modified. The same is done for plain T_EX's font changing commands.

When `\selectfont` is undefined the current format is supposed to be either plain (based) or L^AT_EX 2.09.

```

53.92 \ifx\selectfont\@undefined
53.93   \ifx\prm\@undefined \addto\rm{\textpl}\else \addto\prm{\textpl}\fi
53.94   \ifx\pit\@undefined \addto\it{\textpl}\else \addto\pit{\textpl}\fi
53.95   \ifx\pbf\@undefined \addto\bf{\textpl}\else \addto\pbf{\textpl}\fi
53.96   \ifx\psl\@undefined \addto\sl{\textpl}\else \addto\psl{\textpl}\fi
53.97   \ifx\psf\@undefined \else \addto\psf{\textpl}\fi
53.98   \ifx\psc\@undefined \else \addto\psc{\textpl}\fi
53.99   \ifx\ptt\@undefined \addto\tt{\telepl}\else \addto\ptt{\telepl}\fi
53.100 \else

```

When `\selectfont` exists we assume L^AT_EX 2_ε.

```

53.101 \expandafter\addto\csname selectfont \endcsname{%
53.102   \csname\fontencoding @pl\endcsname}
53.103 \fi

```

Currently we support the OT1 and T1 encodings. For T1 we don't have to make a difference between typewriter fonts and other fonts, they all have the same glyphs.

```

53.104 \expandafter\let\csname T1@pl\endcsname\textpl

```

For OT1 we need to check the current font family, stored in `\fontfamily`. Unfortunately we need a hack as `\ttdefault` is defined as a `\long` macro, while `\fontfamily` is not.

```

53.105 \expandafter\def\csname OT1@pl\endcsname{%
53.106   \long\edef\curr@family{\fontfamily}%
53.107   \ifx\curr@family\ttdefault
53.108     \telepl
53.109   \else
53.110     \textpl
53.111   \fi}

```

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\'` can now be typed as `"`.

```

53.112 \begingroup \catcode'\''12
53.113 \def\x{\endgroup
53.114   \def\dq{"}}
53.115 \x

```

Now we can define the doublequote macros for diacritics,

```

53.116 \declare@shorthand{polish}{a}{\textormath{\aob}{\ddot a}}
53.117 \declare@shorthand{polish}{A}{\textormath{\Aob}{\ddot A}}
53.118 \declare@shorthand{polish}{c}{\textormath{'c}{\acute c}}
53.119 \declare@shorthand{polish}{C}{\textormath{'C}{\acute C}}
53.120 \declare@shorthand{polish}{e}{\textormath{\eob}{\ddot e}}
53.121 \declare@shorthand{polish}{E}{\textormath{\Eob}{\ddot E}}
53.122 \declare@shorthand{polish}{l}{\textormath{\l pb}{\ddot l}}
53.123 \declare@shorthand{polish}{L}{\textormath{\L pb}{\ddot L}}
53.124 \declare@shorthand{polish}{n}{\textormath{'n}{\acute n}}
53.125 \declare@shorthand{polish}{N}{\textormath{'N}{\acute N}}
53.126 \declare@shorthand{polish}{o}{\textormath{'o}{\acute o}}
53.127 \declare@shorthand{polish}{O}{\textormath{'O}{\acute O}}
53.128 \declare@shorthand{polish}{s}{\textormath{'s}{\acute s}}
53.129 \declare@shorthand{polish}{S}{\textormath{'S}{\acute S}}

```

`\polishrz` The command `\polishrz` defines the shorthands `"r`, `"z` and `"x` to produce pointed `z`, accented `z` and `"x`. This is the default as these shorthands were defined by this language definition file for quite some time.

```

53.130 \newcommand*{\polishrz}{%
53.131   \declare@shorthand{polish}{r}{\textormath{\z kb}{\ddot r}}}%

```

```

53.132 \declare@shorthand{polish}{\R}{\textormath{\Zkb}{\ddot R}}%
53.133 \declare@shorthand{polish}{\z}{\textormath{\Zkb}{\acute z}}%
53.134 \declare@shorthand{polish}{\Z}{\textormath{\Zkb}{\acute Z}}%
53.135 \declare@shorthand{polish}{\x}{\dq x}%
53.136 \declare@shorthand{polish}{\X}{\dq X}%
53.137 }
53.138 \polishrz

```

The command `\polishzx` switches to a different set of shorthands, "z", "x" and "r" to produce pointed z, accented z and "r"; a different shorthand notation also in use.

```

53.139 \newcommand*{\polishzx}{%
53.140 \declare@shorthand{polish}{\z}{\textormath{\Zkb}{\ddot z}}%
53.141 \declare@shorthand{polish}{\Z}{\textormath{\Zkb}{\ddot Z}}%
53.142 \declare@shorthand{polish}{\x}{\textormath{\Zkb}{\acute x}}%
53.143 \declare@shorthand{polish}{\X}{\textormath{\Zkb}{\acute X}}%
53.144 \declare@shorthand{polish}{\r}{\dq r}%
53.145 \declare@shorthand{polish}{\R}{\dq R}%
53.146 }

```

Then we define access to two forms of quotation marks, similar to the german and french quotation marks.

```

53.147 \declare@shorthand{polish}{\'}{%
53.148 \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
53.149 \declare@shorthand{polish}{\'}{%
53.150 \textormath{\textquotedblright}{\mbox{\textquotedblright}}}
53.151 \declare@shorthand{polish}{\<}{%
53.152 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
53.153 \declare@shorthand{polish}{\>}{%
53.154 \textormath{\guillemotright}{\mbox{\guillemotright}}}

```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from `\-`.

```

53.155 \declare@shorthand{polish}{\nobreak-}{\nobreak-\bbl@allowhyphens}
53.156 \declare@shorthand{polish}{\hskip}{\hskip\z@skip}

```

And we want to have a shorthand for disabling a ligature.

```

53.157 \declare@shorthand{polish}{\|}{%
53.158 \textormath{\discretionary{-}{-}{\kern.03em}}{}}

```

`\mdqon` All that's left to do now is to define a couple of commands for reasons of compatibility with `polish.tex`.

```

53.159 \def\mdqon{\shorthandon{}}
53.160 \def\mdqoff{\shorthandoff{}}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

53.161 \ldf@finish{polish}
53.162 \code

```


54 The Serbocroatian language

The file `serbian.dtx`⁶³ defines all the language definition macros for the Serbian language, typeset in a latin script. In a future version support for typesetting in a cyrillic script may be added.

For this language the character " is made active. In table 26 an overview is given of its purpose. One of the reasons for this is that in the Serbian language some special characters are used.

"c	\c, also implemented for the lowercase and uppercase s and z.
"d	\dj, also implemented for "D
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
"	disable ligature at this position
"	like "-", but producing no hyphen sign (for compound words with hyphen, e.g. x-"y).
"‘	for Serbian left double quotes (looks like „).
"’	for Serbian right double quotes.
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).

Table 26: The extra definitions made by `serbian.ldf`

Apart from defining shorthands we need to make sure taht the first paragraph of each section is intended. Furthermore the following new math operators are defined (`\tg`, `\ctg`, `\arctg`, `\arcctg`, `\sh`, `\ch`, `\th`, `\cth`, `\arsh`, `\arch`, `\arth`, `\arcth`, `\Prob`, `\Expect`, `\Variance`).

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
54.1 (*code)
54.2 \LdfInit{serbian}\captionsserbian
```

When this file is read as an option, i.e. by the `\usepackage` command, `serbian` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@serbian` to see whether we have to do something here.

```
54.3 \ifx\l@serbian\@undefined
54.4     \nopatterns{Serbian}
54.5     \adddialect\l@serbian0\fi
```

The next step consists of defining commands to switch to (and from) the Serbocroatian language.

`\captionsserbian` The macro `\captionsserbian` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
54.6 \addto\captionsserbian{%
54.7     \def\prefacename{Predgovor}%
54.8     \def\refname{Literatura}%
54.9     \def\abstractname{Sa\v{z}etak}%
54.10    \def\bibname{Bibliografija}%
54.11    \def\chaptername{Glava}%
54.12    \def\appendixname{Dodatak}%
54.13    \def\contentsname{Sadr\v{z}aj}%
54.14    \def\listfigurename{Slike}%
54.15    \def\listtablename{Tabele}%
54.16    \def\indexname{Indeks}%
```

⁶³The file described in this section has version number v1.0d and was last revised on 2005/03/31. A contribution was made by Dejan Muhamedagić (dejan@yunix.com).

```

54.17 \def\figurename{Slika}%
54.18 \def\tablename{Tabela}%
54.19 \def\partname{Deo}%
54.20 \def\enclname{Prilozi}%
54.21 \def\ccname{Kopije}%
54.22 \def\headtoname{Prima}%
54.23 \def\pagename{Strana}%
54.24 \def\seenname{Vidi}%
54.25 \def\alsoname{Vidi tako\dj e}%
54.26 \def\proofname{Dokaz}%
54.27 \def\glossaryname{Glossary}% <-- Needs translation
54.28 }%

```

`\dateserbian` The macro `\dateserbian` redefines the command `\today` to produce Serbocroatian dates.

```

54.29 \def\dateserbian{%
54.30 \def\today{\number\day .~\ifcase\month\or
54.31   januar\or februar\or mart\or april\or maj\or
54.32   juni\or juli\or avgust\or septembar\or oktobar\or novembar\or
54.33   decembar\fi \space \number\year}}

```

`\extrasserbian` The macro `\extrasserbian` will perform all the extra definitions needed for the Serbocroatian language. The macro `\noextrasserbian` is used to cancel the actions of `\extrasserbian`.

For Serbian the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the serbian group of shorthands should be used.

```

54.34 \initiate@active@char{"}
54.35 \addto\extrasserbian{\languageshorthands{serbian}}
54.36 \addto\extrasserbian{\bbl@activate{}}

```

Don't forget to turn the shorthands off again.

```

54.37 \addto\noextrasserbian{\bbl@deactivate{}}

```

First we define shorthands to facilitate the occurrence of letters such as č.

```

54.38 \declare@shorthand{serbian}{"c"}{\textormath{\v c}{\check c}}
54.39 \declare@shorthand{serbian}{"d"}{\textormath{\dj}{\dj}}%%
54.40 \declare@shorthand{serbian}{"s"}{\textormath{\v s}{\check s}}
54.41 \declare@shorthand{serbian}{"z"}{\textormath{\v z}{\check z}}
54.42 \declare@shorthand{serbian}{"C"}{\textormath{\v C}{\check C}}
54.43 \declare@shorthand{serbian}{"D"}{\textormath{\DJ}{\DJ}}%%
54.44 \declare@shorthand{serbian}{"S"}{\textormath{\v S}{\check S}}
54.45 \declare@shorthand{serbian}{"Z"}{\textormath{\v Z}{\check Z}}

```

Then we define access to two forms of quotation marks, similar to the german and french quotation marks.

```

54.46 \declare@shorthand{serbian}{"'"}{%
54.47   \textormath{\quotedblbase}}{\mbox{\quotedblbase}}
54.48 \declare@shorthand{serbian}{"'"}{%
54.49   \textormath{\textquotedblleft}}{\mbox{\textquotedblleft}}
54.50 \declare@shorthand{serbian}{"<"}{%
54.51   \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
54.52 \declare@shorthand{serbian}{">"}{%
54.53   \textormath{\guillemotright}}{\mbox{\guillemotright}}

```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```

54.54 \declare@shorthand{serbian}{"-"}{\nobreak-\bbl@allowhyphens}
54.55 \declare@shorthand{serbian}{"~"}{\hspace{\z@skip}}

```

And we want to have a shorthand for disabling a ligature.

```

54.56 \declare@shorthand{serbian}{"|"}{%
54.57   \textormath{\discretionary{-}{-}{\kern.03em}}{}}

```

`\bbl@frenchindent` In Serbian the first paragraph of each section should be indented. Add this code
`\bbl@nonfrenchindent` only in L^AT_EX.

```

54.58 \ifx\fmtname plain \else
54.59   \let\@aifORI\@afterindentfalse
54.60   \def\bbl@frenchindent{\let\@afterindentfalse\@afterindenttrue
54.61                               \@afterindenttrue}
54.62   \def\bbl@nonfrenchindent{\let\@afterindentfalse\@aifORI
54.63                               \@afterindentfalse}
54.64   \addto\extrasserbian{\bbl@frenchindent}
54.65   \addto\noextrasserbian{\bbl@nonfrenchindent}
54.66 \fi

```

`\mathserbian` Some math functions in Serbian math books have other names: e.g. `\sinh` in Serbian is written as `sh` etc. So we define a number of new math operators.

```

54.67 \def\sh{\mathop{\operator@font sh}\nolimits} % same as \sinh
54.68 \def\ch{\mathop{\operator@font ch}\nolimits} % same as \cosh
54.69 \def\th{\mathop{\operator@font th}\nolimits} % same as \tanh
54.70 \def\cth{\mathop{\operator@font cth}\nolimits} % same as \coth
54.71 \def\arsh{\mathop{\operator@font arsh}\nolimits}
54.72 \def\arch{\mathop{\operator@font arch}\nolimits}
54.73 \def\arth{\mathop{\operator@font arth}\nolimits}
54.74 \def\arcth{\mathop{\operator@font arcth}\nolimits}
54.75 \def\tg{\mathop{\operator@font tg}\nolimits} % same as \tan
54.76 \def\ctg{\mathop{\operator@font ctg}\nolimits} % same as \cot
54.77 \def\arctg{\mathop{\operator@font arctg}\nolimits} % same as \arctan
54.78 \def\arcctg{\mathop{\operator@font arcctg}\nolimits}
54.79 \def\Prob{\mathop{\mathsf P\hskip0pt}\nolimits}
54.80 \def\Expect{\mathop{\mathsf E\hskip0pt}\nolimits}
54.81 \def\Variance{\mathop{\mathsf D\hskip0pt}\nolimits}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

54.82 \ldf@finish{serbian}
54.83 \code

```

55 The Slovak language

The file `slovak.dtx`⁶⁴ defines all the language-specific macros for the Slovak language.

For this language the macro `\q` is defined. It was used with the letters (`ť`, `d`, `l`, and `L`) and adds a `'` to them to simulate a ‘hook’ that should be there. The result looks like `ť`. Since the the T1 font encoding has the corresponding characters it is mapped to `\v`. Therefore we recommend using T1 font encoding. If you don’t want to use this encoding, please, feel free to redefine `\q` in your file. I think babel will honour this ;-).

For this language the characters `"`, `'` and `~` are made active. In table 27 an overview is given of its purpose. Also the vertical placement of the umlaut can be controlled this way.

<code>"a</code>	<code>\"a</code> , also implemented for the other lowercase and uppercase vowels.
<code>~d</code>	<code>\q d</code> , also implemented for <code>l</code> , <code>t</code> and <code>L</code> .
<code>~c</code>	<code>\v c</code> , also implemented for <code>C</code> , <code>D</code> , <code>N</code> , <code>n</code> , <code>T</code> , <code>Z</code> and <code>z</code> .
<code>~o</code>	<code>\^o</code> , also implemented for <code>O</code> .
<code>'a</code>	<code>\'a</code> , also implemented for the other lowercase and uppercase <code>l</code> , <code>r</code> , <code>y</code> and vowels.
<code>" </code>	disable ligature at this position.
<code>"-</code>	an explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>" "</code>	like <code>"-</code> , but producing no hyphen sign (for compound words with hyphen, e.g. <code>x-" "y</code>).
<code>"~</code>	for a compound word mark without a breakpoint.
<code>"=</code>	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
<code>"‘</code>	for German left double quotes (looks like <code>„</code>).
<code>"’</code>	for German right double quotes.
<code>"<</code>	for French left double quotes (similar to <code><<</code>).
<code>"></code>	for French right double quotes (similar to <code>>></code>).

Table 27: The extra definitions made by `slovak.ldf`

The quotes in table 27 can also be typeset by using the commands in table 28.

<code>\glqq</code>	for German left double quotes (looks like <code>„</code>).
<code>\grqq</code>	for German right double quotes (looks like <code>“</code>).
<code>\glq</code>	for German left single quotes (looks like <code>‚</code>).
<code>\grq</code>	for German right single quotes (looks like <code>‘</code>).
<code>\flqq</code>	for French left double quotes (similar to <code><<</code>).
<code>\frqq</code>	for French right double quotes (similar to <code>>></code>).
<code>\flq</code>	for (French) left single quotes (similar to <code><</code>).
<code>\frq</code>	for (French) right single quotes (similar to <code>></code>).
<code>\dq</code>	the original quotes character (<code>"</code>).
<code>\sq</code>	the original single quote (<code>'</code>).

Table 28: More commands which produce quotes, defined by `slovak.ldf`

⁶⁴The file described in this section has version number v3.1a and was last revised on 2008/07/06. It was originally written by Jana Chlebková (`chlebk@euromath.dk`) and modified by Tobias Schlemmer (`Tobias.Schlemmer@web.de`). It was then rewritten by Petr Tesařík (`babel@tesarici.cz`).

55.1 Compatibility

Great care has been taken to ensure backward compatibility with $\mathcal{CS}\text{L}\text{A}\text{T}\text{E}\text{X}$. In particular, documents which load this file with `\usepackage{slovak}` should produce identical output with no modifications to the source. Additionally, all the $\mathcal{CS}\text{L}\text{A}\text{T}\text{E}\text{X}$ options are recognized:

IL2, T1, OT1

These options set the default font encoding. Please note that their use is deprecated. You should use the `fontenc` package to select font encoding.

split, nosplit

These options control whether hyphenated words are automatically split according to Slovak typesetting rules. With the `split` option “je-li” is hyphenated as “je-/li”. The `nosplit` option disables this behavior.

The use of this option is strongly discouraged, as it breaks too many common things—hyphens cannot be used in labels, negative arguments to TEX primitives will not work in horizontal mode (use `\minus` as a workaround), and there are a few other peculiarities with using this mode.

nocaptions

This option was used in $\mathcal{CS}\text{L}\text{A}\text{T}\text{E}\text{X}$ to set up Czech/Slovak typesetting rules, but leave the original captions and dates. The recommended way to achieve this is to use English as the main language of the document and use the environment `otherlanguage*` for Czech text.

olduv There are two version of `\uv`. The older one allows the use of `\verb` inside the quotes but breaks any respective kerning with the quotes (like that in \mathcal{CS} fonts). The newer one honors the kerning in the font but does not allow `\verb` inside the quotes.

The new version is used by default in $\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$ and the old version is used with plain TEX . You may use `olduv` to override the default in $\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$.

cstex This option was used to include the commands `\csprimeson` and `\csprimesoff`. Since these commands are always included now, it has been removed and the empty definition lasts for compatibility.

55.2 Implementation

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

55.1 `{*code}`

55.2 `\LdfInit\CurrentOption{date\CurrentOption}`

When this file is read as an option, i.e. by the `\usepackage` command, `slovak` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@slovak` to see whether we have to do something here.

55.3 `\ifx\l@slovak\@undefined`

55.4 `\@nopatterns{Slovak}`

55.5 `\adddialect\l@slovak0\fi`

We need to define these macros early in the process.

55.6 `\def\cs@iltw@{IL2}`

55.7 `\newif\ifcs@splithyphens`

55.8 `\cs@splithyphensfalse`

If Babel is not loaded, we provide compatibility with $\text{\textit{CSLATEX}}$. However, if macro $\text{\textit{@ifpackageloaded}}$ is not defined, we assume to be loaded from plain and provide compatibility with $\text{\textit{cspain}}$. Of course, this does not work well with $\text{\textit{LATEX}}$ 2.09, but I doubt anyone will ever want to use this file with $\text{\textit{LATEX}}$ 2.09.

```

55.9 \ifx\@ifpackageloaded\@undefined
55.10 \let\cs@compat@plain\relax
55.11 \message{cspain compatibility mode}
55.12 \else
55.13 \@ifpackageloaded{babel}{\%
55.14 \let\cs@compat@latex\relax
55.15 \message{cslatex compatibility mode}}
55.16 \fi
55.17 \ifx\cs@compat@latex\relax
55.18 \ProvidesPackage{slovak}[2008/07/06 v3.1a CTeX Slovak style]

```

Declare $\text{\textit{CSLATEX}}$ options (see also the descriptions on page 277).

```

55.19 \DeclareOption{IL2}{\def\encodingdefault{IL2}}
55.20 \DeclareOption{T1}{\def\encodingdefault{T1}}
55.21 \DeclareOption{OT1}{\def\encodingdefault{OT1}}
55.22 \DeclareOption{nosplit}{\cs@splithyphensfalse}
55.23 \DeclareOption{split}{\cs@splithyphenstrue}
55.24 \DeclareOption{nocaptions}{\let\cs@nocaptions=\relax}
55.25 \DeclareOption{olduv}{\let\cs@olduv=\relax}
55.26 \DeclareOption{cstex}{\relax}

```

Make IL2 encoding the default. This can be overridden with the other font encoding options.

```

55.27 \ExecuteOptions{\cs@iltw@}

```

Now, process the user-supplied options.

```

55.28 \ProcessOptions

```

Standard $\text{\textit{LATEX}}_{2\epsilon}$ does not include the IL2 encoding in the format. The encoding can be loaded later using the fontenc package, but $\text{\textit{CSLATEX}}$ included IL2 by default. This means existing documents for $\text{\textit{CSLATEX}}$ do not load that package, so load the encoding ourselves in compatibility mode.

```

55.29 \ifx\encodingdefault\cs@iltw@
55.30 \input il2enc.def
55.31 \fi

```

Restore the definition of $\text{\textit{CurrentOption}}$, clobbered by processing the options.

```

55.32 \def\CurrentOption{slovak}
55.33 \fi

```

The next step consists of defining commands to switch to (and from) the Slovak language.

$\text{\textit{\captionsslovak}}$ The macro $\text{\textit{\captionsslovak}}$ defines all strings used in the four standard documentclasses provided with $\text{\textit{LATEX}}$.

```

55.34 \@namedef{captions\CurrentOption}{%
55.35 \def\prefacename{Predhovor}%
55.36 \def\refname{Literat\'ura}%
55.37 \def\abstractname{Abstrakt}%
55.38 \def\bibname{Literat\'ura}%
55.39 \def\chaptername{Kapitola}%
55.40 \def\appendixname{Dodatok}%
55.41 \def\contentsname{Obsah}%
55.42 \def\listfigurename{Zoznam obr\'azkov}%
55.43 \def\listtablename{Zoznam tabuliek}%
55.44 \def\indexname{Register}%
55.45 \def\figurename{Obr.}%
55.46 \def\tablename{Tabu\v{1}ka}%
55.47 \def\partname{\v{C}as\v{t}}}%

```

```

55.48 \def\enclname{Pr\'{\i}loha}%
55.49 \def\ccname{cc.}%
55.50 \def\headtoname{Pre}%
55.51 \def\pagename{Str.}%
55.52 \def\seename{vi\v{d}}%
55.53 \def\alsoname{vi\v{d} tie\v{z}}%
55.54 \def\proofname{D\'okaz}%
55.55 \def\glossaryname{Slovn\'{\i}k}%
55.56 }%

```

`\dateslovak` The macro `\dateslovak` redefines the command `\today` to produce Slovak dates.

```

55.57 \@namedef{date\CurrentOption}{%
55.58 \def\today{\number\day.\~\ifcase\month\or
55.59 janu\'ara\or febru\'ara\or marca\or apr\'{\i}la\or m\'aja\or
55.60 j\'una\or j\'ula\or augusta\or septembra\or okt\'obra\or
55.61 novembra\or decembra\fi
55.62 \space \number\year}}

```

`\extrasslovak` The macro `\extrasslovak` will perform all the extra definitions needed for the Slovak language. The macro `\noextrasslovak` is used to cancel the actions of `\extrasslovak`.

For Slovak texts `\frenchspacing` should be in effect. Language group for shorthands is also set here.

```

55.63 \expandafter\addto\csname extras\CurrentOption\endcsname{%
55.64 \bbl@frenchspacing
55.65 \languageshorthands{slovak}}
55.66 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
55.67 \bbl@nonfrenchspacing}

55.68 \expandafter\addto\csname extras\CurrentOption\endcsname{%
55.69 \babel@save\q\let\q\v}

```

For Slovak three characters are used to define shorthands, they need to be made active.

```

55.70 \ifx\cs@compat@latex\relax\else
55.71 \initiate@active@char{~}
55.72 \addto\extrasslovak{\bbl@activate{~}}
55.73 \addto\noextrasslovak{\bbl@deactivate{~}}
55.74 \initiate@active@char{"}
55.75 \addto\extrasslovak{\bbl@activate{"}\umlautlow}
55.76 \addto\noextrasslovak{\bbl@deactivate{"}\umlauthigh}
55.77 \initiate@active@char{' }
55.78 \@ifpackagewith{babel}{activeacute}{%
55.79 \addto\extrasslovak{\bbl@activate{'}}
55.80 \addto\noextrasslovak{\bbl@deactivate{'}}}%
55.81 }{}
55.82 \fi

```

`\sq` We save the original single and double quote characters in `\sq` and `\dq` to make `\dq` them available later. The math accent `\"` can now be typed as `\"`.

```

55.83 \begingroup\catcode'\=12\catcode'\'=12
55.84 \def\x{\endgroup
55.85 \def\squ{' }
55.86 \def\dqu{" }
55.87 \x

```

The slovak hyphenation patterns should be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 3.

```

55.88 \providehyphenmins{\CurrentOption}{\tw@\thr@@}

```

In order to prevent problems with the active \sim we add a shorthand on system level which expands to a ‘normal \sim ’.

```
55.89 \ifx\cs@compat@latex\relax\else
55.90 \declare@shorthand{system}{~}{\csname normal@char\string~\endcsname}
```

Now we can define the doublequote macros: the umlauts,

```
55.91 \declare@shorthand{slovak}{a}{\textormath{"a\allowhyphens}{\ddot a}}
55.92 \declare@shorthand{slovak}{o}{\textormath{"o\allowhyphens}{\ddot o}}
55.93 \declare@shorthand{slovak}{u}{\textormath{"u\allowhyphens}{\ddot u}}
55.94 \declare@shorthand{slovak}{A}{\textormath{"A\allowhyphens}{\ddot A}}
55.95 \declare@shorthand{slovak}{O}{\textormath{"O\allowhyphens}{\ddot O}}
55.96 \declare@shorthand{slovak}{U}{\textormath{"U\allowhyphens}{\ddot U}}
```

tremas,

```
55.97 \declare@shorthand{slovak}{e}{\textormath{"e\allowhyphens}{\ddot e}}
55.98 \declare@shorthand{slovak}{E}{\textormath{"E\allowhyphens}{\ddot E}}
55.99 \declare@shorthand{slovak}{i}{\textormath{"i\allowhyphens}{%
55.100 \ddot\imath}}
55.101 \declare@shorthand{slovak}{I}{\textormath{"I\allowhyphens}{\ddot I}}
```

other slovak characters

```
55.102 \declare@shorthand{slovak}{c}{\textormath{v{c}\allowhyphens}{\check{c}}}
55.103 \declare@shorthand{slovak}{d}{\textormath{q{d}\allowhyphens}{\check{d}}}
55.104 \declare@shorthand{slovak}{l}{\textormath{q{l}\allowhyphens}{\check{l}}}
55.105 \declare@shorthand{slovak}{n}{\textormath{v{n}\allowhyphens}{\check{n}}}
55.106 \declare@shorthand{slovak}{o}{\textormath{~{o}\allowhyphens}{\hat{o}}}
55.107 \declare@shorthand{slovak}{s}{\textormath{v{s}\allowhyphens}{\check{s}}}
55.108 \declare@shorthand{slovak}{t}{\textormath{q{t}\allowhyphens}{\check{t}}}
55.109 \declare@shorthand{slovak}{z}{\textormath{v{z}\allowhyphens}{\check{z}}}
55.110 \declare@shorthand{slovak}{C}{\textormath{v{C}\allowhyphens}{\check{C}}}
55.111 \declare@shorthand{slovak}{D}{\textormath{v{D}\allowhyphens}{\check{D}}}
55.112 \declare@shorthand{slovak}{L}{\textormath{q{L}\allowhyphens}{\check{L}}}
55.113 \declare@shorthand{slovak}{N}{\textormath{v{N}\allowhyphens}{\check{N}}}
55.114 \declare@shorthand{slovak}{O}{\textormath{~{O}\allowhyphens}{\hat{O}}}
55.115 \declare@shorthand{slovak}{S}{\textormath{v{S}\allowhyphens}{\check{S}}}
55.116 \declare@shorthand{slovak}{T}{\textormath{v{T}\allowhyphens}{\check{T}}}
55.117 \declare@shorthand{slovak}{Z}{\textormath{v{Z}\allowhyphens}{\check{Z}}}
```

acute accents,

```
55.118 \@ifpackagewith{babel}{activeacute}{%
55.119 \declare@shorthand{slovak}{a}{\textormath{a\allowhyphens}{~{\prime}a}}
55.120 \declare@shorthand{slovak}{e}{\textormath{e\allowhyphens}{~{\prime}e}}
55.121 \declare@shorthand{slovak}{i}{\textormath{i\allowhyphens}{~{\prime}i}}
55.122 \declare@shorthand{slovak}{l}{\textormath{l\allowhyphens}{~{\prime}l}}
55.123 \declare@shorthand{slovak}{o}{\textormath{o\allowhyphens}{~{\prime}o}}
55.124 \declare@shorthand{slovak}{r}{\textormath{r\allowhyphens}{~{\prime}r}}
55.125 \declare@shorthand{slovak}{u}{\textormath{u\allowhyphens}{~{\prime}u}}
55.126 \declare@shorthand{slovak}{y}{\textormath{y\allowhyphens}{~{\prime}y}}
55.127 \declare@shorthand{slovak}{A}{\textormath{A\allowhyphens}{~{\prime}A}}
55.128 \declare@shorthand{slovak}{E}{\textormath{E\allowhyphens}{~{\prime}E}}
55.129 \declare@shorthand{slovak}{I}{\textormath{I\allowhyphens}{~{\prime}I}}
55.130 \declare@shorthand{slovak}{L}{\textormath{L\allowhyphens}{~{\prime}l}}
55.131 \declare@shorthand{slovak}{O}{\textormath{O\allowhyphens}{~{\prime}O}}
55.132 \declare@shorthand{slovak}{R}{\textormath{R\allowhyphens}{~{\prime}R}}
55.133 \declare@shorthand{slovak}{U}{\textormath{U\allowhyphens}{~{\prime}U}}
55.134 \declare@shorthand{slovak}{Y}{\textormath{Y\allowhyphens}{~{\prime}Y}}
55.135 \declare@shorthand{slovak}{' }{\prime}%
55.136 \textormath{\textquotedblright}{\sp\bgroupprim@s'}}
55.137 }{}
55.138
```

and some additional commands:

```
55.139 \declare@shorthand{slovak}{-}{\nobreak\-\bbl@allowhyphens}
55.140 \declare@shorthand{slovak}{|}{\%
```



```

55.141 \textormath{\penalty\M\discretionary{-}{-}{\kern.03em}%
55.142 \bbl@allowhyphens}{}}
55.143 \declare@shorthand{slovak}{""}{\hskip\z@skip}
55.144 \declare@shorthand{slovak}{"}{\textormath{\leavevmode\hbox{-}}{-}}
55.145 \declare@shorthand{slovak}{"=}{\cs@splithyphen}
55.146 \fi

```

\v L^AT_EX's normal \v accent places a caron over the letter that follows it (ö). This is not what we want for the letters d, t, l and L; for those the accent should change shape. This is achieved by the following.

```

55.147 \AtBeginDocument{%
55.148 \DeclareTextCompositeCommand{\v}{OT1}{t}{t}%
55.149 t\kern-.23em\raise.24ex\hbox{'}}
55.150 \DeclareTextCompositeCommand{\v}{OT1}{d}{d}%
55.151 d\kern-.13em\raise.24ex\hbox{'}}
55.152 \DeclareTextCompositeCommand{\v}{OT1}{l}{\lcaron{}}
55.153 \DeclareTextCompositeCommand{\v}{OT1}{L}{\Lcaron{}}}

```

\lcaron For the letters l and L we want to distinguish between normal fonts and monospaced fonts.

```

\lcaron
55.154 \def\lcaron{%
55.155 \setbox0\hbox{M}\setbox\tw@\hbox{i}%
55.156 \ifdim\wd0>\wd\tw@\relax
55.157 l\kern-.13em\raise.24ex\hbox{'}\kern-.11em%
55.158 \else
55.159 l\raise.45ex\hbox to\z@{\kern-.35em '\hss}%
55.160 \fi}
55.161 \def\Lcaron{%
55.162 \setbox0\hbox{M}\setbox\tw@\hbox{i}%
55.163 \ifdim\wd0>\wd\tw@\relax
55.164 L\raise.24ex\hbox to\z@{\kern-.28em'\hss}%
55.165 \else
55.166 L\raise.45ex\hbox to\z@{\kern-.40em '\hss}%
55.167 \fi}

```

Initialize active quotes. C^SL^AT_EX provides a way of converting English-style quotes into Slovak-style ones. Both single and double quotes are affected, i.e. ‘text’ is converted to something like ,text‘ and ‘text’ is converted to ,text‘. This conversion can be switched on and off with \csprimeson and \csprimesoff.⁶⁵

These quotes present various troubles, e.g. the kerning is broken, apostrophes are converted to closing single quote, some primitives are broken (most notably the \catcode‘\⟨char⟩ syntax will not work any more), and writing them to .aux files cannot be handled correctly. For these reasons, these commands are only available in C^SL^AT_EX compatibility mode.

```

55.168 \ifx\cs@compat@latex\relax
55.169 \let\cs@ltxprim@s\prim@s
55.170 \def\csprimeson{%
55.171 \catcode‘\active \catcode‘\active \let\prim@s\bbl@prim@s}
55.172 \def\csprimesoff{%
55.173 \catcode‘12 \catcode‘12 \let\prim@s\cs@ltxprim@s}
55.174 \begingroup\catcode‘\active
55.175 \def\x{\endgroup
55.176 \def‘{\futurelet\cs@next\cs@openquote}
55.177 \def\cs@openquote{%
55.178 \ifx‘\cs@next \expandafter\cs@opendq
55.179 \else \expandafter\clq
55.180 \fi}%
55.181 }\x

```

⁶⁵By the way, the names of these macros are misleading, because the handling of primes in math mode is rather marginal, the most important thing being the handling of quotes...

```

55.182 \begingroup\catcode'\active
55.183 \def\x{\endgroup
55.184   \def'\{\textormath{\futurelet\cs@next\cs@closequote}
55.185     {\~\bgroup\prim@s}}
55.186   \def\cs@closequote{%
55.187     \ifx'\cs@next \expandafter\cs@closedq
55.188     \else \expandafter\crq
55.189     \fi}%
55.190   }\x
55.191 \def\cs@opendq{\clqq\let\cs@next= }
55.192 \def\cs@closedq{\crqq\let\cs@next= }

```

The way I recommend for typesetting quotes in Slovak documents is to use shorthands similar to those used in German.

```

55.193 \else
55.194   \declare@shorthand{slovak}{''}{\clqq}
55.195   \declare@shorthand{slovak}{'''}{\crqq}
55.196   \declare@shorthand{slovak}{"<"}{\flqq}
55.197   \declare@shorthand{slovak}{">"}{\frqq}
55.198 \fi

```

`\clqq` This is the CS opening quote, which is similar to the German quote (`\glqq`) but the kerning is different.

For the OT1 encoding, the quote is constructed from the right double quote (i.e. the “Opening quotes” character) by moving it down to the baseline and shifting it to the right, or to the left if italic correction is positive.

For T1, the “German Opening quotes” is used. It is moved to the right and the total width is enlarged. This is done in an attempt to minimize the difference between the OT1 and T1 versions.

```

55.199 \ProvideTextCommand{\clqq}{OT1}{%
55.200   \set@low@box{\textquotedblright}%
55.201   \setbox\@ne=\hbox{1/}\dimen\@ne=\wd\@ne
55.202   \setbox\@ne=\hbox{1}\advance\dimen\@ne-\wd\@ne
55.203   \leavevmode
55.204   \ifdim\dimen\@ne>z@\kern-.1em\box\z@\kern.1em
55.205   \else\kern.1em\box\z@\kern-.1em\fi\allowhyphens}
55.206 \ProvideTextCommand{\clqq}{T1}
55.207   {\kern.1em\quotedblbase\kern-.0158em\relax}
55.208 \ProvideTextCommandDefault{\clqq}{\UseTextSymbol{OT1}\clqq}

```

`\crqq` For OT1, the CS closing quote is basically the same as `\grqq`, only the `\textormath` macro is not used, because as far as I know, `\grqq` does not work in math mode anyway.

For T1, the character is slightly wider and shifted to the right to match its OT1 counterpart.

```

55.209 \ProvideTextCommand{\crqq}{OT1}
55.210   {\save@sf@q{\nobreak\kern-.07em\textquotedblleft\kern.07em}}
55.211 \ProvideTextCommand{\crqq}{T1}
55.212   {\save@sf@q{\nobreak\kern.06em\textquotedblleft\kern.024em}}
55.213 \ProvideTextCommandDefault{\crqq}{\UseTextSymbol{OT1}\crqq}

```

`\clq` Single CS quotes are similar to double quotes (see the discussion above).

```

\crq55.214 \ProvideTextCommand{\clq}{OT1}
55.215   {\set@low@box{\textquoteright}\box\z@\kern.04em\allowhyphens}
55.216 \ProvideTextCommand{\clq}{T1}
55.217   {\quotesinglbase\kern-.0428em\relax}
55.218 \ProvideTextCommandDefault{\clq}{\UseTextSymbol{OT1}\clq}
55.219 \ProvideTextCommand{\crq}{OT1}
55.220   {\save@sf@q{\nobreak\textquoteleft\kern.17em}}
55.221 \ProvideTextCommand{\crq}{T1}
55.222   {\save@sf@q{\nobreak\textquoteleft\kern.17em}}
55.223 \ProvideTextCommandDefault{\crq}{\UseTextSymbol{OT1}\crq}

```

`\uv` There are two versions of `\uv`. The older one opens a group and uses `\aftergroup` to typeset the closing quotes. This version allows using `\verb` inside the quotes, because the enclosed text is not passed as an argument, but unfortunately it breaks any kerning with the quotes. Although the kerning with the opening quote could be fixed, the kerning with the closing quote cannot.

The newer version is defined as a command with one parameter. It preserves kerning but since the quoted text is passed as an argument, it cannot contain `\verb`.

Decide which version of `\uv` should be used. For sake of compatibility, we use the older version with plain $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ and the newer version with $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2_{\epsilon}$.

```

55.224 \ifx\cs@compat@plain\@undefined\else\let\cs@olduv=\relax\fi
55.225 \ifx\cs@olduv\@undefined
55.226   \DeclareRobustCommand\uv[1]{\leavevmode\clqq#1\crqq}
55.227 \else
55.228   \DeclareRobustCommand\uv{\bgroup\aftergroup\closequotes
55.229     \leavevmode\clqq\let\cs@next=}
55.230   \def\closequotes{\unskip\crqq\relax}
55.231 \fi

```

`\cs@wordlen` Declare a counter to hold the length of the word after the hyphen.

```

55.232 \newcount\cs@wordlen

```

`\cs@hyphen` Store the original hyphen in a macro. Ditto for the ligatures.

```

\cs@endash55.233 \begingroup\catcode'\-12
\cs@emdash55.234 \def\x{\endgroup
55.235   \def\cs@hyphen{-}
55.236   \def\cs@endash{--}
55.237   \def\cs@emdash{---}

```

`\cs@boxhyphen` Provide a non-breakable hyphen to be used when a compound word is too short to be split, i.e. the second part is shorter than `\righthyphenmin`.

```

55.238 \def\cs@boxhyphen{\hbox{-}}

```

`\cs@splithyphen` The macro `\cs@splithyphen` inserts a split hyphen, while allowing both parts of the compound word to be hyphenated at other places too.

```

55.239 \def\cs@splithyphen{\kern\z@
55.240   \discretionary{-}{\char\hyphenchar\the\font}{-}\nobreak\hskip\z@}
55.241 }\x

```

- To minimize the effects of activating the hyphen character, the active definition expands to the non-active character in all cases where hyphenation cannot occur, i.e. if not typesetting (check `\protect`), not in horizontal mode, or in inner horizontal mode.

```

55.242 \initiate@active@char{-}
55.243 \declare@shorthand{slovak}{-}{%
55.244   \ifx\protect\@typeset@protect
55.245     \ifhmode
55.246       \ifinner
55.247         \bbl@afterelse\bbl@afterelse\bbl@afterelse\cs@hyphen
55.248       \else
55.249         \bbl@afterfi\bbl@afterelse\bbl@afterelse\cs@firsthyphen
55.250       \fi
55.251     \else
55.252       \bbl@afterfi\bbl@afterelse\cs@hyphen
55.253     \fi
55.254   \else
55.255     \bbl@afterfi\cs@hyphen
55.256   \fi}

```

`\cs@firsthyphen` If we encounter a hyphen, check whether it is followed by a second or a third
`\cs@firsthyph@n` hyphen and if so, insert the corresponding ligature.
`\cs@secondhyphen` If we don't find a hyphen, the token found will be placed in `\cs@token` for
`\cs@secondhyph@n` further analysis, and it will also stay in the input.

```

55.257 \begingroup\catcode'\- \active
55.258 \def\x{\endgroup
55.259 \def\cs@firsthyphen{\futurelet\cs@token\cs@firsthyph@n}
55.260 \def\cs@firsthyph@n{%
55.261 \ifx -\cs@token
55.262 \bbl@afterelse\cs@secondhyphen
55.263 \else
55.264 \bbl@afterfi\cs@checkhyphen
55.265 \fi}
55.266 \def\cs@secondhyphen##1{%
55.267 \futurelet\cs@token\cs@secondhyph@n}
55.268 \def\cs@secondhyph@n{%
55.269 \ifx -\cs@token
55.270 \bbl@afterelse\cs@emdash@gobble
55.271 \else
55.272 \bbl@afterfi\cs@endash
55.273 \fi}
55.274 }\x

```

`\cs@checkhyphen` Check that hyphenation is enabled, and if so, start analyzing the rest of the word, i.e. initialize `\cs@word` and `\cs@wordlen` and start processing input with `\cs@scanword`.

```

55.275 \def\cs@checkhyphen{%
55.276 \ifnum\expandafter\hyphenchar\the\font='-
55.277 \def\cs@word{}\cs@wordlen\z@
55.278 \bbl@afterelse\cs@scanword
55.279 \else
55.280 \cs@hyphen
55.281 \fi}

```

`\cs@scanword` Each token is first analyzed with `\cs@scanword`, which expands the token and
`\cs@continuescan` passes the first token of the result to `\cs@gett@ken`. If the expanded token is not
`\cs@gett@ken` identical to the unexpanded one, presume that it might be expanded further and
`\cs@gett@ken` pass it back to `\cs@scanword` until you get an unexpandable token. Then analyze
it in `\cs@examinetoken`.

The `\cs@continuescan` macro does the same thing as `\cs@scanword`, but it does not require the first token to be in `\cs@token` already.

```

55.282 \def\cs@scanword{\let\cs@lasttoken=\cs@token\expandafter\cs@gett@ken}
55.283 \def\cs@continuescan{\let\cs@lasttoken\undefined\expandafter\cs@gett@ken}
55.284 \def\cs@gett@ken{\futurelet\cs@token\cs@gett@ken}
55.285 \def\cs@gett@ken{%
55.286 \ifx\cs@token\cs@lasttoken \def\cs@next{\cs@examinetoken}%
55.287 \else \def\cs@next{\cs@scanword}%
55.288 \fi \cs@next}

```

`\cs@examinetoken` Examine the token in `\cs@token`:

- If it is a letter (catcode 11) or other (catcode 12), add it to `\cs@word` with `\cs@addparam`.
- If it is the `\char` primitive, add it with `\cs@expandchar`.
- If the token starts or ends a group, ignore it with `\cs@ignoretoken`.
- Otherwise analyze the meaning of the token with `\cs@checkchardef` to detect primitives defined with `\chardef`.

```

55.289 \def\cs@examinetoken{%
55.290   \ifcat A\cs@token
55.291     \def\cs@next{\cs@addparam}%
55.292   \else\ifcat O\cs@token
55.293     \def\cs@next{\cs@addparam}%
55.294   \else\ifx\char\cs@token
55.295     \def\cs@next{\afterassignment\cs@expandchar\let\cs@token= }%
55.296   \else\ifx\bgroup\cs@token
55.297     \def\cs@next{\cs@ignoretoken\bgroup}%
55.298   \else\ifx\egroup\cs@token
55.299     \def\cs@next{\cs@ignoretoken\egroup}%
55.300   \else\ifx\begingroup\cs@token
55.301     \def\cs@next{\cs@ignoretoken\begingroup}%
55.302   \else\ifx\endgroup\cs@token
55.303     \def\cs@next{\cs@ignoretoken\endgroup}%
55.304   \else
55.305     \def\cs@next{\expandafter\expandafter\expandafter\cs@checkchardef
55.306       \expandafter\meaning\expandafter\cs@token\string\char\end}%
55.307   \fi\fi\fi\fi\fi\fi\cs@next}

```

\cs@checkchardef Check the meaning of a token and if it is a primitive defined with `\chardef`, pass it to `\\@examinechar` as if it were a `\char` sequence. Otherwise, there are no more word characters, so do the final actions in `\cs@nosplit`.

```

55.308 \expandafter\def\expandafter\cs@checkchardef
55.309   \expandafter#\expandafter1\string\char#2\end{%
55.310   \def\cs@token{#1}%
55.311   \ifx\cs@token@empty
55.312     \def\cs@next{\afterassignment\cs@examinechar\let\cs@token= }%
55.313   \else
55.314     \def\cs@next{\cs@nosplit}%
55.315   \fi \cs@next}

```

\cs@ignoretoken Add a token to `\cs@word` but do not update the `\cs@wordlen` counter. This is mainly useful for group starting and ending primitives, which need to be preserved, but do not affect the word boundary.

```

55.316 \def\cs@ignoretoken#1{%
55.317   \edef\cs@word{\cs@word#1}%
55.318   \afterassignment\cs@continuescan\let\cs@token= }

```

cs@addparam Add a token to `\cs@word` and check its lcode. Note that this macro can only be used for tokens which can be passed as a parameter.

```

55.319 \def\cs@addparam#1{%
55.320   \edef\cs@word{\cs@word#1}%
55.321   \cs@checkcode{\lcode'#1}}

```

\cs@expandchar Add a `\char` sequence to `\cs@word` and check its lcode. The charcode is first parsed in `\cs@expandchar` and then the resulting `\chardef`-defined sequence is analyzed in `\cs@examinechar`.

```

55.322 \def\cs@expandchar{\afterassignment\cs@examinechar\chardef\cs@token=}
55.323 \def\cs@examinechar{%
55.324   \edef\cs@word{\cs@word\char\the\cs@token\space}%
55.325   \cs@checkcode{\lcode\cs@token}}

```

\cs@checkcode Check the lcode of a character. If it is zero, it does not count to the current word, so finish it with `\cs@nosplit`. Otherwise update the `\cs@wordlen` counter and go on scanning the word with `\cs@continuescan`. When enough characters are gathered in `\cs@word` to allow word break, insert the split hyphen and finish.

```

55.326 \def\cs@checkcode#1{%
55.327   \ifnum0=#1
55.328     \def\cs@next{\cs@nosplit}%

```

```

55.329 \else
55.330   \advance\cs@wordlen\@ne
55.331   \ifnum\righthyphenmin>\the\cs@wordlen
55.332     \def\cs@next{\cs@continuescan}%
55.333   \else
55.334     \cs@splithyphen
55.335     \def\cs@next{\cs@word}%
55.336   \fi
55.337 \fi \cs@next}

```

`\cs@nosplit` Insert a non-breakable hyphen followed by the saved word.

```
55.338 \def\cs@nosplit{\cs@boxhyphen\cs@word}
```

`\cs@hyphen` The `\minus` sequence can be used where the active hyphen does not work, e.g. in arguments to \TeX primitives in outer horizontal mode.

```
55.339 \let\minus\cs@hyphen
```

`\standardhyphens` These macros control whether split hyphens are allowed in Czech and/or Slovak texts. You may use them in any language, but the split hyphen is only activated for Czech and Slovak.

```

55.340 \def\standardhyphens{\cs@splithyphensfalse\cs@deactivatehyphens}
55.341 \def\cs@splithyphens{\cs@splithyphenstrue\cs@activatehyphens}

```

`\cs@splitattr` Now we declare the `split` language attribute. This is similar to the `split` package option of `cslatex`, but it only affects Slovak, not Czech.

```

55.342 \def\cs@splitattr{\babel@save\ifcs@splithyphens\splithyphens}
55.343 \bbl@declare@ttribute{slovak}{split}{%
55.344   \addto\extrasslovak{\cs@splitattr}}

```

`\cs@activatehyphens` These macros are defined as `\relax` by default to prevent activating/deactivating the hyphen character. They are redefined when the language is switched to Czech/Slovak. At that moment the hyphen is also activated if split hyphens were requested with `\splithyphens`.

When the language is de-activated, de-activate the hyphen and restore the bogus definitions of these macros.

```

55.345 \let\cs@activatehyphens\relax
55.346 \let\cs@deactivatehyphens\relax
55.347 \expandafter\addto\csname extras\CurrentOption\endcsname{%
55.348   \def\cs@activatehyphens{\bbl@activate{-}}%
55.349   \def\cs@deactivatehyphens{\bbl@deactivate{-}}%
55.350   \ifcs@splithyphens\cs@activatehyphens\fi}
55.351 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
55.352   \cs@deactivatehyphens
55.353   \let\cs@activatehyphens\relax
55.354   \let\cs@deactivatehyphens\relax}

```

`\cs@looseness` One of the most common situations where an active hyphen will not work properly is the `\looseness` primitive. Change its definition so that it deactivates the hyphen if needed.

```

55.355 \let\cs@looseness\looseness
55.356 \def\looseness{%
55.357   \ifcs@splithyphens
55.358     \cs@deactivatehyphens\afterassignment\cs@activatehyphens \fi
55.359   \cs@looseness}

```

`\cs@selectlanguage` Specifying the `nocaptions` option means that captions and dates are not redefined by default, but they can be switched on later with `\captionsslovak` and/or `\dateslovak`.

We mimic this behavior by redefining `\selectlanguage`. This macro is called once at the beginning of the document to set the main language of the document.

If this is `\cs@main@language`, it disables the macros for setting captions and date. In any case, it restores the original definition of `\selectlanguage` and expands it.

The definition of `\selectlanguage` can be shared between Czech and Slovak; the actual language is stored in `\cs@main@language`.

```

55.360 \ifx\cs@nocaptions\@undefined\else
55.361   \edef\cs@main@language{\CurrentOption}
55.362   \ifx\cs@origselect\@undefined
55.363     \let\cs@origselect=\selectlanguage
55.364     \def\selectlanguage{%
55.365       \let\selectlanguage\cs@origselect
55.366       \ifx\bbl@main@language\cs@main@language
55.367         \expandafter\cs@selectlanguage
55.368       \else
55.369         \expandafter\selectlanguage
55.370       \fi}
55.371   \def\cs@selectlanguage{%
55.372     \cs@tempdisable{captions}%
55.373     \cs@tempdisable{date}%
55.374     \selectlanguage}

```

`\cs@tempdisable` `\cs@tempdisable` disables a language setup macro temporarily, i.e. the macro with the name of `<#1>\bbl@main@language` just restores the original definition and purges the saved macro from memory.

```

55.375   \def\cs@tempdisable#1{%
55.376     \def\@tempa{cs@#1}%
55.377     \def\@tempb{#1\bbl@main@language}%
55.378     \expandafter\expandafter\expandafter\let
55.379       \expandafter \csname\expandafter \@tempa \expandafter\endcsname
55.380       \csname \@tempb \endcsname
55.381     \expandafter\edef\csname \@tempb \endcsname{%
55.382       \let \expandafter\noexpand \csname \@tempb \endcsname
55.383       \expandafter\noexpand \csname \@tempa \endcsname
55.384       \let \expandafter\noexpand\csname \@tempa \endcsname
55.385       \noexpand\@undefined}}

```

These macros are not needed, once the initialization is over.

```

55.386   \@onlypreamble\cs@main@language
55.387   \@onlypreamble\cs@origselect
55.388   \@onlypreamble\cs@selectlanguage
55.389   \@onlypreamble\cs@tempdisable
55.390 \fi
55.391 \fi

```

The encoding of mathematical fonts should be changed to IL2. This allows to use accented letter in some font families. Besides, documents do not use CM fonts if there are equivalents in CS-fonts, so there is no need to have both bitmaps of CM-font and CS-font.

`\@font@warning` and `\@font@info` are temporarily redefined to avoid annoying font warnings.

```

55.392 \ifx\cs@compat@plain\@undefined
55.393 \ifx\cs@check@enc\@undefined\else
55.394   \def\cs@check@enc{
55.395     \ifx\encodingdefault\cs@iltw@
55.396       \let\cs@warn\@font@warning \let\@font@warning\@gobble
55.397       \let\cs@info\@font@info \let\@font@info\@gobble
55.398       \SetSymbolFont{operators}{normal}{\cs@iltw@}{cmr}{m}{n}
55.399       \SetSymbolFont{operators}{bold}{\cs@iltw@}{cmr}{bx}{n}
55.400       \SetMathAlphabet\mathbf{normal}{\cs@iltw@}{cmr}{bx}{n}
55.401       \SetMathAlphabet\mathit{normal}{\cs@iltw@}{cmr}{m}{it}

```

```

55.402 \SetMathAlphabet\mathrm{normal}{\cs@iltw@}{cmr}{m}{n}
55.403 \SetMathAlphabet\mathsf{normal}{\cs@iltw@}{cmss}{m}{n}
55.404 \SetMathAlphabet\mathtt{normal}{\cs@iltw@}{cmtt}{m}{n}
55.405 \SetMathAlphabet\mathbf{bold}{\cs@iltw@}{cmr}{bx}{n}
55.406 \SetMathAlphabet\mathit{bold}{\cs@iltw@}{cmr}{bx}{it}
55.407 \SetMathAlphabet\mathrm{bold}{\cs@iltw@}{cmr}{bx}{n}
55.408 \SetMathAlphabet\mathsf{bold}{\cs@iltw@}{cmss}{bx}{n}
55.409 \SetMathAlphabet\mathtt{bold}{\cs@iltw@}{cmtt}{m}{n}
55.410 \let\@font@warning\cs@warn \let\cs@warn\@undefined
55.411 \let\@font@info\cs@info \let\cs@info\@undefined
55.412 \fi
55.413 \let\cs@check@enc\@undefined}
55.414 \AtBeginDocument{\cs@check@enc}
55.415 \fi
55.416 \fi

```

`cs@undoiltw@` The thing is that L^AT_EX 2_ε core only supports the T1 encoding and does not bother changing the uc/lc/sf codes when encoding is switched. :(However, the IL2 encoding *does* change these codes, so if encoding is switched back from IL2, we must also undo the effect of this change to be compatible with L^AT_EX 2_ε. OK, this is not the rightTM solution but it works. Cheers to Petr Olšák.

```

55.417 \def\cs@undoiltw@{%
55.418 \uccode158=208 \lccode158=158 \sfcode158=1000
55.419 \sfcode159=1000
55.420 \uccode165=133 \lccode165=165 \sfcode165=1000
55.421 \uccode169=137 \lccode169=169 \sfcode169=1000
55.422 \uccode171=139 \lccode171=171 \sfcode171=1000
55.423 \uccode174=142 \lccode174=174 \sfcode174=1000
55.424 \uccode181=149
55.425 \uccode185=153
55.426 \uccode187=155
55.427 \uccode190=0 \lccode190=0
55.428 \uccode254=222 \lccode254=254 \sfcode254=1000
55.429 \uccode255=223 \lccode255=255 \sfcode255=1000}

```

`@@enc@update` Redefine the L^AT_EX 2_ε internal function `\@@enc@update` to change the encodings correctly.

```

55.430 \ifx\cs@enc@update\@undefined
55.431 \ifx\@@enc@update\@undefined\else
55.432 \let\cs@enc@update\@@enc@update
55.433 \def\@@enc@update{\ifx\cf@encoding\cs@iltw@\cs@undoiltw@\fi
55.434 \cs@enc@update
55.435 \expandafter\ifnum\csname l@language\endcsname=\the\language
55.436 \expandafter\ifx
55.437 \csname l@language:\f@encoding\endcsname\relax
55.438 \else
55.439 \expandafter\expandafter\expandafter\let
55.440 \expandafter\csname
55.441 \expandafter l\expandafter @\expandafter\language
55.442 \expandafter\endcsname\csname l@language:\f@encoding\endcsname
55.443 \fi
55.444 \language=\csname l@language\endcsname\relax
55.445 \fi}
55.446 \fi\fi

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

55.447 \ldf@finish\CurrentOption
55.448 </code>

```


56 The Slovenian language

The file `slovene.dtx`⁶⁶ defines all the language-specific macros for the Slovenian language.

For this language the character " is made active. In table 29 an overview is given of its purpose. One of the reasons for this is that in the Slovene language some special characters are used.

"c	\c, also implemented for the lowercase and uppercase s and z.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
""	like "-", but producing no hyphen sign (for compound words with hyphen, e.g. x-"y).
"‘	for Slovene left double quotes (looks like „).
"’	for Slovene right double quotes.
"<	for French left double quotes (similar to <<).
">	for French right double quotes (similar to >>).

Table 29: The extra definitions made by `slovene.ldf`

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
56.1 (*code)
56.2 \LdfInit{slovene}\captionsslovene
```

When this file is read as an option, i.e. by the `\usepackage` command, `slovene` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@slovene` to see whether we have to do something here.

```
56.3 \ifx\l@slovene\@undefined
56.4     \nopatterns{Slovene}
56.5     \adddialect\l@slovene0\fi
```

The next step consists of defining commands to switch to the Slovenian language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsslovene` The macro `\captionsslovene` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
56.6 \addto\captionsslovene{%
56.7   \def\prefacename{Predgovor}%
56.8   \def\refname{Literatura}%
56.9   \def\abstractname{Povzetek}%
56.10  \def\bibname{Literatura}%
56.11  \def\chaptername{Poglavje}%
56.12  \def\appendixname{Dodatek}%
56.13  \def\contentsname{Kazalo}%
56.14  \def\listfigurename{Slike}%
56.15  \def\listtablename{Tabele}%
56.16  \def\indexname{Stvarno kazalo}% used to be Indeks
56.17  \def\figurename{Slika}%
56.18  \def\tablename{Tabela}%
56.19  \def\partname{Del}%
56.20  \def\enclname{Priloge}%
56.21  \def\ccname{Kopije}%
56.22  \def\headtoname{Prejme}%
```

⁶⁶The file described in this section has version number v1.2m and was last revised on 2005/03/31. Contributions were made by Danilo Zavrtanik, University of Ljubljana (YU) and Leon Žlajpah (leon.zlajpah@ijs.si).

```

56.23 \def\pagename{Stran}%
56.24 \def\seename{glej}%
56.25 \def\alsoname{glej tudi}%
56.26 \def\proofname{Dokaz}%
56.27 \def\glossaryname{Glossary}% <-- Needs translation
56.28 }%

```

`\dateslovene` The macro `\dateslovene` redefines the command `\today` to produce Slovenian dates.

```

56.29 \def\dateslovene{%
56.30   \def\today{\number\day.~\ifcase\month\or
56.31     januar\or februar\or marec\or april\or maj\or junij\or
56.32     julij\or avgust\or september\or oktober\or november\or december\fi
56.33     \space \number\year}}

```

`\extrasslovene` The macro `\extrasslovene` performs all the extra definitions needed for the Slovenian language. The macro `\noextrasslovene` is used to cancel the actions of `\extrasslovene`.

For Slovene the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the slovenian group of shorthands should be used.

```

56.34 \initiate@active@char{"}
56.35 \addto\extrasslovene{\languageshorthands{slovene}}
56.36 \addto\extrasslovene{\bbl@activate{}}

```

Don't forget to turn the shorthands off again.

```

56.37 \addto\noextrasslovene{\bbl@deactivate{}}

```

First we define shorthands to facilitate the occurrence of letters such as č.

```

56.38 \declare@shorthand{slovene}{"c"}{\textormath{\v c}{\check c}}
56.39 \declare@shorthand{slovene}{"s"}{\textormath{\v s}{\check s}}
56.40 \declare@shorthand{slovene}{"z"}{\textormath{\v z}{\check z}}
56.41 \declare@shorthand{slovene}{"C"}{\textormath{\v C}{\check C}}
56.42 \declare@shorthand{slovene}{"S"}{\textormath{\v S}{\check S}}
56.43 \declare@shorthand{slovene}{"Z"}{\textormath{\v Z}{\check Z}}

```

Then we define access to two forms of quotation marks, similar to the german and french quotation marks.

```

56.44 \declare@shorthand{slovene}{"'"}{%
56.45   \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
56.46 \declare@shorthand{slovene}{"'"}{%
56.47   \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
56.48 \declare@shorthand{slovene}{"<"}{%
56.49   \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
56.50 \declare@shorthand{slovene}{">"}{%
56.51   \textormath{\guillemotright}{\mbox{\guillemotright}}}

```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```

56.52 \declare@shorthand{slovene}{"-"}{\nobreak-\bbl@allowhyphens}
56.53 \declare@shorthand{slovene}{"-"}{\hskip\z@skip}

```

And we want to have a shorthand for disabling a ligature.

```

56.54 \declare@shorthand{slovene}{"|"}{%
56.55   \textormath{\discretionary{-}{-}{\kern.03em}}{}}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of @ to its original value.

```

56.56 \ldf@finish{slovene}
56.57 </code>

```

57 The Russian language

The file `russianb.dtx`⁶⁷ defines all the language-specific macros for the Russian language. It needs the file `cyrnod` for success documentation with Russian encodings (see below).

For this language the character " is made active. In table 30 an overview is given of its purpose.

"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
"---	Cyrillic emdash in plain text.
"--~	Cyrillic emdash in compound names (surnames).
"--*	Cyrillic emdash for denoting direct speech.
"	like "-", but producing no hyphen sign (for compound words with hyphen, e.g. x-"y or some other signs as “disable/enable”).
"~	for a compound word mark without a breakpoint.
"=	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
",	thinspace for initials with a breakpoint in following surname.
"‘	for German left double quotes (looks like „).
"’	for German right double quotes (looks like “).
"<	for French left double quotes (looks like <<).
">	for French right double quotes (looks like >>).

Table 30: The extra definitions made by `russianb`

The quotes in table 30 can also be typeset by using the commands in table 31.

<code>\cdash---</code>	Cyrillic emdash in plain text.
<code>\cdash--~</code>	Cyrillic emdash in compound names (surnames).
<code>\cdash--*</code>	Cyrillic emdash for denoting direct speech.
<code>\glqq</code>	for German left double quotes (looks like „).
<code>\grqq</code>	for German right double quotes (looks like “).
<code>\flqq</code>	for French left double quotes (looks like <<).
<code>\frqq</code>	for French right double quotes (looks like >>).
<code>\dq</code>	the original quotes character (").

Table 31: More commands which produce quotes, defined by `babel`

The French quotes are also available as ligatures ‘<<’ and ‘>>’ in 8-bit Cyrillic font encodings (LCY, X2, T2*) and as ‘<’ and ‘>’ characters in 7-bit Cyrillic font encodings (OT2 and LWN).

The quotation marks traditionally used in Russian were borrowed from other languages (e.g., French and German) so they keep their original names.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

57.1 (*code)

57.2 `\LdfInit{russian}{captionsrussian}`

When this file is read as an option, i.e., by the `\usepackage` command, `russianb` will be an ‘unknown’ language, in which case we have to make it known.

⁶⁷The file described in this section has version number ? and was last revised on ?. This file was initially derived from the original version of `german.sty`, which has some definitions for Russian. Later the definitions from `russian.sty` version 1.0b (for L^AT_EX 2.09), `russian.sty` version v2.5c (for L^AT_EX 2_ε) and `francais.sty` version 4.5c and `germanb.sty` version 2.5c were added.

So we check for the existence of `\l@russian` to see whether we have to do something here.

```
57.3 \ifx\l@russian\@undefined
57.4   \@nopatterns{Russian}
57.5   \addialect\l@russian0
57.6 \fi
```

`\latinencoding` We need to know the encoding for text that is supposed to be which is active at the end of the `babel` package. If the `fontenc` package is loaded later, then... too bad!

```
57.7 \let\latinencoding\cf@encoding
```

The user may choose between different available Cyrillic encodings—e.g., `X2`, `LCY`, or `LWN`. Hopefully, `X2` will eventually replace the two latter encodings (`LCY` and `LWN`). If the user wants to use another font encoding than the default (`T2A`), he has to load the corresponding file *before* `russianb.sty`. This may be done in the following way:

```
% override the default X2 encoding used in Babel
\usepackage[LCY,OT1]{fontenc}
\usepackage[english,russian]{babel}
```

Note: for the Russian language, the `T2A` encoding is better than `X2`, because `X2` does not contain Latin letters, and users should be very careful to switch the language every time they want to typeset a Latin word inside a Russian phrase or vice versa.

We parse the `\cdp@list` containing the encodings known to \LaTeX in the order they were loaded. We set the `\cyrillicencoding` to the *last* loaded encoding in the list of supported Cyrillic encodings: `OT2`, `LWN`, `LCY`, `X2`, `T2C`, `T2B`, `T2A`, if any.

```
57.8 \def\reserved@a#1#2{%
57.9   \edef\reserved@b{#1}%
57.10  \edef\reserved@c{#2}%
57.11  \ifx\reserved@b\reserved@c
57.12    \let\cyrillicencoding\reserved@c
57.13  \fi}
57.14 \def\cdp@elt#1#2#3#4{%
57.15   \reserved@a{#1}{OT2}%
57.16   \reserved@a{#1}{LWN}%
57.17   \reserved@a{#1}{LCY}%
57.18   \reserved@a{#1}{X2}%
57.19   \reserved@a{#1}{T2C}%
57.20   \reserved@a{#1}{T2B}%
57.21   \reserved@a{#1}{T2A}}
57.22 \cdp@list
```

Now, if `\cyrillicencoding` is undefined, then the user did not load any of supported encodings. So, we have to set `\cyrillicencoding` to some default value. We test the presence of the encoding definition files in the order from less preferable to more preferable encodings. We use the lowercase names (i.e., `lcyenc.def` instead of `LCYenc.def`).

```
57.23 \ifx\cyrillicencoding\undefined
57.24   \IfFileExists{ot2enc.def}{\def\cyrillicencoding{OT2}}\relax
57.25   \IfFileExists{lwnenc.def}{\def\cyrillicencoding{LWN}}\relax
57.26   \IfFileExists{lcyenc.def}{\def\cyrillicencoding{LCY}}\relax
57.27   \IfFileExists{x2enc.def}{\def\cyrillicencoding{X2}}\relax
57.28   \IfFileExists{t2cenc.def}{\def\cyrillicencoding{T2C}}\relax
57.29   \IfFileExists{t2benc.def}{\def\cyrillicencoding{T2B}}\relax
57.30   \IfFileExists{t2aenc.def}{\def\cyrillicencoding{T2A}}\relax
```

If `\cyrillicencoding` is still undefined, then the user seems not to have a properly installed distribution. A fatal error.

```

57.31 \ifx\cyrillicencoding\undefined
57.32   \PackageError{babel}%
57.33     {No Cyrillic encoding definition files were found}%
57.34     {Your installation is incomplete.\MessageBreak
57.35       You need at least one of the following files:\MessageBreak
57.36       \space\space
57.37       x2enc.def, t2aenc.def, t2benc.def, t2cenc.def,\MessageBreak
57.38       \space\space
57.39       lcyenc.def, lwnenc.def, ot2enc.def.}%
57.40 \else

```

We avoid `\usepackage[\cyrillicencoding]{fontenc}` because we don't want to force the switch of `\encodingdefault`.

```

57.41   \lowercase
57.42   \expandafter{\expandafter\input\cyrillicencoding enc.def\relax}%
57.43 \fi
57.44 \fi

```

```

\PackageInfo{babel}
{Using ‘\cyrillicencoding’ as a default Cyrillic encoding}%

```

```

57.45 \DeclareRobustCommand{\Russian}{%
57.46   \fontencoding\cyrillicencoding\selectfont
57.47   \let\encodingdefault\cyrillicencoding
57.48   \expandafter\set@hyphenmins\russianhyphenmins
57.49   \language\l@russian}%
57.50 \DeclareRobustCommand{\English}{%
57.51   \fontencoding\latinencoding\selectfont
57.52   \let\encodingdefault\latinencoding
57.53   \expandafter\set@hyphenmins\englishhyphenmins
57.54   \language\l@english}%
57.55 \let\Rus\Russian
57.56 \let\Eng\English
57.57 \let\cyrillictext\Russian
57.58 \let\cyr\Russian

```

Since the X2 encoding does not contain Latin letters, we should make some redefinitions of L^AT_EX macros which implicitly produce Latin letters.

```

57.59 \expandafter\ifx\csname T@X2\endcsname\relax\else

```

We put `\latinencoding` in braces to avoid problems with `\@alph` inside minipages (e.g., footnotes inside minipages) where `\@alph` is expanded and we get for example ‘`\fontencoding OT1`’ (`\fontencoding` is robust).

```

57.60 \def\@alph#1{\fontencoding{\latinencoding}\selectfont
57.61   \ifcase#1\or
57.62     a\or b\or c\or d\or e\or f\or g\or h\or
57.63     i\or j\or k\or l\or m\or n\or o\or p\or
57.64     q\or r\or s\or t\or u\or v\or w\or x\or
57.65     y\or z\else@ctrerr\fi}%
57.66 \def\@Alph#1{\fontencoding{\latinencoding}\selectfont
57.67   \ifcase#1\or
57.68     A\or B\or C\or D\or E\or F\or G\or H\or
57.69     I\or J\or K\or L\or M\or N\or O\or P\or
57.70     Q\or R\or S\or T\or U\or V\or W\or X\or
57.71     Y\or Z\else@ctrerr\fi}%

```

Unfortunately, the commands `\AA` and `\aa` are not encoding dependent in L^AT_EX (unlike e.g., `\oe` or `\DH`). They are defined as `\r{A}` and `\r{a}`. This leads to unpredictable results when the font encoding does not contain the Latin letters ‘A’ and ‘a’ (like X2).

```

57.72 \DeclareTextSymbolDefault{\AA}{OT1}
57.73 \DeclareTextSymbolDefault{\aa}{OT1}
57.74 \DeclareTextCommand{\aa}{OT1}{\r a}
57.75 \DeclareTextCommand{\AA}{OT1}{\r A}
57.76 \fi

```

The following block redefines the character class of uppercase Greek letters and some accents, if it is equal to 7 (variable family), to avoid incorrect results if the font encoding in some math family does not contain these characters in places of OT1 encoding. The code was taken from `amsmath.dtx`. See comments and further explanation there.

```

57.77 % \begingroup\catcode'\=12
57.78 % % uppercase greek letters:
57.79 % \def\@tempa#1{\expandafter\@tempb\meaning#1\relax\relax\relax\relax
57.80 % "0000\@nil#1}
57.81 % \def\@tempb#1"#2#3#4#5#6\@nil#7{%
57.82 % \ifnum"#2=7 \count@"1#3#4#5\relax
57.83 % \ifnum\count@<"1000 \else \global\mathchardef#7="0#3#4#5\relax \fi
57.84 % \fi}
57.85 % \@tempa\Gamma\@tempa\Delta\@tempa\Theta\@tempa\Lambda\@tempa\Xi
57.86 % \@tempa\Pi\@tempa\Sigma\@tempa\Upsilon\@tempa\Phi\@tempa\Psi
57.87 % \@tempa\Omega
57.88 % % some accents:
57.89 % \def\@tempa#1#2\@nil{\def\@tempc{#1}}\def\@tempb{\mathaccent}
57.90 % \expandafter\@tempa\hat\relax\relax\@nil
57.91 % \ifx\@tempb\@tempc
57.92 % \def\@tempa#1\@nil{#1}%
57.93 % \def\@tempb#1{\afterassignment\@tempa\mathchardef\@tempc}%
57.94 % \def\do#1"#2{
57.95 % \def\@tempd#1{\expandafter\@tempb#1\@nil
57.96 % \ifnum\@tempc>"FFF
57.97 % \xdef#1{\mathaccent"\expandafter\do\meaning\@tempc\space}%
57.98 % \fi}
57.99 % \@tempd\hat\@tempd\check\@tempd\tilde\@tempd\acute\@tempd\grave
57.100 % \@tempd\dot\@tempd\ddot\@tempd\breve\@tempd\bar
57.101 % \fi
57.102 % \endgroup

```

The user should use the `inputenc` package when any 8-bit Cyrillic font encoding is used, selecting one of the Cyrillic input encodings. We do not assume any default input encoding, so the user should explicitly call the `inputenc` package by `\usepackage{inputenc}`. We also removed `\AtBeginDocument`, so `inputenc` should be used before `babel`.

```

57.103 \@ifpackageloaded{inputenc}{}{%
57.104 \def\reserved@a{LWN}%
57.105 \ifx\reserved@a\cyrillicencoding\else
57.106 \def\reserved@a{OT2}%
57.107 \ifx\reserved@a\cyrillicencoding\else
57.108 \PackageWarning{babel}%
57.109 {No input encoding specified for Russian language}
57.110 \fi\fi}

```

Now we define two commands that offer the possibility to switch between Cyrillic and Roman encodings.

`\cyrillictext` The command `\cyrillictext` will switch from Latin font encoding to the Cyrillic font encoding, the command `\latintext` switches back. This assumes that the ‘normal’ font encoding is a Latin one. These commands are *declarations*, for shorter peaces of text the commands `\textlatin` and `\textcyrillic` can be used.

```

57.111 %\DeclareRobustCommand{\latintext}{%
57.112 % \fontencoding{\latinencoding}\selectfont

```

```

57.113 % \def\encodingdefault{\latinencoding}}
57.114 \let\lat\latintext

```

`\textcyrillic` These commands take an argument which is then typeset using the requested font encoding.

```

57.115 \DeclareTextFontCommand{\textcyrillic}{\cyrillictext}
57.116 %\DeclareTextFontCommand{\textlatin}{\latintext}

```

We make the T_EX

```

57.117 %\ifx\ltxTeX\undefined\let\ltxTeX\TeX\fi
57.118 %\ProvideTextCommandDefault{\TeX}{\textlatin{\ltxTeX}}

```

and L^AT_EX logos encoding independent.

```

57.119 %\ifx\ltxLaTeX\undefined\let\ltxLaTeX\LaTeX\fi
57.120 %\ProvideTextCommandDefault{\LaTeX}{\textlatin{\ltxLaTeX}}

```

The next step consists of defining commands to switch to (and from) the Russian language.

`\captionsrussian` The macro `\captionsrussian` defines all strings used in the four standard document classes provided with L^AT_EX. The two commands `\cyr` and `\lat` activate Cyrillic resp. Latin encoding.

```

57.121 \addto\captionsrussian{%
57.122 %   FIXME: Where is the \prefacename used?
57.123   \def\prefacename{%
57.124     {\cyr\CYRP\cyrr\cyre\cyrd\cyri\cyrs\cyrl\cyro\cyrv\cyri\cyre}}%
57.125 %   {\cyr\CYRV\cyrv\cyre\cyrd\cyre\cyrn\cyri\cyre}}%
57.126   \def\refname{%
57.127     {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
57.128       \ \cyri\cyrl\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyrery}}%
57.129 % \def\refname{%
57.130 %   {\cyr\CYRL\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
57.131   \def\abstractname{%
57.132     {\cyr\CYRA\cyrn\cyrn\cyro\cyrt\cyra\cyrc\cyri\cyrya}}%
57.133   \def\bibname{%
57.134     {\cyr\CYRL\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
57.135 % \def\bibname{%
57.136 %   {\cyr\CYRB\cyri\cyrb\cyrl\cyri\cyro
57.137 %     \cyrg\cyrr\cyra\cyrf\cyri\cyrya}}%
57.138 % for reports according to GOST:
57.139 % \def\bibname{%
57.140 %   {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
57.141 %     \ \cyri\cyrs\cyrp\cyro\cyrl\cyrsftsn\cyrz\cyro\cyrv\cyra\cyrn
57.142 %       \cyrn\cyrery\cyrh\ \cyri\cyrs\cyrt\cyro\cyrch\cyrn\cyri
57.143 %       \cyrk\cyro\cyrv}}%
57.144   \def\chaptername{{\cyr\CYRG\cyrl\cyra\cyrv\cyra}}%
57.145 % \@ifundefined{chapter}{\cyr\CYRG\cyrl\cyra\cyrv\cyra}}%
57.146 %   \def\chaptername{{\cyr\CYRG\cyrl\cyra\cyrv\cyra}}}%
57.147   \def\appendixname{%
57.148     {\cyr\CYRP\cyrr\cyri\cyrl\cyro\cyrzh\cyre\cyrn\cyri\cyre}}%

```

There are two names for the Table of Contents that are used in Russian publications. For books (and reports) the second variant is appropriate, but for proceedings the first variant is preferred:

```

57.149   \@ifundefined{thechapter}%
57.150     {\def\contentsname{%
57.151       {\cyr\CYRS\cyro\cyrd\cyre\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}}%
57.152     {\def\contentsname{%
57.153       {\cyr\CYRO\cyrg\cyrl\cyra\cyrv\cyrl\cyre\cyrn\cyri\cyre}}}%
57.154   \def\listfigurename{%
57.155     {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
57.156       \ \cyri\cyrl\cyrl\cyryu\cyrs\cyrt\cyrr\cyra\cyrc\cyri\cyrishrt}}%
57.157 % \def\listfigurename{%

```

```

57.158 % {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
57.159 % \ \cyrr\cyri\cyrs\cyru\cyrn\cyrk\cyro\cyrv}}%
57.160 \def\listtablename{%
57.161 {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
57.162 \ \cyrt\cyra\cyrb\cyrl\cyri\cyrc}}%
57.163 \def\indexname{%
57.164 {\cyr\CYRP\cyrr\cyre\cyrd\cyrn\cyre\cyrt\cyrn\cyrrery\cyrishrt
57.165 \ \cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl\cyrsftsn}}%
57.166 \def\authorname{%
57.167 {\cyr\CYRI\cyrn\cyre\cyrn\cyrn\cyro\cyrishrt
57.168 \ \cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl\cyrsftsn}}%
57.169 \def\figurename{{\cyr\CYRR\cyri\cyrs.}}%
57.170 \def\tablename{{\cyr\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyra}}%
57.171 \def\partname{{\cyr\CYRCH\cyra\cyrs\cyrt\cyrsftsn}}%
57.172 \def\enclname{{\cyr\cyrv\cyrk\cyrl.}}%
57.173 \def\ccname{{\cyr\cyri\cyrs\cyrh.}}%
57.174 % \def\ccname{{\cyr\cyri\cyrz}}%
57.175 \def\headtoname{{\cyr\cyrv\cyrh.}}%
57.176 % \def\headtoname{{\cyr\cyrv}}%
57.177 \def\pagename{{\cyr\cyrs.}}%
57.178 % \def\pagename{{\cyr\cyrs\cyrt\cyrr.}}%
57.179 \def\seename{{\cyr\cyrs\cyrn.}}%
57.180 \def\alsoname{{\cyr\cyrs\cyrn.\ \cyrt\cyra\cyrk\cyrz\cyre}}%

57.181 \def\proofname{{\cyr\CYRD\cyro\cyrk\cyra\cyrz\cyra\cyrt
57.182 \cyre\cyrl\cyrsftsn\cyrs\cyrt\cyrv\cyro}}%
57.183 \def\glossaryname{Glossary}% <-- Needs translation
57.184 }

```

\daterussian The macro `\daterussian` redefines the command `\today` to produce Russian dates.

```

57.185 \def\daterussian{%
57.186 \def\today{\number\day~\ifcase\month\or
57.187 \cyrya\cyrn\cyrv\cyra\cyrr\cyrya\or
57.188 \cyrf\cyre\cyrv\cyrr\cyra\cyrl\cyrya\or
57.189 \cyrn\cyra\cyrr\cyrt\cyra\or
57.190 \cyra\cyrp\cyrr\cyre\cyrl\cyrya\or
57.191 \cyrn\cyra\cyrya\or
57.192 \cyri\cyryu\cyrn\cyrya\or
57.193 \cyri\cyryu\cyrl\cyrya\or
57.194 \cyra\cyrv\cyrg\cyru\cyrs\cyrt\cyra\or
57.195 \cyrs\cyre\cyrn\cyrt\cyrya\cyrb\cyrr\cyrya\or
57.196 \cyro\cyrk\cyrt\cyrya\cyrb\cyrr\cyrya\or
57.197 \cyrn\cyro\cyrya\cyrb\cyrr\cyrya\or
57.198 \cyrd\cyre\cyrk\cyra\cyrb\cyrr\cyrya\fi
57.199 \ \number\year~\cyrg.}}

```

\extrasrussian The macro `\extrasrussian` will perform all the extra definitions needed for the Russian language. The macro `\noextrasrussian` is used to cancel the actions of `\extrasrussian`.

The first action we define is to switch on the selected Cyrillic encoding whenever we enter ‘russian’.

```

57.200 \addto\extrasrussian{\cyrillictext}

```

When the encoding definition file was processed by \LaTeX the current font encoding is stored in `\latinencoding`, assuming that \LaTeX uses T1 or OT1 as default. Therefore we switch back to `\latinencoding` whenever the Russian language is no longer ‘active’.

```

57.201 \addto\noextrasrussian{\latintext}

```

\verbatim@font In order to get both Latin and Cyrillic letters in verbatim text we need to change the definition of an internal \LaTeX command somewhat:


```

57.202 %\def\verbatim@font{%
57.203 % \let\encodingdefault\latinencoding
57.204 % \normalfont\ttfamily
57.205 % \expandafter\def\csname\cyrillicencoding-cmd\endcsname##1##2{%
57.206 % \ifx\protect\@typeset@protect
57.207 % \begingroup\UseTextSymbol\cyrillicencoding##1\endgroup
57.208 % \else\noexpand##1\fi}}

```

The category code of the characters ‘:’, ‘;’, ‘!’, and ‘?’ is made `\active` to insert a little white space.

For Russian (as well as for German) the " character also is made active.

Note: It is *very* questionable whether the Russian typesetting tradition requires additional spacing before those punctuation signs. Therefore, we make the corresponding code optional. If you need it, then define the `frenchpunct` docstrip option in `babel.ins`.

Borrowed from french. Some users dislike automatic insertion of a space before ‘double punctuation’, and prefer to decide themselves whether a space should be added or not; so a hook `\NoAutoSpaceBeforeFDP` is provided: if this command is added (in file `russianb.cfg`, or anywhere in a document) `russianb` will respect your typing, and introduce a suitable space before ‘double punctuation’ *if and only if* a space is typed in the source file before those signs.

The command `\AutoSpaceBeforeFDP` switches back to the default behavior of `russianb`.

```

57.209 <*frenchpunct>
57.210 \initiate@active@char{:}
57.211 \initiate@active@char{;}
57.212 </frenchpunct>
57.213 <*frenchpunct | spanishligs>
57.214 \initiate@active@char{!}
57.215 \initiate@active@char{?}
57.216 </frenchpunct | spanishligs>
57.217 \initiate@active@char{"}

```

The code above is necessary because we need extra active characters. The character " is used as indicated in table 30.

We specify that the Russian group of shorthands should be used.

```

57.218 \addto\extrarussian{\languageshorthands{russian}}

```

These characters are ‘turned on’ once, later their definition may vary.

```

57.219 \addto\extrarussian{%
57.220 <frenchpunct> \bbl@activate{:}\bbl@activate{;}%
57.221 <frenchpunct | spanishligs> \bbl@activate{!}\bbl@activate{?}%
57.222 \bbl@activate{"}}
57.223 \addto\noextrarussian{%
57.224 <frenchpunct> \bbl@deactivate{:}\bbl@deactivate{;}%
57.225 <frenchpunct | spanishligs> \bbl@deactivate{!}\bbl@deactivate{?}%
57.226 \bbl@deactivate{"}}

```

The X2 and T2* encodings do not contain `spanish_shriek` and `spanish_query` symbols; as a consequence, the ligatures ‘?’ and ‘!’ do not work with them (these characters are useless for Cyrillic texts anyway). But we define the shorthands to emulate these ligatures (optionally).

We do not use `\latinencoding` here (but instead explicitly use OT1) because the user may choose T2A to be the primary encoding, but it does not contain these characters.

```

57.227 <*spanishligs>
57.228 \declare@shorthand{russian}{?}{\UseTextSymbol{OT1}\textquestiondown}
57.229 \declare@shorthand{russian}{!}{\UseTextSymbol{OT1}\textexclamdown}
57.230 </spanishligs>

```

`\russian@sh@;@` We have to reduce the amount of white space before `;`, `:` and `!`. This should only happen in horizontal mode, hence the test with `\ifhmode`.

```
\russian@sh@!@57.231 (*frenchpunct)
\russian@sh?@57.232 \declare@shorthand{russian}{;}{}%
57.233 \ifhmode
```

In horizontal mode we check for the presence of a ‘space’, ‘unskip’ if it exists and place a 0.1em kerning.

```
57.234 \ifdim\lastskip>\z@
57.235 \unskip\nobreak\kern.1em
57.236 \else
```

If no space has been typed, we add `\FDP@thinspace` which will be defined, up to the user’s wishes, as an automatic added thinspace, or as `\@empty`.

```
57.237 \FDP@thinspace
57.238 \fi
57.239 \fi
```

Now we can insert a ‘;’ character.

```
57.240 \string;}
```

The other definitions are very similar.

```
57.241 \declare@shorthand{russian}{:}{}%
57.242 \ifhmode
57.243 \ifdim\lastskip>\z@
57.244 \unskip\nobreak\kern.1em
57.245 \else
57.246 \FDP@thinspace
57.247 \fi
57.248 \fi
57.249 \string:}

57.250 \declare@shorthand{russian}{!}{}%
57.251 \ifhmode
57.252 \ifdim\lastskip>\z@
57.253 \unskip\nobreak\kern.1em
57.254 \else
57.255 \FDP@thinspace
57.256 \fi
57.257 \fi
57.258 \string!}

57.259 \declare@shorthand{russian}{?}{}%
57.260 \ifhmode
57.261 \ifdim\lastskip>\z@
57.262 \unskip\nobreak\kern.1em
57.263 \else
57.264 \FDP@thinspace
57.265 \fi
57.266 \fi
57.267 \string?}
```

`\AutoSpaceBeforeFDP` `\FDP@thinspace` is defined as unbreakable spaces if `\AutoSpaceBeforeFDP` is activated or as `\@empty` if `\NoAutoSpaceBeforeFDP` is in use. The default is `\FDP@thinspace` `\AutoSpaceBeforeFDP`.

```
57.268 \def\AutoSpaceBeforeFDP{%
57.269 \def\FDP@thinspace{\nobreak\kern.1em}}
57.270 \def\NoAutoSpaceBeforeFDP{\let\FDP@thinspace\@empty}
57.271 \AutoSpaceBeforeFDP
```

`\FDPon` The next macros allow to switch on/off activeness of double punctuation signs.

```
\FDPoff 57.272 \def\FDPon{\bbl@activate{}}%
57.273         \bbl@activate{;}%
57.274         \bbl@activate{?}%
57.275         \bbl@activate{!}%
57.276 \def\FDPoff{\bbl@deactivate{}}%
57.277         \bbl@deactivate{;}%
57.278         \bbl@deactivate{?}%
57.279         \bbl@deactivate{!}%
```

`\system@sh@:` When the active characters appear in an environment where their Russian behaviour is not wanted they should give an ‘expected’ result. Therefore we define `\system@sh@!:` shorthands at system level as well.

```
\system@sh@; 57.280 \declare@shorthand{system}{:}{\string;}
57.281 \declare@shorthand{system}{;}{\string;}
57.282 \frenchpunct
57.283 (*frenchpunct&!spanishlig)
57.284 \declare@shorthand{system}{!}{\string!}
57.285 \declare@shorthand{system}{?}{\string?}
57.286 \frenchpunct&!spanishlig)
```

To be able to define the function of “, we first define a couple of ‘support’ macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent “ can now be typed as “.

```
57.287 \begingroup \catcode'\ "12
57.288 \def\reserved@a{\endgroup
57.289 \def\SS{\mathchar"7019 }
57.290 \def\dq{"}}
57.291 \reserved@a
```

Now we can define the doublequote macros: german and french quotes. We use definitions of these quotes made in `babel.sty`. The french quotes are contained in the T2* encodings.

```
57.292 \declare@shorthand{russian}{"'}{\glqq}
57.293 \declare@shorthand{russian}{"'}{\grqq}
57.294 \declare@shorthand{russian}{"<"}{\flqq}
57.295 \declare@shorthand{russian}{">"}{\frqq}
```

Some additional commands:

```
57.296 \declare@shorthand{russian}{""}{\hskip\z@skip}
57.297 \declare@shorthand{russian}{"~"}{\textormath{\leavevmode\hbox{-}}{-}}
57.298 \declare@shorthand{russian}{"="}{\nobreak-\hskip\z@skip}
57.299 \declare@shorthand{russian}{"|"}{%
57.300 \textormath{\nobreak\discretionary{-}}{\kern.03em}%
57.301 \allowhyphens}{}}
```

The next two macros for “- and “--- are somewhat different. We must check whether the second token is a hyphen character:

```
57.302 \declare@shorthand{russian}{"-}{%
```

If the next token is ‘-’, we typeset an emdash, otherwise a hyphen sign:

```
57.303 \def\russian@sh@tmp{%
57.304 \if\russian@sh@next-\expandafter\russian@sh@emdash
57.305 \else\expandafter\russian@sh@hyphen\fi
57.306 }%
```

TEX looks for the next token after the first ‘-’: the meaning of this token is written to `\russian@sh@next` and `\russian@sh@tmp` is called.

```
57.307 \futurelet\russian@sh@next\russian@sh@tmp}
```

Here are the definitions of hyphen and emdash. First the hyphen:

```
57.308 \def\russian@sh@hyphen{%
57.309 \nobreak\-\bbl@allowhyphens}
```

For the emdash definition, there are the two parameters: we must ‘eat’ two last hyphen signs of our emdash. . . :

```
57.310 \def\russian@sh@emdash#1#2{\cdash-#1#2}
```

`\cdash` . . . these two parameters are useful for another macro: `\cdash`:

```
57.311 %\ifx\cdash\undefined % should be defined earlier
57.312 \def\cdash#1#2#3{\def\tempx@{#3}%
57.313 \def\tempa@{-}\def\tempb@{~}\def\tempc@{*}%
57.314 \ifx\tempx@\tempa@\@Acdash\else
57.315 \ifx\tempx@\tempb@\@Bcdash\else
57.316 \ifx\tempx@\tempc@\@Ccdash\else
57.317 \errmessage{Wrong usage of cdash}\fi\fi\fi}
```

second parameter (or third for `\cdash`) shows what kind of emdash to create in next step

"--- ordinary (plain) Cyrillic emdash inside text: an unbreakable thinspace will be inserted before only in case of a *space* before the dash (it is necessary for dashes after display maths formulae: there could be lists, enumerations etc. started with “— where *a* is ...” i.e., the dash starts a line). (Firstly there were planned rather soft rules for user: he may put a space before the dash or not. But it is difficult to place this thinspace automatically, i.e., by checking modes because after display formulae T_EX uses horizontal mode. Maybe there is a misunderstanding? Maybe there is another way?) After a dash a breakable thinspace is always placed;

```
57.318 % What is more grammatically: .2em or .2\fontdimen6\font ?
57.319 \def\@Acdash{\ifdim\lastskip>\z@\unskip\nobreak\hskip.2em\fi
57.320 \cyrdash\hskip.2em\ignorespaces}%
```

"--~ emdash in compound names or surnames (like Mendeleev–Klapeiron); this dash has no space characters around; after the dash some space is added `\exhyphenalty`

```
57.321 \def\@Bcdash{\leavevmode\ifdim\lastskip>\z@\unskip\fi
57.322 \nobreak\cyrdash\penalty\exhyphenpenalty\hskip\z@skip\ignorespaces}%
```

"--* for denoting direct speech (a space like `\enskip` must follow the emdash);

```
57.323 \def\@Ccdash{\leavevmode
57.324 \nobreak\cyrdash\nobreak\hskip.35em\ignorespaces}%
57.325 %\fi
```

`\cyrdash` Finally the macro for “body” of the Cyrillic emdash. The `\cyrdash` macro will be defined in case this macro hasn’t been defined in a fontenc file. For T2* fonts, `cyrdash` will be placed in the code of the English emdash thus it uses ligature ---.

```
57.326 % Is there an IF necessary?
57.327 \ifx\cyrdash\undefined
57.328 \def\cyrdash{\hbox to.8em{--\hss--}}
57.329 \fi
```

Here a really new macro—to place thinspace between initials. This macro used instead of `\,` allows hyphenation in the following surname.

```
57.330 \declare@shorthand{russian}{\,\}\{\nobreak\hskip.2em\ignorespaces}
```

`\mdqon` All that’s left to do now is to define a couple of commands for “.

```
\mdqoff57.331 \def\mdqon{\bbl@activate{}}
57.332 \def\mdqoff{\bbl@deactivate{}}
```

The Russian hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
57.333 \providehyphenmins{\CurrentOption}{\tw@{\tw@}
57.334 % temporary hack:
57.335 \ifx\englishhyphenmins\undefined
57.336 \def\englishhyphenmins{\tw@{\thr@@}
57.337 \fi
```

Now the action `\extrarussian` has to execute is to make sure that the command `\frenchspacing` is in effect. If this is not the case the execution of `\noextrarussian` will switch it off again.

```
57.338 \addto\extrarussian{\bbl@frenchspacing}
57.339 \addto\noextrarussian{\bbl@nonfrenchspacing}
```

Next we add a new enumeration style for Russian manuscripts with Cyrillic letters, and later on we define some math operator names in accordance with Russian typesetting traditions.

`\Asbuk` We begin by defining `\Asbuk` which works like `\Alph`, but produces (uppercase) Cyrillic letters instead of Latin ones. The letters YO, ISHRT, HRDSN, ERY, and SFTSN are skipped, as usual for such enumeration.

```
57.340 \def\Asbuk#1{\expandafter\@Asbuk\csname c@#1\endcsname}
57.341 \def\@Asbuk#1{\ifcase#1\or
57.342 \CYRA\or\CYRB\or\CYRV\or\CYRG\or\CYRD\or\CYRE\or\CYRZH\or
57.343 \CYRZ\or\CYRI\or\CYRK\or\CYRL\or\CYRM\or\CYRN\or\CYRO\or
57.344 \CYRP\or\CYRR\or\CYRS\or\CYRT\or\CYRU\or\CYRF\or\CYRH\or
57.345 \CYRC\or\CYRCH\or\CYRSH\or\CYRSHCH\or\CYREREV\or\CYRYU\or
57.346 \CYRYA\else\@ctrerr\fi}
```

`\asbuk` The macro `\asbuk` is similar to `\alph`; it produces lowercase Russian letters.

```
57.347 \def\asbuk#1{\expandafter\@asbuk\csname c@#1\endcsname}
57.348 \def\@asbuk#1{\ifcase#1\or
57.349 \cyra\or\cyrb\or\cyrv\or\cyrg\or\cyrd\or\cyre\or\cyrz\or
57.350 \cyrz\or\cyri\or\cyrk\or\cyrl\or\cyrm\or\cyrn\or\cyro\or
57.351 \cyrp\or\cyrr\or\cyrs\or\cyrt\or\cyru\or\cyrf\or\cyrh\or
57.352 \cyrc\or\cyrch\or\cyrsh\or\cyrshch\or\cyrerev\or\cyryu\or
57.353 \cyrya\else\@ctrerr\fi}
```

Set up default Cyrillic math alphabets. To use Cyrillic letters in math mode user should load the `textmath` package *before* loading `fontenc` package (or `babel`). Note, that by default Cyrillic letters are taken from upright font in math mode (unlike Latin letters).

```
57.354 %\RequirePackage{textmath}
57.355 \ifundefined{sym\cyrillicencoding letters}{\%
57.356 \SetSymbolFont{cyrillicencoding letters}{bold}{cyrillicencoding
57.357 \rmdefault\bfdefault\updefault
57.358 \DeclareSymbolFontAlphabet\cyrmathrm{cyrillicencoding letters}}
```

And we need a few commands to be able to switch to different variants.

```
57.359 \DeclareMathAlphabet\cyrmathbf{cyrillicencoding
57.360 \rmdefault\bfdefault\updefault
57.361 \DeclareMathAlphabet\cyrmathsf{cyrillicencoding
57.362 \sfdefault\mddefault\updefault
57.363 \DeclareMathAlphabet\cyrmathit{cyrillicencoding
57.364 \rmdefault\mddefault\itdefault
57.365 \DeclareMathAlphabet\cyrmathtt{cyrillicencoding
57.366 \ttdefault\mddefault\updefault
57.367 %
57.368 \SetMathAlphabet\cyrmathsf{bold}{cyrillicencoding
57.369 \sfdefault\bfdefault\updefault
57.370 \SetMathAlphabet\cyrmathit{bold}{cyrillicencoding
57.371 \rmdefault\bfdefault\itdefault
57.372 }
```

Some math functions in Russian math books have other names: e.g., `\sinh` in Russian is written as `\sh` etc. So we define a number of new math operators.

```

\sinh:
57.373 \def\sh{\mathop{\operator@font sh}\nolimits}
\cosh:
57.374 \def\ch{\mathop{\operator@font ch}\nolimits}
\tan:
57.375 \def\tg{\mathop{\operator@font tg}\nolimits}
\arctan:
57.376 \def\arctg{\mathop{\operator@font arctg}\nolimits}
arcctg:
57.377 \def\arcctg{\mathop{\operator@font arcctg}\nolimits}
The following macro conflicts with \th defined in Latin 1 encoding:
\tanh:
57.378 \addto\extrasrussian{%
57.379 \babel@save{\th}%
57.380 \let\ltx@th\th
57.381 \def\th{\textormath{\ltx@th}%
57.382 {\mathop{\operator@font th}\nolimits}}%
57.383 }
\cot:
57.384 \def\ctg{\mathop{\operator@font ctg}\nolimits}
\coth:
57.385 \def\cth{\mathop{\operator@font cth}\nolimits}
\csc:
57.386 \def\cosec{\mathop{\operator@font cosec}\nolimits}
And finally some other Russian mathematical symbols:
57.387 \def\Prob{\mathop{\kern\z@\mathsf{P}}\nolimits}
57.388 \def\Variance{\mathop{\kern\z@\mathsf{D}}\nolimits}
57.389 \def\nod{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrd.}}\nolimits}
57.390 \def\nok{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrk.}}\nolimits}
57.391 \def\NOD{\mathop{\cyrmathrm{\CYRN\CYRO\CYRD}}\nolimits}
57.392 \def\NOK{\mathop{\cyrmathrm{\CYRN\CYRO\CYRK}}\nolimits}
57.393 \def\Proj{\mathop{\cyrmathrm{\CYRP\cyrr}}\nolimits}

```

This is for compatibility with older Russian packages.

```

57.394 \DeclareRobustCommand{\No}{%
57.395 \ifmmode{\nfss@text{\textnumero}}\else\textnumero\fi}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

57.396 \ldf@finish{russian}
57.397 \code>

```

58 The Bulgarian language

The file `bulgarian.dtx`⁶⁸ provides the language-specific macros for the Bulgarian language.

Users should take note of the various “cyrillic” dashes available now (see below). These should remove many causes of headache. Also, although by default the Bulgarian quotation marks will appear automatically when typesetting in Bulgarian, it is better to use the new commands `\''` and `\"` which explicitly typeset them. Note: automatic switch to Bulgarian quotation is withdrawn for the moment and may not be reintroduced at all.

For this language the character `"` is made active. In table 32 an overview is given of its purpose.

<code>" </code>	disable ligature at this position.
<code>"-</code>	an explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>"---</code>	Cyrillic emdash in plain text.
<code>"--~</code>	Cyrillic emdash in compound names (surnames).
<code>"--*</code>	Cyrillic emdash for denoting direct speech.
<code>"</code>	like <code>"-</code> , but producing no hyphen sign (for compound words with hyphen, e.g. <code>x-"y</code> or some other signs as “disable/enable”).
<code>"~</code>	for a compound word mark without a breakpoint.
<code>"=</code>	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
<code>",</code>	thinspace for initials with a breakpoint in following surname.
<code>"‘</code>	for German left double quotes (looks like „).
<code>"’</code>	for German right double quotes (looks like “).
<code>"<</code>	for French left double quotes (looks like <<).
<code>"></code>	for French right double quotes (looks like >>).

Table 32: The extra definitions made by `bulgarian`

The quotes in table 32 can also be typeset by using the commands in table 33.

<code>\cdash---</code>	Cyrillic emdash in plain text.
<code>\cdash--~</code>	Cyrillic emdash in compound names (surnames).
<code>\cdash--*</code>	Cyrillic emdash for denoting direct speech.
<code>\glqq</code>	for German left double quotes (looks like „).
<code>\grqq</code>	for German right double quotes (looks like “).
<code>\flqq</code>	for French left double quotes (looks like <<).
<code>\frqq</code>	for French right double quotes (looks like >>).
<code>\dq</code>	the original quotes character (<code>"</code>).

Table 33: More commands which produce quotes, defined by `babel`

The French quotes are also available as ligatures `‘<<’` and `‘>>’` in 8-bit Cyrillic font encodings (LCY, X2, T2*) and as `‘<’` and `‘>’` characters in 7-bit Cyrillic font encodings (OT2 and LWN).

The quotation marks traditionally used in Bulgarian were borrowed from German or they keep their original names. French quotation marks may be seen as well in older books.

⁶⁸The file described in this section has version number ? and was last revised on ?. This file was initially derived from the August-1998 version of `russianb.dtx`.

It is (reasonably) backward compatible with the 1994/1996 (non-babel) bulgarian style (`bulgaria.sty`) by Georgi Boshnakov—files prepared for that style should compile successfully (with vastly improved appearance due to usage of standard fonts).

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
58.1 (*code)
58.2 \LdfInit{bulgarian}{captionsbulgarian}
```

When this file is read as an option, i.e., by the `\usepackage` command, `bulgarian` will be an ‘unknown’ language, in which case we have to make it known. So we check for the existence of `\l@bulgarian` to see whether we have to do something here.

```
58.3 \ifx\l@bulgarian\@undefined
58.4   \@nopatterns{Bulgarian}
58.5   \adddialect\l@bulgarian0
58.6 \fi
```

`\latinencoding` We need to know the encoding for text that is supposed to be which is active at the end of the `babel` package. If the `fontenc` package is loaded later, then ... too bad!

```
58.7 \let\latinencoding\cf@encoding
```

The user may choose between different available Cyrillic encodings—e.g., `X2`, `LCY`, or `LWN`. If the user wants to use a font encoding other than the default (`T2A`), he has to load the corresponding file *before* `bulgarian.sty`. This may be done in the following way:

```
\usepackage[LCY,OT1]{fontenc}      %overwrite the default encoding;
\usepackage[english,bulgarian]{babel}
```

Note: most people would prefer the `T2A` to `X2`, because `X2` does not contain Latin letters, and users should be very careful to switch the language every time they want to typeset a Latin word inside a Bulgarian phrase or vice versa. On the other hand, switching the language is a good practice anyway. With a decent text processing program it does not involve more work than switching between the Bulgarian and English keyboard. Moreover that the far most common disruption occurs as a result of forgetting to switch back to cyrillic keyboard.

We parse the `\cdp@list` containing the encodings known to \LaTeX in the order they were loaded. We set the `\cyrillicencoding` to the *last* loaded encoding in the list of supported Cyrillic encodings: `OT2`, `LWN`, `LCY`, `X2`, `T2C`, `T2B`, `T2A`, if any.

```
58.8 \def\reserved@a#1#2{%
58.9   \edef\reserved@b{#1}%
58.10  \edef\reserved@c{#2}%
58.11  \ifx\reserved@b\reserved@c
58.12    \let\cyrillicencoding\reserved@c
58.13  \fi}
58.14 \def\cdp@elt#1#2#3#4{%
58.15   \reserved@a{#1}{OT2}%
58.16   \reserved@a{#1}{LWN}%
58.17   \reserved@a{#1}{LCY}%
58.18   \reserved@a{#1}{X2}%
58.19   \reserved@a{#1}{T2C}%
58.20   \reserved@a{#1}{T2B}%
58.21   \reserved@a{#1}{T2A}}
58.22 \cdp@list
```

Now, if `\cyrillicencoding` is undefined, then the user did not load any of supported encodings. So, we have to set `\cyrillicencoding` to some default value. We test the presence of the encoding definition files in the order from less preferable to more preferable encodings. We use the lowercase names (i.e., `lcyenc.def` instead of `LCYenc.def`).

```
58.23 \ifx\cyrillicencoding\undefined
58.24   \IfFileExists{ot2enc.def}{\def\cyrillicencoding{OT2}}\relax
```



```

58.25 \IfFileExists{lwnenc.def}{\def\cyrillicencoding{LWN}}\relax
58.26 \IfFileExists{lcyenc.def}{\def\cyrillicencoding{LCY}}\relax
58.27 \IfFileExists{x2enc.def}{\def\cyrillicencoding{X2}}\relax
58.28 \IfFileExists{t2cenc.def}{\def\cyrillicencoding{T2C}}\relax
58.29 \IfFileExists{t2benc.def}{\def\cyrillicencoding{T2B}}\relax
58.30 \IfFileExists{t2aenc.def}{\def\cyrillicencoding{T2A}}\relax

```

If `\cyrillicencoding` is still undefined, then the user seems not to have a properly installed distribution. A fatal error.

```

58.31 \ifx\cyrillicencoding\undefined
58.32   \PackageError{babel}%
58.33     {No Cyrillic encoding definition files were found}%
58.34     {Your installation is incomplete. \MessageBreak
58.35     You need at least one of the following files: \MessageBreak
58.36     \space\space
58.37     x2enc.def, t2aenc.def, t2benc.def, t2cenc.def, \MessageBreak
58.38     \space\space
58.39     lcyenc.def, lwnenc.def, ot2enc.def.}%
58.40 \else

```

We avoid `\usepackage[\cyrillicencoding]{fontenc}` because we don't want to force the switch of `\encodingdefault`.

```

58.41   \lowercase
58.42   \expandafter{\expandafter\input\cyrillicencoding enc.def\relax}%
58.43 \fi
58.44 \fi

```

```

\PackageInfo{babel}
{Using '\cyrillicencoding' as a default Cyrillic encoding}%

```

```

58.45 \DeclareRobustCommand{\Bulgarian}{%
58.46   \fontencoding\cyrillicencoding\selectfont
58.47   \let\encodingdefault\cyrillicencoding
58.48   \expandafter\set@hyphenmins\bulgarianhyphenmins
58.49   \language\l@bulgarian}
58.50 \DeclareRobustCommand{\English}{%
58.51   \fontencoding\latinencoding\selectfont
58.52   \let\encodingdefault\latinencoding
58.53   \expandafter\set@hyphenmins\englishhyphenmins
58.54   \language\l@english}
58.55 \let\Bul\Bulgarian
58.56 \let\Bg\Bulgarian
58.57 \let\cyrillictext\Bulgarian
58.58 \let\cyr\Bulgarian
58.59 \let\Eng\English
58.60 \def\selectengl@language{\selectlanguage{english}}
58.61 \def\selectbgl@language{\selectlanguage{bulgarian}}

```

Since the X2 encoding does not contain Latin letters, we should make some redefinitions of L^AT_EX macros which implicitly produce Latin letters.

```

58.62 \expandafter\ifx\csname T@X2\endcsname\relax\else

```

We put `\latinencoding` in braces to avoid problems with `\@alph` inside minipages (e.g., footnotes inside minipages) where `\@alph` is expanded and we get for example `'\fontencoding OT1'` (`\fontencoding` is robust).

```

58.63 \def\@Alph@eng#1{{\fontencoding{\latinencoding}\selectfont
58.64   \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
58.65   K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or
58.66   X\or Y\or Z\else \@ctrerr\fi}}%
58.67 \def\@alph@eng#1{{\fontencoding{\latinencoding}\selectfont
58.68   \ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
58.69   k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or

```

```

58.70      x\or y\or z\else \@ctrerr\fi}}%
58.71 \let\@Alph\@Alph@eng
58.72 \let\@alph\@alph@eng

```

Unfortunately, the commands `\AA` and `\aa` are not encoding dependent in L^AT_EX (unlike e.g., `\oe` or `\DH`). They are defined as `\r{A}` and `\r{a}`. This leads to unpredictable results when the font encoding does not contain the Latin letters ‘A’ and ‘a’ (like X2).

```

58.73 \DeclareTextSymbolDefault{\AA}{OT1}
58.74 \DeclareTextSymbolDefault{\aa}{OT1}
58.75 \DeclareTextCommand{\AA}{OT1}{\r A}
58.76 \DeclareTextCommand{\aa}{OT1}{\r a}
58.77 \fi

```

The following block redefines the character class of uppercase Greek letters and some accents, if it is equal to 7 (variable family), to avoid incorrect results if the font encoding in some math family does not contain these characters in places of OT1 encoding. The code was taken from `amsmath.dtx`. See comments and further explanation there.

```

58.78 \begingroup\catcode'\="=12
58.79 % uppercase greek letters:
58.80 \def\@tempa#1{\expandafter\@tempb\meaning#1\relax\relax\relax\relax
58.81 "0000\@nil#1}
58.82 \def\@tempb#1"#2#3#4#5#6\@nil#7{%
58.83 \ifnum"#2=7 \count@"1#3#4#5\relax
58.84 \ifnum\count@<"1000 \else \global\mathchardef#7="0#3#4#5\relax \fi
58.85 \fi}
58.86 \@tempa\Gamma\@tempa\Delta\@tempa\Theta\@tempa\Lambda\@tempa\Xi
58.87 \@tempa\Pi\@tempa\Sigma\@tempa\Upsilon\@tempa\Phi\@tempa\Psi
58.88 \@tempa\Omega
58.89 % some accents:
58.90 \def\@tempa#1#2\@nil{\def\@tempc{#1}}\def\@tempb{\mathaccent}
58.91 \expandafter\@tempa\hat\relax\relax\@nil
58.92 \ifx\@tempb\@tempc
58.93 \def\@tempa#1\@nil{#1}%
58.94 \def\@tempb#1{\afterassignment\@tempa\mathchardef\@tempc=}%
58.95 \def\do#1"#2{}
58.96 \def\@tempd#1{\expandafter\@tempb#1\@nil
58.97 \ifnum\@tempc>"FFF
58.98 \xdef#1{\mathaccent"\expandafter\do\meaning\@tempc\space}%
58.99 \fi}
58.100 \@tempd\hat\@tempd\check\@tempd\tilde\@tempd\acute\@tempd\grave
58.101 \@tempd\dot\@tempd\ddot\@tempd\breve\@tempd\bar
58.102 \fi
58.103 \endgroup

```

The user should use the `inputenc` package when any 8-bit Cyrillic font encoding is used, selecting one of the Cyrillic input encodings. We do not assume any default input encoding, so the user should explicitly call the `inputenc` package by `\usepackage{inputenc}`. We also removed `\AtBeginDocument`, so `inputenc` should be used before `babel`.

```

58.104 \@ifpackageloaded{inputenc}{\fi}%
58.105 \def\reserved@a{LWN}%
58.106 \ifx\reserved@a\cyrillicencoding\else
58.107 \def\reserved@a{OT2}%
58.108 \ifx\reserved@a\cyrillicencoding\else
58.109 \PackageWarning{babel}%
58.110 {No input encoding specified for Bulgarian language}\fi\fi}

```

Now we define two commands that offer the possibility to switch between Cyrillic and Roman encodings.

`\cyrillictext` The command `\cyrillictext` will switch from Latin font encoding to the Cyrillic font encoding, the command `\latintext` switches back. This assumes that the ‘normal’ font encoding is a Latin one. These commands are *declarations*, for shorter peaces of text the commands `\textlatin` and `\textcyrillic` can be used.

We comment out `\latintext` since it is defined in the core of babel (babel.def). We add the shorthand `\lat` for `\latintext`. Note that `\cyrillictext` has been defined above.

```
58.111 % \DeclareRobustCommand{\latintext}{%
58.112 % \fontencoding{\latinencoding}\selectfont
58.113 % \def\encodingdefault{\latinencoding}}
58.114 \let\lat\latintext
```

`\textcyrillic` These commands take an argument which is then typeset using the requested font encoding. `\textlatin` is commented out since it is defined in the core of babel. (It is defined there with `\DeclareRobustCommand` instead.)

```
58.115 \DeclareTextFontCommand{\textcyrillic}{\cyrillictext}
58.116 % \DeclareTextFontCommand{\textlatin}{\latintext}
```

The next step consists of defining commands to switch to (and from) the Bulgarian language.

`\captionsbulgarian` The macro `\captionsbulgarian` defines all strings used in the four standard document classes provided with L^AT_EX. The two commands `\cyr` and `\lat` activate Cyrillic resp. Latin encoding.

```
58.117 \addto\captionsbulgarian{%
58.118   \def\prefacename{%
58.119     {\cyr\CYRP\cyrr\cyre\cyrd\cyrg\cyro\cyrv\cyro\cyrr}}%
58.120   \def\refname{%
58.121     {\cyr\CYRL\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
58.122   \def\abstractname{%
58.123     {\cyr\CYRA\cyrb\cyrs\cyrt\cyrr\cyra\cyrk\cyrt}}%
58.124   \def\bibname{%
58.125     {\cyr\CYRB\cyri\cyrb\cyrl\cyri\cyro\cyrg\cyrr\cyra\cyrf\cyri\cyrya}}%
58.126   \def\chaptername{%
58.127     {\cyr\CYRG\cyrl\cyra\cyrv\cyra}}%
58.128   \def\appendixname{%
58.129     {\cyr\CYRP\cyrr\cyri\cyrl\cyro\cyrzh\cyre\cyrn\cyri\cyre}}%
58.130   \def\contentsname{%
58.131     {\cyr\CYRS\cyrhdsn\cyrd\cyrhdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
58.132   \def\listfigurename{%
58.133     {\cyr\CYRS\cyrp\cyri\cyrs\cyrhdsn\cyrk\ \cyrn\cyra\ \cyrf\cyri\cyrg\cyru\cyrr\cyri\cyrt}}%
58.134   \def\listtablename{%
58.135     {\cyr\CYRS\cyrp\cyri\cyrs\cyrhdsn\cyrk\ \cyrn\cyra\ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyri}}%
58.136   \def\indexname{%
58.137     {\cyr\CYRA\cyrz\cyrb\cyru\cyrch\cyre\cyrn\ \cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl}}%
58.138   \def\authorname{%
58.139     {\cyr\CYRI\cyrm\cyre\cyrn\cyre\cyrn\ \cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl}}%
58.140   \def\figurename{%
58.141     {\cyr\CYRF\cyri\cyrg\cyru\cyrr\cyra}}%
58.142   \def\tablename{%
58.143     {\cyr\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyra}}%
58.144   \def\partname{%
58.145     {\cyr\CYRCH\cyra\cyrs\cyrt}}%
58.146   \def\enclname{%
58.147     {\cyr\CYRP\cyrr\cyri\cyrl\cyro\cyrzh\cyre\cyrn\cyri\cyrya}}%
58.148   \def\ccname{%
58.149     {\cyr\cyrk\cyro\cyrp\cyri\cyrya}}%
58.150   \def\headtoname{%
58.151     {\cyr\CYZ\cyra}}%
58.152   \def\pagename{%
```

```

58.153   {\cyr\CYRS\cyrt\cyrr.}}%
58.154 \def\seename{%
58.155   {\cyr\cyrv\cyrzh.}}%
58.156 \def\alsoname{%
58.157   {\cyr\cyrv\cyrzh.\ \cyr\cyrhrdsn\cyrshch\cyro\ \cyri}}%
58.158 \def\proofname{Proof}% <-- Needs translation
58.159 \def\glossaryname{Glossary}% <-- Needs translation
58.160 }

```

`\datebulgarian` The macro `\datebulgarian` redefines the command `\today` to produce Bulgarian dates. It also provides the command `\todayRoman` which produces the date with the month in capital roman numerals, a popular format for dates in Bulgarian.

```

58.161 \def\datebulgarian{%
58.162   \def\month@bulgarian{\ifcase\month\or
58.163     \cyr\cyra\cyrn\cyru\cyra\cyrr\cyri\or
58.164     \cyr\cyrf\cyre\cyrv\cyrr\cyru\cyra\cyrr\cyri\or
58.165     \cyr\cyrm\cyra\cyrr\cyrt\or
58.166     \cyr\cyra\cyrp\cyrr\cyri\cyrl\or
58.167     \cyr\cyrm\cyra\cyrishrt\or
58.168     \cyr\cyryu\cyrn\cyri\or
58.169     \cyr\cyryu\cyrl\cyri\or
58.170     \cyr\cyra\cyrv\cyrg\cyru\cyr\cyrt\or
58.171     \cyr\cyr\cyre\cyrp\cyrt\cyre\cyr\cyrm\cyrv\cyrr\cyri\or
58.172     \cyr\cyro\cyrk\cyrt\cyro\cyr\cyrm\cyrv\cyrr\cyri\or
58.173     \cyr\cyrn\cyro\cyre\cyr\cyrm\cyrv\cyrr\cyri\or
58.174     \cyr\cyrd\cyre\cyrk\cyre\cyr\cyrm\cyrv\cyrr\cyri\fi}%
58.175   \def\month@Roman{\expandafter\@Roman\month}%
58.176   \def\today{\number\day~\month@bulgarian\ \number\year~\cyr.}%
58.177   \def\todayRoman{\number\day.\,\month@Roman.\,\number\year~\cyr.}%
58.178 }

```

`\todayRoman` The month is often written with roman numbers in Bulgarian dates. Here we define date in this format:

```

58.179 \def\Romannumeral#1{\uppercase\expandafter\romannumeral #1}%
58.180 \def\todayRoman{\number\day.\Romannumeral{\month}.\number\year~\cyr.}

```

`\extrasbulgarian` The macro `\extrasbulgarian` will perform all the extra definitions needed for the Bulgarian language. The macro `\noextrasbulgarian` is used to cancel the actions of `\extrasbulgarian`.

The first action we define is to switch on the selected Cyrillic encoding whenever we enter ‘bulgarian’.

```

58.181 \addto\extrasbulgarian{\cyrillictext}

```

When the encoding definition file was processed by \LaTeX the current font encoding is stored in `\latinencoding`, assuming that \LaTeX uses T1 or OT1 as default. Therefore we switch back to `\latinencoding` whenever the Bulgarian language is no longer ‘active’.

```

58.182 \addto\noextrasbulgarian{\latintext}

```

For Bulgarian the " character also is made active.

```

58.183 \initiate@active@char{"}

```

The code above is necessary because we need extra active characters. The character " is used as indicated in table 32. We specify that the Bulgarian group of shorthands should be used.

```

58.184 \addto\extrasbulgarian{\languageshorthands{bulgarian}}

```

These characters are ‘turned on’ once, later their definition may vary.

```

58.185 \addto\extrasbulgarian{%
58.186   \bbl@activate{"}%
58.187 \addto\noextrasbulgarian{%
58.188   \bbl@deactivate{"}%

```

The X2 and T2* encodings do not contain `spanish_shriek` and `spanish_query` symbols; as a consequence, the ligatures ‘?’ and ‘!’ do not work with them (these characters are useless for Cyrillic texts anyway). But we define the shorthands to emulate these ligatures (optionally).

We do not use `\latinencoding` here (but instead explicitly use OT1) because the user may choose T2A to be the primary encoding, but it does not contain these characters.

```
58.189 (*spanishligs)
58.190 \declare@shorthand{bulgarian}{?'}{\UseTextSymbol{OT1}\textquestiondown}
58.191 \declare@shorthand{bulgarian}{!'}{\UseTextSymbol{OT1}\textexclamdown}
58.192 /spanishligs
```

To be able to define the function of “”, we first define a couple of ‘support’ macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent “ can now be typed as “.

```
58.193 \begingroup \catcode'\ "12
58.194 \def\reserved@a{\endgroup
58.195 \def\@SS{\mathchar"7019}
58.196 \def\dq{"}}
58.197 \reserved@a
```

Now we can define the doublequote macros: german and french quotes. We use definitions of these quotes made in `babel.sty`. The french quotes are contained in the T2* encodings.

```
58.198 \declare@shorthand{bulgarian}{"'}{\glqq}
58.199 \declare@shorthand{bulgarian}{"'}{\grqq}
58.200 \declare@shorthand{bulgarian}{"<'}{\flqq}
58.201 \declare@shorthand{bulgarian}{">'}{\frqq}
```

Some additional commands:

```
58.202 \declare@shorthand{bulgarian}{""}{\hskip\z@skip}
58.203 \declare@shorthand{bulgarian}{""}{\textormath{\leavevmode\hbox{-}}{-}}
58.204 \declare@shorthand{bulgarian}{""}{\nobreak-\hskip\z@skip}
58.205 \declare@shorthand{bulgarian}{""}{\}%
58.206 \textormath{\nobreak\discretionary{-}}{\kern.03em}%
58.207 \allowhyphens}{}}
```

The next two macros for “- and “--- are somewhat different. We must check whether the second token is a hyphen character:

```
58.208 \declare@shorthand{bulgarian}{"-}{\%
```

If the next token is ‘-’, we typeset an emdash, otherwise a hyphen sign:

```
58.209 \def\bulgarian@sh@tmp{%
58.210 \if\bulgarian@sh@next-\expandafter\bulgarian@sh@emdash
58.211 \else\expandafter\bulgarian@sh@hyphen\fi
58.212 }%
```

TeX looks for the next token after the first ‘-’: the meaning of this token is written to `\bulgarian@sh@next` and `\bulgarian@sh@tmp` is called.

```
58.213 \futurelet\bulgarian@sh@next\bulgarian@sh@tmp}
```

Here are the definitions of hyphen and emdash. First the hyphen:

```
58.214 \def\bulgarian@sh@hyphen{\nobreak\-\bbl@allowhyphens}
```

For the emdash definition, there are the two parameters: we must ‘eat’ two last hyphen signs of our emdash . . . :

```
58.215 \def\bulgarian@sh@emdash#1#2{\cdash-#1#2}
```

`\cdash` ... these two parameters are useful for another macro: `\cdash`:

```

58.216 \ifx\cdash\undefined % should be defined earlier
58.217 \def\cdash#1#2#3{\def\tempx@{#3}%
58.218 \def\tempa@{-}\def\tempb@{~}\def\tempc@{*}%
58.219 \ifx\tempx@\tempa@\@Acdash\else
58.220 \ifx\tempx@\tempb@\@Bcdash\else
58.221 \ifx\tempx@\tempc@\@Ccdash\else
58.222 \errmessage{Wrong usage of cdash}\fi\fi\fi}

```

second parameter (or third for `\cdash`) shows what kind of emdash to create in next step

"--- ordinary (plain) Cyrillic emdash inside text: an unbreakable thinspace will be inserted before only in case of a *space* before the dash (it is necessary for dashes after display maths formulae: there could be lists, enumerations etc. started with “—where *a* is ...” i.e., the dash starts a line). (Firstly there were planned rather soft rules for user:he may put a space before the dash or not. But it is difficult to place this thinspace automatically, i.e., by checking modes because after display formulae T_EX uses horizontal mode. Maybe there is a misunderstanding? Maybe there is another way?) After a dash a breakable thinspace is always placed;

```

58.223 % What is more grammatically: .2em or .2\fontdimen6\font?
58.224 \def\@Acdash{\ifdim\lastskip>\z@\unskip\nobreak\hskip.2em\fi
58.225 \cyrdash\hskip.2em\ignorespaces}%

```

"--~ emdash in compound names or surnames (like Mendeleev–Klapeiron); this dash has no space characters around; after the dash some space is added `\exhyphenalty`

```

58.226 \def\@Bcdash{\leavevmode\ifdim\lastskip>\z@\unskip\fi
58.227 \nobreak\cyrdash\penalty\exhyphenpenalty\hskip\z@skip\ignorespaces}%

```

"--* for denoting direct speech (a space like `\enskip` must follow the emdash);

```

58.228 \def\@Ccdash{\leavevmode
58.229 \nobreak\cyrdash\nobreak\hskip.35em\ignorespaces}%
58.230 %\fi

```

`\cyrdash` Finally the macro for “body” of the Cyrillic emdash. The `\cyrdash` macro will be defined in case this macro hasn’t been defined in a fontenc file. For T2*fonts, `cyrdash` will be placed in the code of the English emdash thus it uses ligature ---.

```

58.231 % Is there an IF necessary?
58.232 \ifx\cyrdash\undefined
58.233 \def\cyrdash{\hbox to.8em{--\hss--}}
58.234 \fi

```

Here a really new macro—to place thinspace between initials. This macro used instead of `\,` allows hyphenation in the following surname.

```

58.235 \declare@shorthand{bulgarian}{",}{\nobreak\hskip.2em\ignorespaces}

```

The Bulgarian hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```

58.236 \providehyphenmins{\CurrentOption}{\tw@\tw@}
58.237 \fi

```

Now the action `\extrasbulgarian` has to execute is to make sure that the command `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasbulgarian` will switch it off again.

```

58.238 \addto\extrasbulgarian{\bbl@frenchspacing}
58.239 \addto\noextrasbulgarian{\bbl@nonfrenchspacing}

```

Make the double quotes produce the traditional quotes used in Bulgarian texts (these are the German quotes).

```

58.240 % \initiate@active@char{'}
58.241 % \initiate@active@char{' }
58.242 % \addto\extrasbulgarian{%
58.243 %   \bbl@activate{'}}
58.244 % \addto\extrasbulgarian{%
58.245 %   \bbl@activate{' }}
58.246 % \addto\noextrasbulgarian{%
58.247 %   \bbl@deactivate{'}}
58.248 % \addto\noextrasbulgarian{%
58.249 %   \bbl@deactivate{' }}
58.250 % \def\mlron{\bbl@activate{'}\bbl@activate{'}}
58.251 % \def\mlroff{\bbl@deactivate{'}\bbl@deactivate{'}}
58.252 % \declare@shorthand{bulgarian}{' '}{\glqq}
58.253 % \declare@shorthand{bulgarian}{' '}{\grqq}

```

Next we add a new enumeration style for Bulgarian manuscripts with Cyrillic letters, and later on we define some math operator names in accordance with Bulgarian typesetting traditions.

`\@Alph@bul` We begin by defining `\@Alph@bul` which works like `\@Alph`, but produces (uppercase) Cyrillic letters instead of Latin ones. The letters ISHRT, HRDSN and SFTSN are skipped, as usual for such enumeration.

```

58.254 \def\enumBul{\let\@Alph\@Alph@bul \let\@alph\@alph@bul}
58.255 \def\enumEng{\let\@Alph\@Alph@eng \let\@alph\@alph@eng}
58.256 \def\enumLat{\let\@Alph\@Alph@eng \let\@alph\@alph@eng}
58.257 \addto\extrasbulgarian{\enumBul}
58.258 \addto\noextrasbulgarian{\enumLat}
58.259 \def\@Alph@bul#1{%
58.260   \ifcase#1\or
58.261   \CYRA\or \CYRB\or \CYRV\or \CYRG\or \CYRD\or \CYRE\or \CYRZH\or
58.262   \CYRZ\or \CYRI\or \CYRK\or \CYRL\or \CYRM\or \CYRN\or \CYRO\or
58.263   \CYRP\or \CYRR\or \CYRS\or \CYRT\or \CYRU\or \CYRF\or \CYRH\or
58.264   \CYRC\or \CYRCH\or \CYRSH\or \CYRSHCH\or \CYRYU\or \CYRYA\else
58.265   \@ctrerr\fi
58.266 }
58.267 \def\@Alph@eng#1{%
58.268   \ifcase#1\or
58.269   A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or K\or L\or M\or
58.270   N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or Y\or Z\else
58.271   \@ctrerr\fi
58.272 }

```

`\@alph@bul` The macro `\@alph@bul` is similar to `\@Alph@bul`; it produces lowercase Bulgarian letters.

```

58.273 \def\@alph@bul#1{%
58.274   \ifcase#1\or
58.275   \cyra\or \cyrb\or \cyrv\or \cyrg\or \cyrd\or \cyre\or \cyrzh\or
58.276   \cyrz\or \cyri\or \cyrk\or \cyrl\or \cyrm\or \cyrn\or \cyro\or
58.277   \cyrp\or \cyrr\or \cyrs\or \cyrt\or \cyru\or \cyrf\or \cyrh\or
58.278   \cyrc\or \cyrch\or \cyrsh\or \cyrshch\or \cyryu\or \cyrya\else
58.279   \@ctrerr\fi
58.280 }
58.281 \def\@alph@eng#1{%
58.282   \ifcase#1\or
58.283   a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or k\or l\or m\or
58.284   n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or y\or z\else
58.285   \@ctrerr\fi
58.286 }

```

Set up default Cyrillic math alphabets. To use Cyrillic letters in math mode user should load the `textmath` package *before* loading fontenc package (or `babel`). Note, that by default Cyrillic letters are taken from upright font in math mode (unlike Latin letters).

```
58.287 %\RequirePackage{textmath}
58.288 \@ifundefined{sym\cyrillicencoding letters}{\cyrillicencoding
58.289 \SetSymbolFont{\cyrillicencoding letters}{bold}\cyrillicencoding
58.290 \rmdefault\bfdefault\updefault
58.291 \DeclareSymbolFontAlphabet\cyrmathrm{\cyrillicencoding letters}
```

And we need a few commands to be able to switch to different variants.

```
58.292 \DeclareMathAlphabet\cyrmathbf\cyrillicencoding
58.293 \rmdefault\bfdefault\updefault
58.294 \DeclareMathAlphabet\cyrmathsf\cyrillicencoding
58.295 \sfdefault\mddefault\updefault
58.296 \DeclareMathAlphabet\cyrmathit\cyrillicencoding
58.297 \rmdefault\mddefault\itdefault
58.298 \DeclareMathAlphabet\cyrmathtt\cyrillicencoding
58.299 \ttdefault\mddefault\updefault
58.300 \SetMathAlphabet\cyrmathsf{bold}\cyrillicencoding
58.301 \sfdefault\bfdefault\updefault
58.302 \SetMathAlphabet\cyrmathit{bold}\cyrillicencoding
58.303 \rmdefault\bfdefault\itdefault
58.304 }
```

Some math functions in Bulgarian math books have other names: e.g., `sinh` in Bulgarian is written as `sh` etc. So we define a number of new math operators.

```
\sinh:
58.305 \def\sh{\mathop{\operator@font sh}\nolimits}
\cosh:
58.306 \def\ch{\mathop{\operator@font ch}\nolimits}
\tan:
58.307 \def\tg{\mathop{\operator@font tg}\nolimits}
\arctan:
58.308 \def\arctg{\mathop{\operator@font arctg}\nolimits}
\arccot:
58.309 \def\arcctg{\mathop{\operator@font arcctg}\nolimits}
```

The following macro conflicts with `\th` defined in Latin 1 encoding: `\tanh`:

```
58.310 \addto\extrasrussian{%
58.311 \babel@save{\th}%
58.312 \let\ltx@th\th
58.313 \def\th{\textormath{\ltx@th}%
58.314 {\mathop{\operator@font th}\nolimits}}%
58.315 }
\cot:
58.316 \def\ctg{\mathop{\operator@font ctg}\nolimits}
\coth:
58.317 \def\cth{\mathop{\operator@font cth}\nolimits}
\csc:
58.318 \def\cosec{\mathop{\operator@font cosec}\nolimits}
```

This is for compatibility with older Bulgarian packages.

```
58.319 \DeclareRobustCommand{\No}{%
58.320 \ifmmode{\nfss@text{\textnumero}}\else\textnumero\fi}
```


The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
58.321 \ldf@finish{bulgarian}
```

```
58.322 \code}
```

59 The Ukrainian language

The file `ukraineb.dtx`⁶⁹ defines all the language-specific macros for the Ukrainian language. It needs the file `cyrnod` for success documentation with Ukrainian encodings (see below).

For this language the character " is made active. In table 34 an overview is given of its purpose.

"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
"---	Cyrillic emdash in plain text.
"--~	Cyrillic emdash in compound names (surnames).
"--*	Cyrillic emdash for denoting direct speech.
"	like "-", but producing no hyphen sign (for compound words with hyphen, e.g. x-"y or some other signs as "disable/enable").
"~	for a compound word mark without a breakpoint.
"=	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
",	thinspace for initials with a breakpoint in following surname.
"‘	for German left double quotes (looks like „).
"’	for German right double quotes (looks like “).
"<	for French left double quotes (looks like <<).
">	for French right double quotes (looks like >>).

Table 34: The extra definitions made by `ukraineb`

The quotes in table 34 (see, also table 30) can also be typeset by using the commands in table 35 (see, also table 31).

<code>\cdash---</code>	Cyrillic emdash in plain text.
<code>\cdash--~</code>	Cyrillic emdash in compound names (surnames).
<code>\cdash--*</code>	Cyrillic emdash for denoting direct speech.
<code>\glqq</code>	for German left double quotes (looks like „).
<code>\grqq</code>	for German right double quotes (looks like “).
<code>\flqq</code>	for French left double quotes (looks like <<).
<code>\frqq</code>	for French right double quotes (looks like >>).
<code>\dq</code>	the original quotes character (").

Table 35: More commands which produce quotes, defined by `babel`

The French quotes are also available as ligatures ‘<<’ and ‘>>’ in 8-bit Cyrillic font encodings (LCY, X2, T2*) and as ‘<’ and ‘>’ characters in 7-bit Cyrillic font encodings (OT2 and LWN).

The quotation marks traditionally used in Ukrainian and Russian languages were borrowed from other languages (e.g. French and German) so they keep their original names.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

59.1 (*code)

59.2 `\LdfInit{ukrainian}{captionsukrainian}`

⁶⁹The file described in this section has version number ?. This file was derived from the `ruasianb.dtx` version 1.1g.

When this file is read as an option, i.e., by the `\usepackage` command, `ukraineb` will be an ‘unknown’ language, in which case we have to make it known. So we check for the existence of `\l@ukrainian` to see whether we have to do something here.

```
59.3 \ifx\l@ukrainian\@undefined
59.4 \@nopatterns{Ukrainian}
59.5 \adddialect\l@ukrainian0
59.6 \fi
```

`\latinencoding` We need to know the encoding for text that is supposed to be which is active at the end of the `babel` package. If the `fontenc` package is loaded later, then... too bad!

```
59.7 \let\latinencoding\cf@encoding
```

The user may choose between different available Cyrillic encodings—e.g., `X2`, `LCY`, or `LWN`. Hopefully, `X2` will eventually replace the two latter encodings (`LCY` and `LWN`). If the user wants to use another font encoding than the default (`T2A`), he has to load the corresponding file *before* `ukraineb.sty`. This may be done in the following way:

```
% override the default X2 encoding used in Babel
\usepackage[LCY,OT1]{fontenc}
\usepackage[english,ukrainian]{babel}
```

Note: for the Ukrainian language, the `T2A` encoding is better than `X2`, because `X2` does not contain Latin letters, and users should be very careful to switch the language every time they want to typeset a Latin word inside a Ukrainian phrase or vice versa.

We parse the `\cdp@list` containing the encodings known to \LaTeX in the order they were loaded. We set the `\cyrillicencoding` to the *last* loaded encoding in the list of supported Cyrillic encodings: `OT2`, `LWN`, `LCY`, `X2`, `T2C`, `T2B`, `T2A`, if any.

```
59.8 \def\reserved@a#1#2{%
59.9 \edef\reserved@b{#1}%
59.10 \edef\reserved@c{#2}%
59.11 \ifx\reserved@b\reserved@c
59.12 \let\cyrillicencoding\reserved@c
59.13 \fi}
59.14 \def\cdp@elt#1#2#3#4{%
59.15 \reserved@a{#1}{OT2}%
59.16 \reserved@a{#1}{LWN}%
59.17 \reserved@a{#1}{LCY}%
59.18 \reserved@a{#1}{X2}%
59.19 \reserved@a{#1}{T2C}%
59.20 \reserved@a{#1}{T2B}%
59.21 \reserved@a{#1}{T2A}}
59.22 \cdp@list
```

Now, if `\cyrillicencoding` is undefined, then the user did not load any of supported encodings. So, we have to set `\cyrillicencoding` to some default value. We test the presence of the encoding definition files in the order from less preferable to more preferable encodings. We use the lowercase names (i.e., `lcyenc.def` instead of `LCYenc.def`).

```
59.23 \ifx\cyrillicencoding\undefined
59.24 \IfFileExists{ot2enc.def}{\def\cyrillicencoding{OT2}}\relax
59.25 \IfFileExists{lwnenc.def}{\def\cyrillicencoding{LWN}}\relax
59.26 \IfFileExists{lcyenc.def}{\def\cyrillicencoding{LCY}}\relax
59.27 \IfFileExists{x2enc.def}{\def\cyrillicencoding{X2}}\relax
59.28 \IfFileExists{t2cenc.def}{\def\cyrillicencoding{T2C}}\relax
59.29 \IfFileExists{t2benc.def}{\def\cyrillicencoding{T2B}}\relax
59.30 \IfFileExists{t2aenc.def}{\def\cyrillicencoding{T2A}}\relax
```

If `\cyrillicencoding` is still undefined, then the user seems not to have a properly installed distribution. A fatal error.

```

59.31 \ifx\cyrillicencoding\undefined
59.32   \PackageError{babel}%
59.33     {No Cyrillic encoding definition files were found}%
59.34     {Your installation is incomplete.\MessageBreak
59.35       You need at least one of the following files:\MessageBreak
59.36       \space\space
59.37       x2enc.def, t2aenc.def, t2benc.def, t2cenc.def,\MessageBreak
59.38       \space\space
59.39       lcyenc.def, lwnenc.def, ot2enc.def.}%
59.40 \else

```

We avoid `\usepackage[\cyrillicencoding]{fontenc}` because we don't want to force the switch of `\encodingdefault`.

```

59.41   \lowercase
59.42   \expandafter{\expandafter\input\cyrillicencoding enc.def\relax}%
59.43 \fi
59.44 \fi

```

```

\PackageInfo{babel}
{Using ‘\cyrillicencoding’ as a default Cyrillic encoding}%

```

```

59.45 \DeclareRobustCommand{\Ukrainian}{%
59.46   \fontencoding\cyrillicencoding\selectfont
59.47   \let\encodingdefault\cyrillicencoding
59.48   \expandafter\set@hyphenmins\ukrainianhyphenmins
59.49   \language\l@ukrainian}%
59.50 \DeclareRobustCommand{\English}{%
59.51   \fontencoding\latinencoding\selectfont
59.52   \let\encodingdefault\latinencoding
59.53   \expandafter\set@hyphenmins\englishhyphenmins
59.54   \language\l@english}%
59.55 \let\Ukr\Ukrainian
59.56 \let\Eng\English
59.57 \let\cyrillictext\Ukrainian
59.58 \let\cyr\Ukrainian

```

Since the X2 encoding does not contain Latin letters, we should make some redefinitions of L^AT_EX macros which implicitly produce Latin letters.

```

59.59 \expandafter\ifx\csname T@X2\endcsname\relax\else

```

We put `\latinencoding` in braces to avoid problems with `\@alph` inside minipages (e.g., footnotes inside minipages) where `\@alph` is expanded and we get for example ‘`\fontencoding OT1`’ (`\fontencoding` is robust).

```

59.60 \def\@alph#1{\fontencoding{\latinencoding}\selectfont
59.61   \ifcase#1\or
59.62     a\or b\or c\or d\or e\or f\or g\or h\or
59.63     i\or j\or k\or l\or m\or n\or o\or p\or
59.64     q\or r\or s\or t\or u\or v\or w\or x\or
59.65     y\or z\else\@ctrerr\fi}%
59.66 \def\@Alph#1{\fontencoding{\latinencoding}\selectfont
59.67   \ifcase#1\or
59.68     A\or B\or C\or D\or E\or F\or G\or H\or
59.69     I\or J\or K\or L\or M\or N\or O\or P\or
59.70     Q\or R\or S\or T\or U\or V\or W\or X\or
59.71     Y\or Z\else\@ctrerr\fi}%

```

Unfortunately, the commands `\AA` and `\aa` are not encoding dependent in L^AT_EX (unlike e.g., `\oe` or `\DH`). They are defined as `\r{A}` and `\r{a}`. This leads to unpredictable results when the font encoding does not contain the Latin letters ‘A’ and ‘a’ (like X2).

```

59.72 \DeclareTextSymbolDefault{\AA}{OT1}
59.73 \DeclareTextSymbolDefault{\aa}{OT1}
59.74 \DeclareTextCommand{\aa}{OT1}{\r a}
59.75 \DeclareTextCommand{\AA}{OT1}{\r A}
59.76 \fi

```

The following block redefines the character class of uppercase Greek letters and some accents, if it is equal to 7 (variable family), to avoid incorrect results if the font encoding in some math family does not contain these characters in places of OT1 encoding. The code was taken from `amsmath.dtx`. See comments and further explanation there.

```

59.77 % \begingroup\catcode'\=12
59.78 % % uppercase greek letters:
59.79 % \def\@tempa#1{\expandafter\@tempb\meaning#1\relax\relax\relax\relax
59.80 % "0000\@nil#1}
59.81 % \def\@tempb#1"#2#3#4#5#6\@nil#7{%
59.82 % \ifnum"#2=7 \count@"1#3#4#5\relax
59.83 % \ifnum\count@<"1000 \else \global\mathchardef#7="0#3#4#5\relax \fi
59.84 % \fi}
59.85 % \@tempa\Gamma\@tempa\Delta\@tempa\Theta\@tempa\Lambda\@tempa\Xi
59.86 % \@tempa\Pi\@tempa\Sigma\@tempa\Upsilon\@tempa\Phi\@tempa\Psi
59.87 % \@tempa\Omega
59.88 % % some accents:
59.89 % \def\@tempa#1#2\@nil{\def\@tempc{#1}}\def\@tempb{\mathaccent}
59.90 % \expandafter\@tempa\hat\relax\relax\@nil
59.91 % \ifx\@tempb\@tempc
59.92 % \def\@tempa#1\@nil{#1}%
59.93 % \def\@tempb#1{\afterassignment\@tempa\mathchardef\@tempc}%
59.94 % \def\do#1"#2{
59.95 % \def\@tempd#1{\expandafter\@tempb#1\@nil
59.96 % \ifnum\@tempc>"FFF
59.97 % \xdef#1{\mathaccent"\expandafter\do\meaning\@tempc\space}%
59.98 % \fi}
59.99 % \@tempd\hat\@tempd\check\@tempd\tilde\@tempd\acute\@tempd\grave
59.100 % \@tempd\dot\@tempd\ddot\@tempd\breve\@tempd\bar
59.101 % \fi
59.102 % \endgroup

```

The user must use the `inputenc` package when any 8-bit Cyrillic font encoding is used, selecting one of the Cyrillic input encodings. We do not assume any default input encoding, so the user should explicitly call the `inputenc` package by `\usepackage{inputenc}`. We also removed `\AtBeginDocument`, so `inputenc` should be used before `babel`.

```

59.103 \@ifpackageloaded{inputenc}{}{%
59.104 \def\reserved@a{LWN}%
59.105 \ifx\reserved@a\cyrillicencoding\else
59.106 \def\reserved@a{OT2}%
59.107 \ifx\reserved@a\cyrillicencoding\else
59.108 \PackageWarning{babel}%
59.109 {No input encoding specified for Ukrainian language}
59.110 \fi\fi}

```

Now we define two commands that offer the possibility to switch between Cyrillic and Roman encodings.

`\cyrillictext` The command `\cyrillictext` will switch from Latin font encoding to the Cyrillic font encoding, the command `\latintext` switches back. This assumes that the ‘normal’ font encoding is a Latin one. These commands are *declarations*, for shorter peaces of text the commands `\textlatin` and `\textcyrillic` can be used.

```

59.111 %\DeclareRobustCommand{\latintext}{%
59.112 % \fontencoding{\latinencoding}\selectfont

```

```

59.113 % \def\encodingdefault{\latinencoding}}
59.114 \let\lat\latintext

```

`\textcyrillic` These commands take an argument which is then typeset using the requested font encoding.

```

59.115 \DeclareTextFontCommand{\textcyrillic}{\cyrillictext}
59.116 %\DeclareTextFontCommand{\textlatin}{\latintext}

```

We make the T_EX

```

59.117 %\ifx\ltxTeX\undefined\let\ltxTeX\TeX\fi
59.118 %\ProvideTextCommandDefault{\TeX}{\textlatin{\ltxTeX}}

```

and L^AT_EX logos encoding independent.

```

59.119 %\ifx\ltxLaTeX\undefined\let\ltxLaTeX\LaTeX\fi
59.120 %\ProvideTextCommandDefault{\LaTeX}{\textlatin{\ltxLaTeX}}

```

The next step consists of defining commands to switch to (and from) the Ukrainian language.

`\captionsukrainian` The macro `\captionsukrainian` defines all strings used in the four standard document classes provided with L^AT_EX. The two commands `\cyr` and `\lat` activate Cyrillic resp. Latin encoding.

```

59.121 \addto\captionsukrainian{%
59.122   \def\prefacename{{\cyr\CYRV\cyrs\cyrt\cyru\cyrp}}}%
59.123 % \def\prefacename{{\cyr\CYRP\cyre\cyrr\cyre\cyrd\cyrm\cyro\cyrv\cyra}}}%
59.124   \def\refname{%
59.125     {\cyr\CYRL\cyrii\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}}%
59.126 %   \def\refname{%
59.127 %     {\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
59.128 %       \ \cyrp\cyro\cyrs\cyri\cyrl\cyra\cyrn\cyrstsn}}}%
59.129   \def\abstractname{%
59.130     {\cyr\CYRA\cyrn\cyro\cyrt\cyra\cyrc\cyrii\cyrya}}}%
59.131 %   \def\abstractname{{\cyr\CYRR\cyre\cyrf\cyre\cyrr\cyra\cyrt}}}%
59.132   \def\bibname{%
59.133     {\cyr\CYRB\cyrii\cyrb\cyrl\cyrii\cyro\cyrgup\cyrr\cyra\cyrf\cyrii\cyrya}}}%
59.134 % \def\bibname{{\cyr\CYRL\cyrii\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}}%
59.135   \def\chaptername{{\cyr\CYRR\cyro\cyrz\cyrd\cyrii\cyrl}}}%
59.136 %   \def\chaptername{{\cyr\CYRG\cyrl\cyra\cyrv\cyra}}}%
59.137   \def\appendixname{{\cyr\CYRD\cyro\cyrd\cyra\cyrt\cyro\cyrk}}}%
59.138   \def\contentsname{{\cyr\CYRZ\cyrm\cyrii\cyrs\cyrt}}}%
59.139   \def\listfigurename{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
59.140     \ \cyrii\cyrl\cyru\cyrs\cyrt\cyrr\cyra\cyrc\cyrii\cyrishrt}}}%
59.141   \def\listtablename{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
59.142     \ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyrstsn}}}%
59.143   \def\indexname{{\cyr\CYRP\cyro\cyrk\cyra\cyrz\cyrch\cyri\cyrk}}}%
59.144   \def\authorname{{\cyr\CYRII\cyrm\cyre\cyrn\cyrn\cyri\cyrishrt
59.145     \ \cyrp\cyro\cyrk\cyra\cyrz\cyrch\cyri\cyrk}}}%
59.146   \def\figurename{{\cyr\CYRR\cyri\cyrs.}}}%
59.147 %   \def\figurename{{\cyr\CYRR\cyri\cyrs\cyru\cyrn\cyro\cyrk}}}%
59.148   \def\tablename{{\cyr\CYRT\cyra\cyrb\cyrl.}}}%
59.149 %   \def\tablename{{\cyr\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyrya}}}%
59.150   \def\partname{{\cyr\CYRCH\cyra\cyrs\cyrt\cyri\cyrn\cyra}}}%
59.151   \def\enclname{{\cyr\cyrv\cyrk\cyrl\cyra\cyrd\cyrk\cyra}}}%
59.152   \def\ccname{{\cyr\cyrk\cyro\cyrp\cyrii\cyrya}}}%
59.153   \def\headtoname{{\cyr\CYRD\cyro}}}%
59.154   \def\pagename{{\cyr\cyrs.}}}%
59.155 %   \def\pagename{{\cyr\cyrs\cyrt\cyro\cyrr\cyrii\cyrn\cyrk\cyra}}}%
59.156   \def\seename{{\cyr\cyrd\cyri\cyrv.}}}%
59.157   \def\alsoname{{\cyr\cyrd\cyri\cyrv.\ \cyrt\cyra\cyrk\cyro\cyrz}}}%
59.158   \def\proofname{{\cyr\CYRD\cyro\cyrv\cyre\cyrd\cyre\cyrn\cyrn\cyrya}}}%
59.159   \def\glossaryname{{\cyr\CYRS\cyrl\cyro\cyrv\cyrn\cyri\cyrk\
59.160     \ \cyrt\cyre\cyrr\cyrm\cyrii\cyrn\cyrii\cyrv}}}%
59.161 }

```

`\dateukrainian` The macro `\dateukrainian` redefines the command `\today` to produce Ukrainian dates.

```

59.162 \def\dateukrainian{%
59.163   \def\today{\number\day~\ifcase\month\or
59.164     \cyr\cyrii\cyrch\cyrn\cyrya\or
59.165     \cyr\cyryu\cyrt\cyro\cyrg\cyro\or
59.166     \cyrb\cyre\cyrr\cyre\cyrz\cyrn\cyrya\or
59.167     \cyrk\cyrv\cyrii\cyrt\cyrn\cyrya\or
59.168     \cyrt\cyrr\cyra\cyrv\cyrn\cyrya\or
59.169     \cyrch\cyre\cyrr\cyrv\cyrn\cyrya\or
59.170     \cyr\cyri\cyrp\cyrn\cyrya\or
59.171     \cyr\cyre\cyrr\cyrp\cyrn\cyrya\or
59.172     \cyrv\cyre\cyrr\cyre\cyr\cyrn\cyrya\or
59.173     \cyrr\cyro\cyrv\cyrt\cyrn\cyrya\or
59.174     \cyr\cyri\cyr\cyrt\cyro\cyrp\cyra\cyrd\cyra\or
59.175     \cyrg\cyrr\cyru\cyrd\cyrn\cyrya\fi
59.176   \space\number\year~\cyr.}}

```

`\extrasukrainian` The macro `\extrasukrainian` will perform all the extra definitions needed for the Ukrainian language. The macro `\noextrasukrainian` is used to cancel the actions of `\extrasukrainian`.

The first action we define is to switch on the selected Cyrillic encoding whenever we enter ‘ukrainian’.

```

59.177 \addto\extrasukrainian{\cyrillictext}

```

When the encoding definition file was processed by \LaTeX the current font encoding is stored in `\latinencoding`, assuming that \LaTeX uses T1 or OT1 as default. Therefore we switch back to `\latinencoding` whenever the Ukrainian language is no longer ‘active’.

```

59.178 \addto\noextrasukrainian{\latinintext}

```

Next we must allow hyphenation in the Ukrainian words with apostrophe whenever we enter ‘ukrainian’. This solution was proposed by Vladimir Volovich <vvv@vvv.vsu.ru>

```

59.179 \addto\extrasukrainian{\lccode‘\’=‘\’}
59.180 \addto\noextrasukrainian{\lccode‘\’=0}

```

`\verbatim@font` In order to get both Latin and Cyrillic letters in verbatim text we need to change the definition of an internal \LaTeX command somewhat:

```

59.181 %\def\verbatim@font{%
59.182 %  \let\encodingdefault\latinencoding
59.183 %  \normalfont\ttfamily
59.184 %  \expandafter\def\csname\cyrillicencoding-cmd\endcsname##1##2{%
59.185 %    \ifx\protect\@typeset@protect
59.186 %      \begingroup\UseTextSymbol\cyrillicencoding##1\endgroup
59.187 %    \else\noexpand##1\fi}}

```

The category code of the characters ‘:’, ‘;’, ‘!’, and ‘?’ is made `\active` to insert a little white space.

For Ukrainian (as well as for Russian and German) the " character also is made active.

Note: It is *very* questionable whether the Russian typesetting tradition requires additional spacing before those punctuation signs. Therefore, we make the corresponding code optional. If you need it, then define the `frenchpunct` docstrip option in `babel.ins`.

Borrowed from french. Some users dislike automatic insertion of a space before ‘double punctuation’, and prefer to decide themselves whether a space should be added or not; so a hook `\NoAutoSpaceBeforeFDP` is provided: if this command is added (in file `ukraineb.cfg`, or anywhere in a document) `ukraineb` will respect

your typing, and introduce a suitable space before ‘double punctuation’ *if and only if* a space is typed in the source file before those signs.

The command `\AutoSpaceBeforeFDP` switches back to the default behavior of `ukraineb`.

```
59.188 (*frenchpunct)
59.189 \initiate@active@char{:}
59.190 \initiate@active@char{;}
59.191 /frenchpunct
59.192 (*frenchpunct | spanishlig)
59.193 \initiate@active@char{!}
59.194 \initiate@active@char{?}
59.195 /frenchpunct | spanishlig)
59.196 \initiate@active@char{"}
```

The code above is necessary because we need extra active characters. The character " is used as indicated in table 34.

We specify that the Ukrainian group of shorthands should be used.

```
59.197 \addto\extrasukrainian{\languageshorthands{ukrainian}}
```

These characters are ‘turned on’ once, later their definition may vary.

```
59.198 \addto\extrasukrainian{%
59.199 (frenchpunct) \bbl@activate{:}\bbl@activate{;}%
59.200 (frenchpunct | spanishlig) \bbl@activate{!}\bbl@activate{?}%
59.201 \bbl@activate{"}}
59.202 \addto\noextrasukrainian{%
59.203 (frenchpunct) \bbl@deactivate{:}\bbl@deactivate{;}%
59.204 (frenchpunct | spanishlig) \bbl@deactivate{!}\bbl@deactivate{?}%
59.205 \bbl@deactivate{"}}
```

The X2 and T2* encodings do not contain `spanish_shriek` and `spanish_query` symbols; as a consequence, the ligatures ‘?’ and ‘!’ do not work with them (these characters are useless for Cyrillic texts anyway). But we define the shorthands to emulate these ligatures (optionally).

We do not use `\latinencoding` here (but instead explicitly use OT1) because the user may choose T2A to be the primary encoding, but it does not contain these characters.

```
59.206 (*spanishlig)
59.207 \declare@shorthand{ukrainian}{?'}{\UseTextSymbol{OT1}\textquestiondown}
59.208 \declare@shorthand{ukrainian}{!''}{\UseTextSymbol{OT1}\textexclamdown}
59.209 /spanishlig)
```

`\ukrainian@sh@;` We have to reduce the amount of white space before ;, : and !. This should only happen in horizontal mode, hence the test with `\ifhmode`.

```
\ukrainian@sh@!@59.210 (*frenchpunct)
\ukrainian@sh@?@59.211 \declare@shorthand{ukrainian}{;};}%
59.212 \ifhmode
```

In horizontal mode we check for the presence of a ‘space’, ‘unskip’ if it exists and place a 0.1em kerning.

```
59.213 \ifdim\lastskip>\z@
59.214 \unskip\nobreak\kern.1em
59.215 \else
```

If no space has been typed, we add `\FDP@thinspace` which will be defined, up to the user’s wishes, as an automatic added thinspace, or as `\@empty`.

```
59.216 \FDP@thinspace
59.217 \fi
59.218 \fi
```

Now we can insert a ‘;’ character.

```
59.219 \string;}
```


The other definitions are very similar.

```

59.220 \declare@shorthand{ukrainian}{:}{}%
59.221   \ifhmode
59.222     \ifdim\lastskip>\z@
59.223       \unskip\nobreak\kern.1em
59.224     \else
59.225       \FDP@thinspace
59.226     \fi
59.227   \fi
59.228   \string:}

59.229 \declare@shorthand{ukrainian}{!}{}%
59.230   \ifhmode
59.231     \ifdim\lastskip>\z@
59.232       \unskip\nobreak\kern.1em
59.233     \else
59.234       \FDP@thinspace
59.235     \fi
59.236   \fi
59.237   \string!}

59.238 \declare@shorthand{ukrainian}{?}{}%
59.239   \ifhmode
59.240     \ifdim\lastskip>\z@
59.241       \unskip\nobreak\kern.1em
59.242     \else
59.243       \FDP@thinspace
59.244     \fi
59.245   \fi
59.246   \string?}

```

`\AutoSpaceBeforeFDP` `\FDP@thinspace` is defined as unbreakable spaces if `\AutoSpaceBeforeFDP` is activated or as `\@empty` if `\NoAutoSpaceBeforeFDP` is in use. The default is `\FDP@thinspace` `\AutoSpaceBeforeFDP`.

```

59.247 \def\AutoSpaceBeforeFDP{%
59.248   \def\FDP@thinspace{\nobreak\kern.1em}}
59.249 \def\NoAutoSpaceBeforeFDP{\let\FDP@thinspace\@empty}
59.250 \AutoSpaceBeforeFDP

```

`\FDPon` The next macros allow to switch on/off activeness of double punctuation signs.

```

\FDPoff59.251 \def\FDPon{\bbl@activate{}}%
59.252   \bbl@activate{;}%
59.253   \bbl@activate{?}%
59.254   \bbl@activate{!}}
59.255 \def\FDPoff{\bbl@deactivate{}}%
59.256   \bbl@deactivate{;}%
59.257   \bbl@deactivate{?}%
59.258   \bbl@deactivate{!}}

```

`\system@sh@:` When the active characters appear in an environment where their Ukrainian behaviour is not wanted they should give an ‘expected’ result. Therefore we define `\system@sh@!` shorthands at system level as well.

```

\system@sh@;59.259 \declare@shorthand{system}{:}{\string:}
59.260 \declare@shorthand{system}{;}{\string;}
59.261 \frenchpunct
59.262 (*frenchpunct&!spanishlig)
59.263 \declare@shorthand{system}{!}{\string!}
59.264 \declare@shorthand{system}{?}{\string?}
59.265 \frenchpunct&!spanishlig)

```

To be able to define the function of ‘”’, we first define a couple of ‘support’ macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`.

```
59.266 \begingroup \catcode'\''12
59.267 \def\reserved@a{\endgroup
59.268 \def\@SS{\mathchar"7019 }
59.269 \def\dq{"}}
59.270 \reserved@a
```

Now we can define the doublequote macros: german and french quotes. We use definitions of these quotes made in `babel.sty`. The french quotes are contained in the T2* encodings.

```
59.271 \declare@shorthand{ukrainian}{''}{\glqq}
59.272 \declare@shorthand{ukrainian}{'''}{\grqq}
59.273 \declare@shorthand{ukrainian}{"<"}{\flqq}
59.274 \declare@shorthand{ukrainian}{">"}{\frqq}
```

Some additional commands:

```
59.275 \declare@shorthand{ukrainian}{""}{\hskip\z@skip}
59.276 \declare@shorthand{ukrainian}{""~}{\textormath{\leavevmode\hbox{-}}{-}}
59.277 \declare@shorthand{ukrainian}{""=}{\nobreak-\hskip\z@skip}
59.278 \declare@shorthand{ukrainian}{""|}{%
59.279 \textormath{\nobreak\discretionary{-}{-}{\kern.03em}%
59.280 \allowhyphens}{-}}}
```

The next two macros for `"-` and `---` are somewhat different. We must check whether the second token is a hyphen character:

```
59.281 \declare@shorthand{ukrainian}{"-}{%
```

If the next token is `'-'`, we typeset an emdash, otherwise a hyphen sign:

```
59.282 \def\ukrainian@sh@tmp{%
59.283 \if\ukrainian@sh@next-\expandafter\ukrainian@sh@emdash
59.284 \else\expandafter\ukrainian@sh@hyphen\fi
59.285 }%
```

TeX looks for the next token after the first `'-'`: the meaning of this token is written to `\ukrainian@sh@next` and `\ukrainian@sh@tmp` is called.

```
59.286 \futurelet\ukrainian@sh@next\ukrainian@sh@tmp}
```

Here are the definitions of hyphen and emdash. First the hyphen:

```
59.287 \def\ukrainian@sh@hyphen{%
59.288 \nobreak-\bbl@allowhyphens}
```

For the emdash definition, there are the two parameters: we must `'eat'` two last hyphen signs of our emdash...

```
59.289 \def\ukrainian@sh@emdash#1#2{\cdash-#1#2}
```

`\cdash` ... these two parameters are useful for another macro: `\cdash`:

```
59.290 %\ifx\cdash\undefined % should be defined earlier
59.291 \def\cdash#1#2#3{\def\tempx@{#3}%
59.292 \def\tempa@{-}\def\tempb@{~}\def\tempc@{*}%
59.293 \ifx\tempx@\tempa@\@Acdash\else
59.294 \ifx\tempx@\tempb@\@Bcdash\else
59.295 \ifx\tempx@\tempc@\@Ccdash\else
59.296 \errmessage{Wrong usage of cdash}\fi\fi\fi}
```

second parameter (or third for `\cdash`) shows what kind of emdash to create in next step

"--- ordinary (plain) Cyrillic emdash inside text: an unbreakable thinspace will be inserted before only in case of a *space* before the dash (it is necessary for dashes after display maths formulae: there could be lists, enumerations etc. started with “— where *a* is ...” i.e., the dash starts a line). (Firstly there were planned rather soft rules for user: he may put a space before the dash or not. But it is difficult to place this thinspace automatically, i.e., by checking modes because after display formulae T_EX uses horizontal mode. Maybe there is a misunderstanding? Maybe there is another way?) After a dash a breakable thinspace is always placed;

```
59.297 % What is more grammatically: .2em or .2\fontdimen6\font ?
59.298 \def\@Acdash{\ifdim\lastskip>\z@\unskip\nobreak\hskip.2em\fi
59.299 \cyrdash\hskip.2em\ignorespaces}%
```

--~ emdash in compound names or surnames (like Mendeleev–Klapeiron); this dash has no space characters around; after the dash some space is added
`\exhyphenalty`

```
59.300 \def\@Bcdash{\leavevmode\ifdim\lastskip>\z@\unskip\fi
59.301 \nobreak\cyrdash\penalty\exhyphenpenalty\hskip\z@skip\ignorespaces}%
```

--* for denoting direct speech (a space like `\enskip` must follow the emdash);

```
59.302 \def\@Ccdash{\leavevmode
59.303 \nobreak\cyrdash\nobreak\hskip.35em\ignorespaces}%
59.304 %\fi
```

`\cyrdash` Finally the macro for “body” of the Cyrillic emdash. The `\cyrdash` macro will be defined in case this macro hasn’t been defined in a fontenc file. For T2* fonts, `cyrdash` will be placed in the code of the English emdash thus it uses ligature ---.

```
59.305 % Is there an IF necessary?
59.306 \ifx\cyrdash\undefined
59.307 \def\cyrdash{\hbox to.8em{--\hss--}}
59.308 \fi
```

Here a really new macro—to place thinspace between initials. This macro used instead of `\,` allows hyphenation in the following surname.

```
59.309 \declare@shorthand{ukrainian}{",}{\nobreak\hskip.2em\ignorespaces}
```

`\mdqon` All that’s left to do now is to define a couple of commands for “.

```
\mdqoff59.310 \def\mdqon{\bbl@activate{}}
59.311 \def\mdqoff{\bbl@deactivate{}}
```

The Ukrainian hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
59.312 \providehyphenmins{\CurrentOption}{\tw@\tw@}
59.313 % temporary hack:
59.314 \ifx\englishhyphenmins\undefined
59.315 \def\englishhyphenmins{\tw@\thr@@}
59.316 \fi
```

Now the action `\extrasukrainian` has to execute is to make sure that the command `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasukrainian` will switch it off again.

```
59.317 \addto\extrasukrainian{\bbl@frenchspacing}
59.318 \addto\noextrasukrainian{\bbl@nonfrenchspacing}
```

Next we add a new enumeration style for Ukrainian manuscripts with Cyrillic letters, and later on we define some math operator names in accordance with Ukrainian and Russian typesetting traditions.

`\Asbuk` We begin by defining `\Asbuk` which works like `\Alph`, but produces (uppercase) Cyrillic letters instead of Latin ones. The letters CYRGUP, and SFTSN are skipped, as usual for such enumeration.

```

59.319 \def\Asbuk#1{\expandafter\@Asbuk\csname c@#1\endcsname}
59.320 \def\@Asbuk#1{\ifcase#1\or
59.321   \CYRA\or\CYRB\or\CYRV\or\CYRG\or\CYRD\or\CYRE\or\CYRIE\or
59.322   \CYRZH\or\CYRZ\or\CYRI\or\CYRII\or\CYRYI\or\CYRISHRT\or
59.323   \CYRK\or\CYRL\or\CYRM\or\CYRN\or\CYRO\or\CYRP\or\CYRR\or
59.324   \CYRS\or\CYRT\or\CYRU\or\CYRF\or\CYRH\or\CYRC\or\CYRCH\or
59.325   \CYRSH\or\CYRSHCH\or\CYRYU\or\CYRYA\else\@ctrerr\fi}

```

`\asbuk` The macro `\asbuk` is similar to `\alph`; it produces lowercase Ukrainian letters.

```

59.326 \def\asbuk#1{\expandafter\@asbuk\csname c@#1\endcsname}
59.327 \def\@asbuk#1{\ifcase#1\or
59.328   \cyra\or\cyrb\or\cyrv\or\cyrg\or\cyrd\or\cyre\or\cyrie\or
59.329   \cyrzh\or\cyrz\or\cyri\or\cyrii\or\cyryi\or\cyrishrt\or
59.330   \cyrk\or\cyrl\or\cyrm\or\cyrn\or\cyro\or\cyrp\or\cyrr\or
59.331   \cyrs\or\cyrt\or\cyru\or\cyrf\or\cyrh\or\cyrc\or\cyrch\or
59.332   \cyrsh\or\cyrshch\or\cyryu\or\cyrya\else\@ctrerr\fi}

```

Set up default Cyrillic math alphabets. The math groups for Cyrillic letters are defined in the encoding definition files. First, declare a new alphabet for symbols, `\cyrmathrm`, based on the symbol font for Cyrillic letters defined in the encoding definition file. Note, that by default Cyrillic letters are taken from upright font in math mode (unlike Latin letters).

```

59.333 %\RequirePackage{textmath}
59.334 \@ifundefined{sym\cyrillicencoding letters}{\{%
59.335   \SetSymbolFont{cyrillicencoding letters}{bold}\cyrillicencoding
59.336   \rmdefault\bfdefault\updefault
59.337   \DeclareSymbolFontAlphabet\cyrmathrm{\cyrillicencoding letters}

```

And we need a few commands to be able to switch to different variants.

```

59.338 \DeclareMathAlphabet\cyrmathbf\cyrillicencoding
59.339   \rmdefault\bfdefault\updefault
59.340 \DeclareMathAlphabet\cyrmathsf\cyrillicencoding
59.341   \sfdefault\mddefault\updefault
59.342 \DeclareMathAlphabet\cyrmathit\cyrillicencoding
59.343   \rmdefault\mddefault\itdefault
59.344 \DeclareMathAlphabet\cyrmathtt\cyrillicencoding
59.345   \ttdefault\mddefault\updefault
59.346 %
59.347 \SetMathAlphabet\cyrmathsf{bold}\cyrillicencoding
59.348   \sfdefault\bfdefault\updefault
59.349 \SetMathAlphabet\cyrmathit{bold}\cyrillicencoding
59.350   \rmdefault\bfdefault\itdefault
59.351 }

```

Some math functions in Ukrainian and Russian math books have other names: e.g., `\sinh` in Russian is written as `sh` etc. So we define a number of new math operators.

`\sinh`:

```

59.352 \def\sh{\mathop{\operator@font sh}\nolimits}

```

`\cosh`:

```

59.353 \def\ch{\mathop{\operator@font ch}\nolimits}

```

`\tan`:

```

59.354 \def\tg{\mathop{\operator@font tg}\nolimits}

```

`\arctan`:

```

59.355 \def\arctg{\mathop{\operator@font arctg}\nolimits}

```

`arcctg`:

```

59.356 \def\arcctg{\mathop{\operator@font arcctg}\nolimits}

```

The following macro conflicts with `\th` defined in Latin 1 encoding:

```

\tanh:
59.357 \addto\extrasrussian{%
59.358   \babel@save{\th}%
59.359   \let\ltx@th\th
59.360   \def\th{\textormath{\ltx@th}%
59.361     {\mathop{\operator@font th}\nolimits}}%
59.362 }

```

`\cot:`

```

59.363 \def\ctg{\mathop{\operator@font ctg}\nolimits}

```

`\coth:`

```

59.364 \def\cth{\mathop{\operator@font cth}\nolimits}

```

`\csc:`

```

59.365 \def\cosec{\mathop{\operator@font cosec}\nolimits}

```

And finally some other Ukrainian and Russian mathematical symbols:

```

59.366 \def\Prob{\mathop{\kern\z@\mathsf{P}}\nolimits}
59.367 \def\Variance{\mathop{\kern\z@\mathsf{D}}\nolimits}
59.368 \def\nsd{\mathop{\cyrmathrm{\cyrn.\cyrs.\cyrd.}}\nolimits}
59.369 \def\nsk{\mathop{\cyrmathrm{\cyrn.\cyrs.\cyrk.}}\nolimits}
59.370 \def\NSD{\mathop{\cyrmathrm{\CYRN\CYRS\CYRD}}\nolimits}
59.371 \def\NSK{\mathop{\cyrmathrm{\CYRN\CYRS\CYRK}}\nolimits}
59.372   \def\nod{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrd.}}\nolimits} % ??????
59.373   \def\nok{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrk.}}\nolimits} % ??????
59.374   \def\NOD{\mathop{\cyrmathrm{\CYRN\CYRO\CYRD}}\nolimits} % ??????
59.375   \def\NOK{\mathop{\cyrmathrm{\CYRN\CYRO\CYRK}}\nolimits} % ??????
59.376 \def\Proj{\mathop{\cyrmathrm{\CYRP\cyrr}}\nolimits}

```

This is for compatibility with older Ukrainian packages.

```

59.377 \DeclareRobustCommand{\No}{%
59.378   \ifmmode{\nfss@text{\textnumero}}\else\textnumero\fi}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

59.379 \ldf@finish{ukrainian}
59.380 \code

```

60 The Lower Sorbian language

The file `lsorbian.dtx`⁷⁰ It defines all the language-specific macros for Lower Sorbian.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
60.1 (*code)
60.2 \LdfInit\CurrentOption{date\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `lsorbian` will be an ‘unknown’ language, in which case we have to make it known. So we check for the existence of `\l@lsorbian` to see whether we have to do something here. As `babel` also knows the option `lowersorbian` we have to check that as well.

```
60.3 \ifx\l@lowersorbian\@undefined
60.4   \ifx\l@lsorbian\@undefined
60.5     \nopatterns{Lsorbian}
60.6     \adddialect\l@lsorbian\z@
60.7     \let\l@lowersorbian\l@lsorbian
60.8   \else
60.9     \let\l@lowersorbian\l@lsorbian
60.10  \fi
60.11 \else
60.12   \let\l@lsorbian\l@lowersorbian
60.13 \fi
```

The next step consists of defining commands to switch to (and from) the Lower Sorbian language.

`\captionslsorbian` The macro `\captionslsorbian` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
60.14 \@namedef{captions\CurrentOption}{%
60.15   \def\prefacename{Zawod}%
60.16   \def\refname{Referency}%
60.17   \def\abstractname{Abstrakt}%
60.18   \def\bibname{Literatura}%
60.19   \def\chaptername{Kapitl}%
60.20   \def\appendixname{Dodawki}%
60.21   \def\contentsname{Wop\’simje\’se}%
60.22   \def\listfigurename{Zapis wobrazow}%
60.23   \def\listtablename{Zapis tabulkow}%
60.24   \def\indexname{Indeks}%
60.25   \def\figurename{Wobraz}%
60.26   \def\tablename{Tabulka}%
60.27   \def\partname{\’Z\’v el}%
60.28   \def\enclname{P\’si\’l oga}%
60.29   \def\ccname{CC}%
60.30   \def\headtoname{Komu}%
60.31   \def\pagename{Strona}%
60.32   \def\seename{gl.}%
60.33   \def\alsoname{gl.~teke}%
60.34   \def\proofname{Proof}% <-- needs translation
60.35   \def\glossaryname{Glossary}% <-- Needs translation
60.36 }%
```

`\newdatelsorbian` The macro `\newdatelsorbian` redefines the command `\today` to produce Lower Sorbian dates.

```
60.37 \@namedef{newdate\CurrentOption}{%
60.38   \def\today{\number\day.\~\ifcase\month\or
```

⁷⁰The file described in this section has version number v1.0g and was last revised on 2008/03/17. It was written by Eduard Werner (edi@kaihh.hanse.de).

```

60.39     januara\or februara\or m\ v erca\or apryla\or maja\or
60.40     junija\or julija\or awgusta\or septembra\or oktobra\or
60.41     nowembra\or decembra\fi
60.42     \space \number\year}}

```

`\olddatelsorbian` The macro `\olddatelsorbian` redefines the command `\today` to produce old-style Lower Sorbian dates.

```

60.43 \namedef{olddate\CurrentOption}{%
60.44   \def\today{\number\day.\~\ifcase\month\or
60.45     wjelikego ro\ v zka\or
60.46     ma\l ego ro\ v zka\or
60.47     nal\ v etnika\or
60.48     jat\ v sownika\or
60.49     ro\ v zownika\or
60.50     sma\ v znika\or
60.51     pra\ v znika\or
60.52     \v znje\ 'nca\or
60.53     po\ v znje\ 'nca\or
60.54     winowca\or
60.55     nazymnika\or
60.56     godownika\fi \space \number\year}}

```

The default will be the new-style dates.

```

60.57 \expandafter\let\csname date\CurrentOption\expandafter\endcsname
60.58       \csname newdate\CurrentOption\endcsname

```

`\extraslsorbian` The macro `\extraslsorbian` will perform all the extra definitions needed for the Sorbian language. The macro `\noextraslsorbian` is used to cancel the actions of `\extraslsorbian`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```

60.59 \namedef{extras\CurrentOption}{}
60.60 \namedef{noextras\CurrentOption}{}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

60.61 \ldf@finish\CurrentOption
60.62 \code

```

61 The Upper Sorbian language

The file `usorbian.dtx`⁷¹ It defines all the language-specific macros for Upper Sorbian.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
61.1 (*code)
61.2 \LdfInit\CurrentOption{date\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `usorbian` will be an ‘unknown’ language, in which case we have to make it known. So we check for the existence of `\l@usorbian` to see whether we have to do something here. As `babel` also knows the option `uppersorbian` we have to check that as well.

```
61.3 \ifx\l@uppersorbian\@undefined
61.4   \ifx\l@usorbian\@undefined
61.5     \nopatterns{Usorbian}
61.6     \adddialect\l@usorbian\z@
61.7     \let\l@uppersorbian\l@usorbian
61.8   \else
61.9     \let\l@uppersorbian\l@usorbian
61.10  \fi
61.11 \else
61.12   \let\l@usorbian\l@uppersorbian
61.13 \fi
```

The next step consists of defining commands to switch to (and from) the Upper Sorbian language.

`\captionsusorbian` The macro `\captionsusorbian` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
61.14 \@namedef{captions\CurrentOption}{%
61.15   \def\prefacename{Zawod}%
61.16   \def\refname{Referency}%
61.17   \def\abstractname{Abstrakt}%
61.18   \def\bibname{Literatura}%
61.19   \def\chaptername{Kapitl}%
61.20   \def\appendixname{Dodawki}%
61.21   \def\contentsname{Wobsah}%
61.22   \def\listfigurename{Zapis wobrazow}%
61.23   \def\listtablename{Zapis tabulkow}%
61.24   \def\indexname{Indeks}%
61.25   \def\figurename{Wobraz}%
61.26   \def\tablename{Tabulka}%
61.27   \def\partname{D\`z\`v el}%
61.28   \def\enclname{P\`v r\`l oha}%
61.29   \def\ccname{CC}%
61.30   \def\headtoname{Komu}%
61.31   \def\pagename{Strona}%
61.32   \def\seename{hl.}%
61.33   \def\alsoname{hl.~te\`v z}
61.34   \def\proofname{Proof}% <-- needs translation
61.35   \def\glossaryname{Glossary}% <-- Needs translation
61.36 }%
```

`\newdateusorbian` The macro `\newdateusorbian` redefines the command `\today` to produce Upper Sorbian dates.

```
61.37 \@namedef{newdate\CurrentOption}{%
61.38   \def\today{\number\day.\~\ifcase\month\or
```

⁷¹The file described in this section has version number v1.0k and was last revised on 2008/03/17. It was written by Eduard Werner (edi@kaihh.hanse.de).


```

61.39   januara\or februara\or m\ v erca\or apryla\or meje\or junija\or
61.40   julija\or awgusta\or septembra\or oktobra\or
61.41   nowembra\or decembra\fi
61.42   \space \number\year}}

```

`\olddateusorbian` The macro `\olddateusorbian` redefines the command `\today` to produce old-style Upper Sorbian dates.

```

61.43 \namedef{olddate\CurrentOption}{%
61.44   \def\today{\number\day.\~\ifcase\month\or
61.45     wulkeho r\ 'o\ v zka\or ma\ l eho r\ 'o\ v zka\or nal\ v etnika\or
61.46     jutrownika\or r\ 'o\ v zownika\or sma\ v znika\or pra\ v znika\or
61.47     \v znjenca\or po\ v znjenca\or winowca\or nazymnika\or
61.48     hodownika\fi \space \number\year}}

```

The default will be the new-style dates.

```

61.49 \expandafter\let\csname date\CurrentOption\expandafter\endcsname
61.50       \csname newdate\CurrentOption\endcsname

```

`\extrasusorbian` The macro `\extrasusorbian` will perform all the extra definitions needed for the Upper Sorbian language. It's pirated from `germanb.sty`. The macro `\noextrasusorbian` is used to cancel the actions of `\extrasusorbian`.

Because for Upper Sorbian (as well as for Dutch) the " character is made active. This is done once, later on its definition may vary.

```

61.51 \initiate@active@char{"}
61.52 \namedef{extras\CurrentOption}{\languageshorthands{usorbian}}
61.53 \expandafter\addto\csname extras\CurrentOption\endcsname{%
61.54   \bbl@activate{"}}

```

Don't forget to turn the shorthands off again.

```

61.55 \expandafter\addto\csname extras\CurrentOption\endcsname{%
61.56   \bbl@deactivate{"}}

```

In order for \TeX to be able to hyphenate German Upper Sorbian words which contain 'k' we have to give the character a nonzero `\lccode` (see Appendix H, the \TeX book). As some of the other language definitions turn the character \sim into a shorthand we need to make sure that it has its original definition here.

```

61.57 \begingroup \catcode'\~7
61.58 \def\x{\endgroup
61.59   \expandafter\addto\csname extras\CurrentOption\endcsname{%
61.60     \babel@savevariable{\lccode'\~Y}%
61.61     \lccode'\~Y'\~Y}}
61.62 \x

```

The umlaut accent macro `\"` is changed to lower the umlaut dots. The redefinition is done with the help of `\umlautlow`.

```

61.63 \expandafter\addto\csname extras\CurrentOption\endcsname{%
61.64   \babel@save"\umlautlow}
61.65 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
61.66   \umlauthigh}

```

The Upper Sorbian hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```

61.67 \providehyphenmins{\CurrentOption}{\tw@ \tw@}

```

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`. Also we store the original meaning of the command `\"` for future use.

```

61.68 \begingroup \catcode'\ "12
61.69 \def\x{\endgroup
61.70   \def\SS{\mathchar"7019 }
61.71   \def\dq{"}}
61.72 \x

```

Now we can define the doublequote macros: the umlauts,

```

61.73 \declare@shorthand{usorbian}{a}{\textormath{\{a\}\ddot a}}
61.74 \declare@shorthand{usorbian}{o}{\textormath{\{o\}\ddot o}}
61.75 \declare@shorthand{usorbian}{u}{\textormath{\{u\}\ddot u}}
61.76 \declare@shorthand{usorbian}{A}{\textormath{\{A\}\ddot A}}
61.77 \declare@shorthand{usorbian}{O}{\textormath{\{O\}\ddot O}}
61.78 \declare@shorthand{usorbian}{U}{\textormath{\{U\}\ddot U}}

tremas,

61.79 \declare@shorthand{usorbian}{e}{\textormath{\{e\}\ddot e}}
61.80 \declare@shorthand{usorbian}{E}{\textormath{\{E\}\ddot E}}
61.81 \declare@shorthand{usorbian}{i}{\textormath{\{i\}\ddot i\imath}}
61.82 \declare@shorthand{usorbian}{I}{\textormath{\{I\}\ddot I}}

usorbian es-zet (sharp s),

61.83 \declare@shorthand{usorbian}{s}{\textormath{\ss}}{\@SS}}
61.84 \declare@shorthand{usorbian}{S}{SS}

german and french quotes,

61.85 \declare@shorthand{usorbian}{‘}{%
61.86 \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
61.87 \declare@shorthand{usorbian}{’}{%
61.88 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
61.89 \declare@shorthand{usorbian}{<}{%
61.90 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
61.91 \declare@shorthand{usorbian}{>}{%
61.92 \textormath{\guillemotright}{\mbox{\guillemotright}}}

discretionary commands

61.93 \declare@shorthand{usorbian}{c}{\textormath{\bbl@disc ck}{c}}
61.94 \declare@shorthand{usorbian}{C}{\textormath{\bbl@disc CK}{C}}
61.95 \declare@shorthand{usorbian}{f}{\textormath{\bbl@disc f{ff}}{f}}
61.96 \declare@shorthand{usorbian}{F}{\textormath{\bbl@disc F{FF}}{F}}
61.97 \declare@shorthand{usorbian}{l}{\textormath{\bbl@disc l{ll}}{l}}
61.98 \declare@shorthand{usorbian}{L}{\textormath{\bbl@disc L{LL}}{L}}
61.99 \declare@shorthand{usorbian}{m}{\textormath{\bbl@disc m{mm}}{m}}
61.100 \declare@shorthand{usorbian}{M}{\textormath{\bbl@disc M{MM}}{M}}
61.101 \declare@shorthand{usorbian}{n}{\textormath{\bbl@disc n{nn}}{n}}
61.102 \declare@shorthand{usorbian}{N}{\textormath{\bbl@disc N{NN}}{N}}
61.103 \declare@shorthand{usorbian}{p}{\textormath{\bbl@disc p{pp}}{p}}
61.104 \declare@shorthand{usorbian}{P}{\textormath{\bbl@disc P{PP}}{P}}
61.105 \declare@shorthand{usorbian}{t}{\textormath{\bbl@disc t{tt}}{t}}
61.106 \declare@shorthand{usorbian}{T}{\textormath{\bbl@disc T{TT}}{T}}

and some additional commands:

61.107 \declare@shorthand{usorbian}{-}{\nobreak\-\bbl@allowhyphens}
61.108 \declare@shorthand{usorbian}{|}{%
61.109 \textormath{\nobreak\discretionary{-}{\kern.03em}%
61.110 \allowhyphens}{}}
61.111 \declare@shorthand{usorbian}{"}{\hskip\z@skip}

```

\mdqon All that’s left to do now is to define a couple of commands for reasons of compat-
\mdqoff ibility with german.sty.

```

\ck61.112 \def\mdqon{\shorthandon{}}
61.113 \def\mdqoff{\shorthandoff{}}
61.114 \def\ck{\allowhyphens\discretionary{k-}{k}{ck}\allowhyphens}

```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at \begin{document} and resetting the category code of @ to its original value.

```

61.115 \ldf@finish\CurrentOption
61.116 \code>

```

62 The Turkish language

The file `turkish.dtx`⁷² defines all the language definition macros for the Turkish language⁷³.

Turkish typographic rules specify that a little ‘white space’ should be added before the characters ‘:’, ‘!’ and ‘=’. In order to insert this white space automatically these characters are made ‘active’. Also `\frenhspace` is set.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
62.1 (*code)
62.2 \LdfInit{turkish}\captionsturkish
```

When this file is read as an option, i.e. by the `\usepackage` command, `turkish` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@turkish` to see whether we have to do something here.

```
62.3 \ifx\l@turkish\undefined
62.4 \nopatterns{Turkish}
62.5 \addialect\l@turkish0\fi
```

The next step consists of defining commands to switch to (and from) the Turkish language.

`\captionsturkish` The macro `\captionsturkish` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```
62.6 \addto\captionsturkish{%
62.7 \def\prefacename{"Ons"oz}%
62.8 \def\refname{Kaynaklar}%
62.9 \def\abstractname{"Ozet"%
62.10 \def\bibname{Kaynakca}%
62.11 \def\chaptername{Bölüm}%
62.12 \def\appendixname{Ek}%
62.13 \def\contentsname{İçindekiler}%
62.14 \def\listfigurename{Çekirdek Listesi}%
62.15 \def\listtablename{Tablo Listesi}%
62.16 \def\indexname{Dizin}%
62.17 \def\figurename{Çekirdek}%
62.18 \def\tablename{Tablo}%
62.19 \def\partname{Kısım}%
62.20 \def\enclname{İlişik}%
62.21 \def\ccname{Diğer Alıcılar}%
62.22 \def\headtoname{Alıcı}%
62.23 \def\pagename{Sayfa}%
62.24 \def\subjectname{İlgili}%
62.25 \def\seename{bkz.}%
62.26 \def\alsoname{ayrıca bkz.}%
62.27 \def\proofname{Kanıt}%
62.28 \def\glossaryname{Glossary}% <-- Needs translation
62.29 }%
```

`\dateturkish` The macro `\dateturkish` redefines the command `\today` to produce Turkish dates.

```
62.30 \def\dateturkish{%
62.31 \def\today{\number\day~\ifcase\month\or
62.32 Ocak\or \c Subat\or Mart\or Nisan\or Mayıs\or Haziran\or
62.33 Temmuz\or Ağuustos\or Eylül\or Ekim\or Kasım\or
62.34 Aralık\or\fi}
```

⁷²The file described in this section has version number v1.2m and was last revised on 2005/03/31.

⁷³Mustafa Burc, `z6001@rziris01.rrz.uni-hamburg.de` provided the code for this file. It is based on the work by Pierre Mackay; Turgut Uyar, `uyar@cs.itu.edu.tr` supplied additional translations in version 1.2j and later

62.35 `\space\number\year}}`

`\extrasturkish` The macro `\extrasturkish` will perform all the extra definitions needed for the Turkish language. The macro `\noextrasturkish` is used to cancel the actions of `\extrasturkish`.

Turkish typographic rules specify that a little ‘white space’ should be added before the characters ‘:’, ‘!’ and ‘=’. In order to insert this white space automatically these characters are made `\active`, so they have to be treated in a special way.

62.36 `\initiate@active@char{:}`

62.37 `\initiate@active@char{!}`

62.38 `\initiate@active@char{=}`

We specify that the turkish group of shorthands should be used.

62.39 `\addto\extrasturkish{\languageshorthands{turkish}}`

These characters are ‘turned on’ once, later their definition may vary.

62.40 `\addto\extrasturkish{%`

62.41 `\bbl@activate{:}\bbl@activate{!}\bbl@activate{=}}`

For Turkish texts `\frenchspacing` should be in effect. We make sure this is the case and reset it if necessary.

62.42 `\addto\extrasturkish{\bbl@frenchspacing}`

62.43 `\addto\noextrasturkish{\bbl@nonfrenchspacing}`

`\turkish@sh@!@` The definitions for the three active characters were made using intermediate macros. These are defined now. The insertion of extra ‘white space’ should only happen outside math mode, hence the check `\ifmmode` in the macros.

62.44 `\declare@shorthand{turkish}{:}{%`

62.45 `\ifmmode`

62.46 `\string:%`

62.47 `\else\relax`

62.48 `\ifhmode`

62.49 `\ifdim\lastskip>\z@`

62.50 `\unskip\penalty\@M\thinspace`

62.51 `\fi`

62.52 `\fi`

62.53 `\string:%`

62.54 `\fi}`

62.55 `\declare@shorthand{turkish}{!}{%`

62.56 `\ifmmode`

62.57 `\string!%`

62.58 `\else\relax`

62.59 `\ifhmode`

62.60 `\ifdim\lastskip>\z@`

62.61 `\unskip\penalty\@M\thinspace`

62.62 `\fi`

62.63 `\fi`

62.64 `\string!%`

62.65 `\fi}`

62.66 `\declare@shorthand{turkish}{=}{%`

62.67 `\ifmmode`

62.68 `\string=%`

62.69 `\else\relax`

62.70 `\ifhmode`

62.71 `\ifdim\lastskip>\z@`

62.72 `\unskip\kern\fontdimen2\font`

62.73 `\kern-1.4\fontdimen3\font`

62.74 `\fi`

62.75 `\fi`

62.76 `\string=%`

62.77 `\fi}`

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
62.78 \ldf@finish{turkish}
```

```
62.79 \end{code}
```

63 The Hebrew language

The file `hebrew.dtx`⁷⁴ provides the following packages and files for Hebrew language support:

`hebrew.ldf` file defines all the language-specific macros for the Hebrew language.

`rlbabel.def` file is used by `hebrew.ldf` for bidirectional versions of the major L^AT_EX commands and environments. It is designed to be used with other right-to-left languages, not only with Hebrew.

`hebcals` package defines a set of macros for computing Hebrew date from Gregorian one.

Additional Hebrew input and font encoding definition files that should be included and used with `hebrew.ldf` are:

`hebinp.dtx` provides Hebrew input encodings, such as ISO 8859-8, MS Windows codepage 1255 or IBM PC codepage 862 (see Section 64 on page 371).

`hebrew.fdd` contains Hebrew font encodings, related font definition files and `hebfont` package that provides Hebrew font switching commands (see Section 65 on page 377 for further details).

L^AT_EX 2.09 compatibility files are included with `heb209.dtx` and gives possibility to compile existing L^AT_EX 2.09 Hebrew documents with small (if any) changes (see Section 66 on page 392 for details).

Finally, optional document class `hebtech` may be useful for writing theses and dissertations in both Hebrew and English (and any other languages included with `babel`). It designed to meet requirements of the Graduate School of the Technion — Israel Institute of Technology.

As of version 2.3e hebtech is no longer distributed together with heblatex. It should be part of a new "hebclasses" package

63.1 Acknowledgement

The following people have contributed to Hebrew package in one way or another, knowingly or unknowingly. In alphabetical order: Irina Abramovici, Yaniv Bargury, Yael Dubinsky, Sergio Fogel, Dan Haran, Rama Porrat, Michail Rozman, Alon Ziv.

Tatiana Samoilov and Vitaly Surazhsky found a number of serious bugs in preliminary version of Hebrew package.

A number of other people have contributed comments and information. Specific contributions are acknowledged within the document.

I want to thank my wife, Vita, and son, Mishka, for their infinite love and patience.

If you made a contribution and I haven't mentioned it, don't worry, it was an accident. I'm sorry. Just tell me and I will add you to the next version.

63.2 The DOCSTRIP modules

The following modules are used in the implementation to direct DOCSTRIP in generating external files:

<code>driver</code>	produce a documentation driver file
<code>hebrew</code>	produce Hebrew language support file
<code>rightleft</code>	create right-to-left support file
<code>calendar</code>	create Hebrew calendar package

⁷⁴The Hebrew language support files described in this section have version number v2.3h and were last revised on 2005/03/30.

A typical DOCSTRIP command file would then have entries like:

```
\generateFile{hebrew.ldf}{t}{\from{hebrew.dtx}{hebrew}}
```

63.3 Hebrew language definitions

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
63.1 (*hebrew)
63.2 \LdfInit{hebrew}{captionshebrew}
```

When this file is read as an option, i.e., by the `\usepackage` command, `hebrew` will be an ‘unknown’ language, in which case we have to make it known. So we check for the existence of `\l@hebrew` to see whether we have to do something here.

```
63.3 \ifx\l@hebrew\@undefined
63.4 \nopatterns{Hebrew}%
63.5 \adddialect\l@hebrew0
63.6 \fi
```

`\hebrewencoding` *FIX DOCS REGARDING 8BIT*

Typesetting Hebrew texts implies that a special input and output encoding needs to be used. Generally, the user may choose between different available Hebrew encodings provided. The current support for Hebrew uses all available fonts from the Hebrew University of Jerusalem encoded in ‘old-code’ 7-bit encoding also known as Israeli Standard SI-960. We define for these fonts the Local Hebrew Encoding LHE (see the file `hebrew.fdd` for more details), and the LHE encoding definition file should be loaded by default.

Other fonts are available in `windows-cp1255` (a superset of ISO-8859-8 with nikud). For those, the encoding `HE8` should be used. Such fonts are, e.g., windows’ TrueType fonts (once converted to Type1 or MetaFont) and IBM’s Type1 fonts.

However, if an user wants to use another font encoding, for example, cyrillic encoding `T2` and extended latin encoding `T1`, — he/she has to load the corresponding file *before* the `hebrew` package. This may be done in the following way:

```
\usepackage[LHE,T2,T1]{fontenc}
\usepackage[hebrew,russian,english]{babel}
```

We make sure that the LHE encoding is known to \LaTeX at end of this package.

Also note that if you want to use the encoding `HE8`, you should define the following in your document, *before loading babel*:

```
\def\HeblatexEncoding{HE8}
\def\HeblatexEncodingFile{he8enc}
```

```
63.7 \providecommand{\HeblatexEncoding}{LHE}%
63.8 \providecommand{\HeblatexEncodingFile}{lheenc}%
63.9 \newcommand{\heblatex@set@encoding}[2]{
63.10 }
63.11 \AtEndOfPackage{%
63.12 \ifpackageloaded{fontenc}{%
63.13 \ifl@aded{def}{%
63.14 \HeblatexEncodingFile}{\def\hebrewencoding{\HeblatexEncoding}}{%
63.15 }{%
63.16 \input{\HeblatexEncodingFile.def}%
63.17 \def\hebrewencoding{\HeblatexEncoding}%
63.18 }}
```

We also need to load `inputenc` package with one of the Hebrew input encodings. By default, we set up the 8859-8 codepage. If an user wants to use many input encodings in the same document, for example, the MS Windows Hebrew codepage `cp1255` and the standard IBM PC Russian codepage `cp866`, he/she has to load the corresponding file *before* the `hebrew` package too. This may be done in the following way:

```
\usepackage[cp1255,cp866]{inputenc}
\usepackage[hebrew,russian,english]{babel}
```

An user can switch input encodings in the document using the command `\inputencoding`, for example, to use the cp1255:

```
\inputencoding{cp1255}
```

```
63.19 \AtEndOfPackage{%
63.20   \ifpackageloaded{inputenc}{\RequirePackage[8859-8]{inputenc}}}
```

The next step consists of defining commands to switch to (and from) the Hebrew language.

`\hebrewhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`. They are set to 2.

```
63.21 \providehyphenmins{\CurrentOption}{\tw@}{\tw@}
```

`\captionshebrew` The macro `\captionshebrew` replaces all captions used in the four standard document classes provided with L^AT_EX 2_ε with their Hebrew equivalents.

```
63.22 \addto\captionshebrew{%
63.23   \def\prefacename{\@ensure@R{\hebmem\hebbet\hebvav\hebalef}}}%
63.24   \def\refname{\@ensure@R{\hebresh\hebshin\hebyod\hebmem\hebtav\
63.25     \hebmem\hebqof\hebvav\hebresh\hebvav\hebtav}}}%
63.26   \def\abstractname{\@ensure@R{\hebtav\hebqof\hebtsadi\hebyod\hebresh}}}%
63.27   \def\bibname{\@ensure@R{\hebbet\hebyod\hebbet\heblamed\hebyod\hebvav\
63.28     \hebgimel\hebresh\hebpe\hebyod\hebhe}}}%
63.29   \def\chaptername{\@ensure@R{\hebpe\hebresh\hebqof}}}%
63.30   \def\appendixname{\@ensure@R{\hebnun\hebsamekh\hebpe\hebbet}}}%
63.31   \def\contentsname{\@ensure@R{%
63.32     \hebtav\hebvav\hebkaf\hebfinalnun\
63.33     \hebayin\hebnun\hebyod\hebyod\hebnun\hebyod\hebfinalmem}}}%
63.34   \def\listfigurename{\@ensure@R{%
63.35     \hebresh\hebshin\hebyod\hebmem\hebtav\
63.36     \hebalef\hebyod\hebvav\hebresh\hebyod\hebfinalmem}}}%
63.37   \def\listtablename{\@ensure@R{%
63.38     \hebresh\hebshin\hebyod\hebmem\hebtav\
63.39     \hebtet\hebbet\heblamed\hebalef\hebvav\hebtav}}}%
63.40   \def\indexname{\@ensure@R{\hebmem\hebpe\hebtav\hebbet}}}%
63.41   \def\figurename{\@ensure@R{\hebalef\hebyod\hebvav\hebresh}}}%
63.42   \def\tablename{\@ensure@R{\hebtet\hebbet\heblamed\hebhe}}}%
63.43   \def\partname{\@ensure@R{\hebbet\heblamed\hebqof}}}%
63.44   \def\enclname{\@ensure@R{\hebresh\hebtsadi\hebbet}}}%
63.45   \def\ccname{\@ensure@R{\hebhe\hebayin\hebtav\hebqof\hebyod\
63.46     \hebfinalmem}}}%
63.47   \def\headtoname{\@ensure@R{\hebalef\heblamed}}}%
63.48   \def\pagename{\@ensure@R{\hebayin\hebmem\hebvav\hebdalet}}}%
63.49   \def\psname{\@ensure@R{\hebnun.\hebbet.}}}%
63.50   \def\seename{\@ensure@R{\hebresh\hebalef\hebhe}}}%
63.51   \def\alsoname{\@ensure@R{\hebresh\hebalef\hebhe\hebgimel\
63.52     \hebmemesof}}}%
63.53   \def\proofname{\@ensure@R{\hebhe\hebvav\hebkaf\hebbet\hebhe}}}%
63.54   \def\glossaryname{\@ensure@L{Glossary}}}% <-- Needs translation
63.55 }
```

`\slidelabel` Here we fix the macro `slidelabel` of the seminar package. Note that this still won't work well enough when overlays will be involved

```
63.56 \ifclassloaded{seminar}{%
63.57   \def\slidelabel{\bf \if@rlR{\hebshin\hebqof\hebfinalpe}\theslide}%
63.58   \else\L{Slide \theslide}%
63.59   \fi}%
63.60 }
```


Here we provide an user with translation of Gregorian dates to Hebrew. In addition, the `hebc` package can be used to create Hebrew calendar dates.

`\hebmonth` The macro `\hebmonth{month}` produces month names in Hebrew.

```
63.61 \def\hebmonth#1{%
63.62   \ifcase#1\or \hebyod\hebnun\hebvav\hebalef\hebreash\or %
63.63     \hebpe\hebbet\hebreash\hebvav\hebalef\hebreash\or %
63.64     \hebmam\hebreash\hebfinalsadi\or %
63.65     \hebalef\hebpe\hebreash\hebyod\heblamed\or %
63.66     \hebmam\hebalef\hebyod\or \hebyod\hebvav\hebnun\hebyod\or %
63.67     \hebyod\hebvav\heblamed\hebyod\or %
63.68     \hebalef\hebvav\hebgimel\hebvav\hebsamekh\hebtet\or %
63.69     \hebsamekh\hebpe\hebtet\hebmam\hebbet\hebreash\or %
63.70     \hebalef\hebvav\hebgof\hebtet\hebvav\hebbet\hebreash\or %
63.71     \hebnun\hebvav\hebbet\hebmam\hebbet\hebreash\or %
63.72     \hebdalet\hebtsadi\hebmam\hebbet\hebreash\fi}
```

`\hebdate` The macro `\hebdate{day}{month}{year}` translates a given Gregorian date to Hebrew.

```
63.73 \def\hebdate#1#2#3{%
63.74   \beginR\beginL\number#1\endL\ \hebbet\hebmonth{#2}
63.75     \beginL\number#3\endL\endR}
```

`\hebd` The macro `\hebd` will replace `\today` command when in Hebrew mode.

```
63.76 \def\hebd{\hebdate{\day}{\month}{\year}}
```

`\datehebrew` The macro `\datehebrew` redefines the command `\today` to produce Gregorian dates in Hebrew. It uses the macro `\hebd`.

```
63.77 \def\datehebrew{\let\today=\hebd}
```

The macro `\extrashebrew` will perform all the extra definitions needed for the Hebrew language. The macro `\noextrashebrew` is used to cancel the actions of `\extrashebrew`.

`\extrashebrew` We switch font encoding to Hebrew and direction to right-to-left. We cannot use the regular language switching commands (for example, `\sethebrew` and `\unsethebrew` or `\selectlanguage{hebrew}`), when in restricted horizontal mode, because it will result in *unbalanced* `\beginR` or `\beginL` primitives. Instead, in T_EX's restricted horizontal mode, the `\L{latin text}` and `\R{hebrew text}`, or `\embox{latin text}` and `\hmbbox{hebrew text}` should be used.

Hence, we use `\beginR` and `\beginL` switching commands only when not in restricted horizontal mode.

```
63.78 \addto\extrashebrew{%
63.79   \tohebrew%
63.80   \ifhmode\ifinner\else\beginR\fi\fi}
```

`\noextrashebrew` The macro `\noextrashebrew` is used to cancel the actions of `\extrashebrew`. We switch back to the previous font encoding and restore left-to-right direction.

```
63.81 \addto\noextrashebrew{%
63.82   \fromhebrew%
63.83   \ifhmode\ifinner\else\beginL\fi\fi}
```

Generally, we can switch to- and from- Hebrew by means of standard `babel`-defined commands, for example,

```
\selectlanguage{hebrew}
```

or

```
\begin{otherlanguage}{hebrew}
  some Hebrew text
\end{otherlanguage}
```

Now we define two additional commands that offer the possibility to switch to and from Hebrew language. These commands are backward compatible with the previous versions of `hebrew.sty`.

`\sethebrew` The command `\sethebrew` will switch from the current font encoding to the hebrew font encoding, and from the current direction of text to the right-to-left mode. The command `\unsethebrew` switches back.

Both commands use standard right-to-left switching macros `\setrlanguage{rtl language name}` and `\unsetrlanguage{rtl language name}`, that defined in the `rlbabel.def` file.

```
63.84 \def\sethebrew{\setrlanguage{hebrew}}
63.85 \def\unsethebrew{\unsetrlanguage{hebrew}}
```

`\hebrewtext` The following two commands are *obsolete* and work only in L^AT_EX2.09 compatibility mode. They are synonyms of `\sethebrew` and `\unsethebrew` defined above.

```
63.86 \if@compatibility
63.87   \let\hebrewtext=\sethebrew
63.88   \let\nohebrewtext=\unsethebrew
63.89 \fi
```

`\tohebrew` These two commands change only the current font encoding to- and from- Hebrew encoding. Their implementation uses `\@torl{language name}` and `\@fromrl` macros defined in `rlbabel.def` file. Both commands may be useful *only* for package and class writers, not for regular users.

```
63.90 \def\tohebrew{\@torl{hebrew}}%
63.91 \def\fromhebrew{\@fromrl}
```

`\@hebrew` Sometimes we need to preserve Hebrew mode without knowing in which environment we are located now. For these cases, the `\@hebrew{hebrew text}` macro will be useful. Not that this macro is similar to the `\@number` and `\@latin` macros defined in `rlbabel.def` file.

```
63.92 \def\@hebrew#1{\beginR{\tohebrew#1}\endR}
63.93 \def\@hebrew{\protect\@hebrew}
```

63.3.1 Hebrew numerals

We provide commands to print numbers in the traditional notation using Hebrew letters. We need commands that print a Hebrew number from a decimal input, as well as commands to print the value of a counter as a Hebrew number.

`\if@gim@apost` Hebrew numbers can be written in various styles: with or without apostrophes, and with the letters kaf, mem, nun, pe, tsadi as either final or initial forms when they are the last letters in the sequence. We provide two flags to set the style options.

```
63.94 \newif\if@gim@apost % whether we print apostrophes
63.95 \newif\if@gim@final  % whether we use final or initial letters
```

`\hebrewnumeral` The commands that print a Hebrew number must specify the style locally: relying on a global style option could cause a counter to print in an inconsistent manner—for instance, page numbers might appear in different styles if the global style option changed mid-way through a document. The commands only allow three of the four possible flag combinations (I do not know of a use that requires the combination of final letters and no apostrophes –RA).

Each command sets the style flags and calls `\@hebrew@numeral`. Double braces are used in order to protect the values of `\@tempcnta` and `\@tempcntb`, which are changed by this call; they also keep the flag assignments local (this is not important because the global values are never used).

```
63.96 \newcommand*{\hebrewnumeral}[1] % no apostrophe, no final letters
63.97 {\@gim@finalfalse\@gim@apostfalse\@hebrew@numeral{#1}}
```

```

63.98 \newcommand*{\Hebrewnumeral}[1] % apostrophe, no final letters
63.99 {\@gim@finalfalse\@gim@aposttrue\@hebrew@numeral{#1}}
63.100 \newcommand*{\Hebrewnumeralfinal}[1] % apostrophe, final letters
63.101 {\@gim@finaltrue\@gim@aposttrue\@hebrew@numeral{#1}}

```

`\alph` Counter-printing commands are based on the above commands. The natural name for the counter-printing commands is `\alph`, because Hebrew numerals are the only way to represent numbers with Hebrew letters (kaf always means 20, never 11). `\Alph` Hebrew has no uppercase letters, hence no need for the familiar meaning of `\Alph`; we therefore define `\alph` to print counters as Hebrew numerals without apostrophes, and `\Alph` to print with apostrophes. A third form, `\Alphfinal`, is provided to print with apostrophes and final letters, as is required for Hebrew year designators. The commands `\alph` and `\Alph` are defined in `latex.ltx`, and we only need to redefine the internal commands `\@alph` and `\@Alph`; for `\Alphfinal` we need to provide both a wrapper and an internal command. The counter printing commands are made semi-robust: without the `\protect`, commands like `\theenumii` break (I'm not quite clear on why this happens, -RA); at the same time, we cannot make the commands too robust (e.g. with `\DeclareRobustCommand`) because this would enter the command name rather than its value into files like `.aux`, `.toc` etc. The old meanings of meaning of `\@alph` and `\@Alph` are saved upon entering Hebrew mode and restored upon exiting it.

```

63.102 \addto\extrashebrew{%
63.103   \let\saved@alph=\@alph%
63.104   \let\saved@Alph=\@Alph%
63.105   \renewcommand*{\@alph}[1]{\protect\hebrewnumeral{\number#1}}%
63.106   \renewcommand*{\@Alph}[1]{\protect\Hebrewnumeral{\number#1}}%
63.107   \def\Alphfinal#1{\expandafter\@Alphfinal\csname c@#1\endcsname}%
63.108   \providecommand*{\@Alphfinal}[1]{\protect\Hebrewnumeralfinal{\number#1}}%
63.109 \addto\noextrashebrew{%
63.110   \let\@alph=\saved@alph%
63.111   \let\@Alph=\saved@Alph}

```

Note that `\alph` (without apostrophes) is already the appropriate choice for the second-level enumerate label, and `\Alph` (with apostrophes) is an appropriate choice for appendix; however, the default L^AT_EX labels need to be redefined for appropriate cross-referencing, see below. L^AT_EX default class files specify `\Alph` for the fourth-level enumerate level, this should probably be changed. Also, the way labels get flushed left by default looks inappropriate for Hebrew numerals, so we should redefine `\labelenumii` as well as `\labelenumiv` (presently not implemented).

`\theenumii` Cross-references to counter labels need to be printed according to the language environment in which a label was issued, not the environment in which it is called: `\theenumiv` for example, a label (1b) issued in a Latin environment should be referred to as (1b) in a Hebrew text, and label (2dalet) issued in a Hebrew environment should be referred to as (2dalet) in a Latin text. This was the unanimous opinion in a poll sent to the IvriT_EX list. We therefore redefine `\theenumii` and `\theenumiv`, so that an explicit language instruction gets written to the `.aux` file.

```

63.112 \renewcommand{\theenumii}
63.113 {\ifrl\protect\hebrewnumeral{\number\c@enumii}%
63.114   \else\protect\L{\protect\@alph{\number\c@enumii}}\fi}
63.115 \renewcommand{\theenumiv}
63.116 {\ifrl\protect\Hebrewnumeral{\number\c@enumiv}%
63.117   \else\protect\L{\protect\@Alph{\number\c@enumiv}}\fi}

```

We also need to control for the font and direction in which a counter label is printed. Direction is straightforward: a Latin label like (1b) should be written left-to-right when called in a Hebrew text, and a Hebrew label like (2dalet) should be written right-to-left when called in a Latin text. The font question is more delicate, because we should decide whether the numerals should be typeset in

the font of the language environment in which the label was issued, or that of the environment in which it is called.

- A purely numeric label like (23) looks best if it is set in the font of the surrounding language.
- But a mixed alphanumeric label like (1b) looks weird if the ‘1’ is taken from the Hebrew font; likewise, (2dalet) looks weird if the ‘2’ is taken from a Latin font.
- Finally, mixing the two possibilities is worst, because a single Hebrew sentence referring to examples (1b) and (2) would take the ‘1’ from the Latin font and the ‘2’ from the Hebrew font, and this looks really awful. (It is also very hard to implement).

In light of the conflicting considerations it seems like there’s no perfect solution. I have chosen to implement the top option, where numerals are taken from the font of the surrounding language, because it seems to me that reference to purely numeric labels is the most common, so this gives a good solution to the majority of cases and a mediocre solution to the minority.

We redefine the `\label` command which writes to the `.aux` file. Depending on the language environment we issue appropriate `\beginR/L... \endR/L` commands to control the direction without affecting the font. Since these commands do not affect the value of `\if@rl`, we cannot use the macro `\@brackets` to determine the correct brackets to be used with `\p@enumiii`; instead, we let the language environment determine an explicit definition.

```

63.118 \def\label#1{\@bsphack
63.119   \if@rl
63.120     \def\p@enumiii{\p@enumii}\theenumii}%
63.121     \protected@write\@auxout{%
63.122       {\string\newlabel{#1}{\beginR\@currentlabel\endR}{\thepage}}}%
63.123   \else
63.124     \def\p@enumiii{\p@enumii}\theenumii}%
63.125     \protected@write\@auxout{%
63.126       {\string\newlabel{#1}{\beginL\@currentlabel\endL}{\thepage}}}%
63.127   \fi
63.128   \@esphack}

```

NOTE: it appears that the definition of `\label` is language-independent and thus belongs in `rlbabel.def`, but this is not the case. The decision to typeset label numerals in the font of the surrounding language is reasonable for Hebrew, because mixed-font (1b) and (2dalet) are somewhat acceptable. The same may not be acceptable for Arabic, whose numeral glyphs are radically different from those in the Latin fonts. The decision about the direction may also be different for Arabic, which is more right-to-left oriented than Hebrew (two examples: dates like 15/6/2003 are written left-to-right in Hebrew but right-to-left in Arabic; equations like $1 + 2 = 3$ are written left-to-right in Hebrew but right-to-left in Arabic elementary school textbooks using Arabic numeral glyphs). My personal hunch is that a label like (1b) in an Arabic text would be typeset left-to-right if the ‘1’ is a Western glyph, but right-to-left if the ‘1’ is an Arabic glyph. But this is just a guess, I’d have to ask Arab typesetters to find the correct answer. –RA.

`\appendix` The following code provides for the proper printing of appendix numbers in tables of contents. Section and chapter headings are normally bilingual: regardless of the text language, the author supplies each section/chapter with two headings—one for the Hebrew table of contents and one for the Latin table of contents. It makes sense that the label should be a Latin letter in the Latin table of contents and a Hebrew letter in the Hebrew table of contents. The definition is similar to that of `\theenumii` and `\theenumiv` above, but additional `\protect` commands ensure that the entire condition is written to the `.aux` file. The appendix number will therefore be typeset according to the environment in which it is used rather

than issued: a Hebrew number (with apostrophes) in a Hebrew environment and a Latin capital letter in a Latin environment (the command `\@@Alph` is set in `rlbabel.def` to hold the default meaning of L^AT_EX [latin] `\@Alph`, regardless of the mode in which it is issued). The net result is that the second appendix will be marked with ‘B’ in the Latin table of contents and with ‘bet’ in the Hebrew table of contents; the mark in the main text will depend on the language of the appendix itself.

```

63.129 \ifclassloaded{letter}{}{%
63.130 \ifclassloaded{slides}{}{%
63.131   \let\@@appendix=\appendix%
63.132   \ifclassloaded{article}{%
63.133     \renewcommand\appendix{\@@appendix%
63.134       \renewcommand\thesection
63.135         {\protect\if@rl\protect\Hebrewnumeral{\number\c@section}%
63.136         \protect\else\@@Alph\c@section\protect\fi}}
63.137   {\renewcommand\appendix{\@@appendix%
63.138     \renewcommand\thechapter
63.139       {\protect\if@rl\protect\Hebrewnumeral{\number\c@chapter}%
63.140       \protect\else\@@Alph\c@chapter\protect\fi}}}}

```

QUESTION: is this also the appropriate way to refer to an appendix in the text, or should we retain the original label the same way we did with `enumerate` labels?

ANOTHER QUESTION: are similar redefinitions needed for other counters that generate texts in bilingual lists like `.lof/.fol` and `.lot/.tol`? –RA.

`\@hebrew@numeral` The command `\@hebrew@numeral` prints a Hebrew number. The groups of thousands, millions, billions are separated by apostrophes and typeset without apostrophes or final letters; the remainder (under 1000) is typeset conventionally, with the selected styles for apostrophes and final letters. The function calls on `\gim@no@mil` to typeset each three-digit block. The algorithm is recursive, but the maximum recursion depth is 4 because T_EX only allows numbers up to $2^{31} - 1 = 2,147,483,647$. The typesetting routine is wrapped in `\@hebrew` in order to ensure that numbers are always typeset in Hebrew mode.

Initialize: `\@tempcnta` holds the value, `\@tempcntb` is used for calculations.

```

63.141 \newcommand*{\@hebrew@numeral}[1]
63.142 {\@hebrew{\@tempcnta=#1\@tempcntb=#1\relax
63.143   \divide\@tempcntb by 1000
        If we're under 1000, call \gim@nomil
63.144   \ifnum\@tempcntb=0\gim@nomil\@tempcnta\relax
        If we're above 1000 then force no apostrophe and no final letter styles for the
        value above 1000, recur for the value above 1000, add an apostrophe, and call
        \gim@nomil for the remainder.
63.145   \else{\gim@apostfalse\gim@finalfalse\@hebrew@numeral\@tempcntb}'%
63.146     \multiply\@tempcntb by 1000\relax
63.147     \advance\@tempcnta by -\@tempcntb\relax
63.148     \gim@nomil\@tempcnta\relax
63.149   \fi
63.150 }}

```

NOTE: is it the case that 15,000 and 16,000 are written as yod-he and yod-vav, rather than tet-vav and tet-zayin? This vaguely rings a bell, but I’m not certain. If this is the case, then the current behavior is incorrect and should be changed. –RA.

`\gim@nomil` The command `\gim@nomil` typesets an integer between 0 and 999 (for 0 it typesets nothing). The code has been modified from the old `hebcsl.sty` (appropriate credits—Boris Lavva and Michail Rozman ?). `\@tempcnta` holds the total value that remains to be typeset. At each stage we find the highest valued letter that is less than or equal to `\@tempcnta`, and call on `\gim@print` to subtract this value and print the letter.

Initialize: \@tempcnta holds the value, there is no previous letter.

```
63.151 \newcommand*{\gim@nomil}[1]{\@tempcnta=#1\@gim@prevfalse}
```

Find the hundreds digit.

```
63.152 \@tempcntb=\@tempcnta\divide\@tempcntb by 100\relax % hundreds digit
63.153 \ifcase\@tempcntb % print nothing if no hundreds
63.154 \or\gim@print{100}{\hebqof}%
63.155 \or\gim@print{200}{\hebresh}%
63.156 \or\gim@print{300}{\hebshin}%
63.157 \or\gim@print{400}{\hebtav}%
63.158 \or\hebtav\@gim@prevtrue\gim@print{500}{\hebqof}%
63.159 \or\hebtav\@gim@prevtrue\gim@print{600}{\hebresh}%
63.160 \or\hebtav\@gim@prevtrue\gim@print{700}{\hebshin}%
63.161 \or\hebtav\@gim@prevtrue\gim@print{800}{\hebtav}%
63.162 \or\hebtav\@gim@prevtrue\hebtav\gim@print{900}{\hebqof}%
63.163 \fi
```

Find the tens digit. The numbers 15 and 16 are traditionally printed as tet-vav (9 + 6) and tet-zayin (9 + 7) to avoid spelling the Lord's name.

```
63.164 \@tempcntb=\@tempcnta\divide\@tempcntb by 10\relax % tens digit
63.165 \ifcase\@tempcntb % print nothing if no tens
63.166 \or % number between 10 and 19
63.167 \ifnum\@tempcnta = 16 \gim@print {9}{\hebtet}% tet-zayin
63.168 \else\ifnum\@tempcnta = 15 \gim@print {9}{\hebtet}% tet-vav
63.169 \else \gim@print{10}{\hebyod}%
63.170 \fi % \@tempcnta = 15
63.171 \fi % \@tempcnta = 16
```

Initial or final forms are selected according to the current style option; \gim@print will force a non-final letter in non-final position by means of a local style change.

```
63.172 \or\gim@print{20}{\if@gim@final\hebfinalkaf\else\hebqof\fi}%
63.173 \or\gim@print{30}{\heblamed}%
63.174 \or\gim@print{40}{\if@gim@final\hebfinalmem\else\hebmam\fi}%
63.175 \or\gim@print{50}{\if@gim@final\hebfinalnun\else\hebnun\fi}%
63.176 \or\gim@print{60}{\hebsamekh}%
63.177 \or\gim@print{70}{\hebayin}%
63.178 \or\gim@print{80}{\if@gim@final\hebfinalpe\else\hebpe\fi}%
63.179 \or\gim@print{90}{\if@gim@final\hebfinaltsadi\else\hebttsadi\fi}%
63.180 \fi
```

Print the ones digit.

```
63.181 \ifcase\@tempcnta % print nothing if no ones
63.182 \or\gim@print{1}{\hebalef}%
63.183 \or\gim@print{2}{\hebbet}%
63.184 \or\gim@print{3}{\hebgimel}%
63.185 \or\gim@print{4}{\hebdalet}%
63.186 \or\gim@print{5}{\hebhe}%
63.187 \or\gim@print{6}{\hebvav}%
63.188 \or\gim@print{7}{\hebzayin}%
63.189 \or\gim@print{8}{\hebbet}%
63.190 \or\gim@print{9}{\hebtet}%
63.191 \fi
63.192 }
```

\gim@print The actual printing routine typesets a digit with the appropriate apostrophes:
 \if@gim@prev if a number sequence consists of a single letter then it is followed by a single apostrophe, and if it consists of more than one letter then a double apostrophe is inserted before the last letter. We typeset the letters one at a time, keeping a flag that tells us if any previous letters had been typeset.

```
63.193 \newif\if@gim@prev % flag if a previous letter has been typeset
```

For each letter, we first subtract its value from the total. Then,

- if the result is zero then this is the last letter; we check the flag to see if this is the only letter and print it with the appropriate apostrophe;

- if the result is not zero then there remain additional letters to be typeset; we print without an apostrophe and set the ‘previous letter’ flag.

`\@tempcnta` holds the total value that remains to be typeset. We first deduct the letter’s value from `\@tempcnta`, so `\@tempcnta` is zero if and only if this is the last letter.

```
63.194 \newcommand*{\gim@print}[2]{%    #2 is a letter, #1 is its value.
63.195   \advance\@tempcnta by -#1\relax% deduct the value from the remainder
```

If this is the last letter, we print with the appropriate apostrophe (depending on the style option): if there is a preceding letter, print "x if the style calls for apostrophes, x if it doesn’t; otherwise, this is the only letter: print x’ if the style calls for apostrophes, x if it doesn’t.

```
63.196   \ifnum\@tempcnta=0% if this is the last letter
63.197     \ifgim@prev\ifgim@apost"\fi#2%
63.198     \else#2\ifgim@apost'\fi\fi%
```

If this is not the last letter: print a non-final form (by forcing a local style option) and set the ‘previous letter’ flag.

```
63.199   \else{\@gim@finalfalse#2}\@gim@prevtrue\fi}
```

`\hebr` The older Hebrew counter commands `\hebr` and `\gim` are retained in order to keep older documents from breaking. They are set to be equivalent to `\alph`, and their use is deprecated. Note that `\hebr` gives different results than it had in the past—it now typesets 11 as yod-alef rather than kaf.

```
63.200 \let\hebr=\alph
63.201 \let\gim=\alph
```

For backward compatibility with ‘older’ `hebrew.sty` packages, we define Hebrew equivalents of some useful L^AT_EX commands. Note, however, that 8-bit macros defined in Hebrew are no longer supported.

```
63.202 \def\hebcopy{\protect\R{\hebhe\hebayin\hebtav\hebqof}}
63.203 \def\hebincl{\protect\R{\hebresh\hebtasadi"\hebbet}}
63.204 \def\hebpagel{\protect\R{\hebayin\hebmam\hebvav\hebdalet}}
63.205 \def\hebto{\protect\R{\hebayin\hebdalet}}
```

`\hadgesh` produce “poor man’s bold” (heavy printout), when used with normal font glyphs. It is advisable to use bold font (for example, *Dead Sea*) instead of this macro.

```
63.206 \def\hadgesh#1{\leavevmode\setbox0=\hbox{#1}%
63.207   \kern-.025em\copy0\kern-\wd0
63.208   \kern.05em\copy0\kern-\wd0
63.209   \kern-.025em\raise.0433em\box0 }
```

`\piska` and `\piskapiska` sometimes used in ‘older’ hebrew sources, and should not be used in L^AT_EX 2_ε.

```
63.210 \ifcompatibility
63.211   \def\piska#1{\item{#1}\hangindent=-\hangindent}
63.212   \def\piskapiska#1{\itemitem{#1}\hangindent=-\hangindent}
63.213 \fi
```

The following commands are simply synonyms for the standard ones, provided with L^AT_EX 2_ε.

```
63.214 \let\makafgadol=\textendash
63.215 \let\makafanak=\textemdash
63.216 \let\geresh=\textquoteright
63.217 \let\opengeresh=\textquoteright
63.218 \let\closegeresh=\textquoteleft
63.219 \let\openquote=\textquotedblright
63.220 \let\closequote=\textquotedblleft
63.221 \let\leftquotation=\textquotedblright
63.222 \let\rightquotation=\textquotedblleft
```

We need to ensure that Hebrew is used as the default right-to-left language at `\begin{document}`. The mechanism of defining the `\@rllanguagename` is the same as in `babel`'s `\language`: the last right-to-left language in the `\usepackage{babel}` line is set as the default right-to-left language at document beginning.

For example, the following code:

```
\usepackage[russian,hebrew,arabic,greek,english]{babel}
```

will set the Arabic language as the default right-to-left language and the English language as the default language. As a result, the commands `\L{}` and `\embox{}` will use English and `\R{}` and `\hmbbox{}` will use Arabic by default. These defaults can be changed with the next `\sethebrew` or `\selectlanguage{language name}` command.

```
63.223 \AtBeginDocument{\def\@rllanguagename{hebrew}}
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
63.224 \ldf@finish{hebrew}
```

```
63.225 </hebrew>
```

63.4 Right to left support

This file `rlbabel.def` defines necessary bidirectional macro support for $\text{\LaTeX} 2_{\epsilon}$. It is designed for use not only with Hebrew, but with any Right-to-Left languages, supported by `babel`. The macros provided in this file are language and encoding independent.

Right-to-left languages will use \TeX extensions, namely \TeX primitives `\beginL`, `\endL` and `\beginR`, `\endR`, currently implemented only in $\epsilon\text{-TeX}$ and in \TeX--XeT .

If $\epsilon\text{-TeX}$ is used, we should switch it to the *enhanced* mode:

```
63.226 (*rightleft)
63.227 \ifx\TeXeTstate\undefined\else%
63.228   \TeXeTstate=1
63.229 \fi
```

Note, that $\epsilon\text{-TeX}$'s format file should be created for *extended* mode. Mode can be checked by running $\epsilon\text{-TeX}$ on some \TeX file, for example:

```
This is e-TeX, Version 3.14159-1.1 (Web2c 7.0)
entering extended mode
```

The second line should be `entering extended mode`.

We check if user uses Right-to-Left enabled engine instead of regular Knuth's \TeX :

```
63.230 \ifx\beginL\@undefined%
63.231   \newlinechar'\^^J
63.232   \typeout{^^JTo avoid this error message,^^J%
63.233     run TeX--XeT or e-TeX engine instead of regular TeX.^^J}
63.234   \errmessage{Right-to-Left Support Error: use TeX--XeT or e-TeX
63.235     engine}%
63.236 \fi
```

63.4.1 Switching from LR to RL mode and back

`\@torl` and `\@fromrl` are called each time the horizontal direction changes. They do all that is necessary besides changing the direction. Currently their task is to change the encoding information and mode (condition `\if@rl`). They should not normally be called by users: user-level macros, such as `\sethebrew`

and `\unsethebrew`, as well as babel's `\selectlanguage` are defined in language-definition files and should be used to change default language (and direction).

Local direction changing commands (for small pieces of text): `\L{}`, `\R{}`, `\embox{}` and `\hmbx{}` are defined below in this file in language-independent manner.

`\if@rl` `rltrue` means that the main mode is currently Right-to-Left.
`rlfalse` means that the main mode is currently Left-to-Right.

```
63.237 \newif\if@rl
```

`\if@rlmain` This is the main direction of the document. Unlike `\if@rl` it is set once and never changes.

`rltrue` means that the document is Right-to-Left.
`rlfalse` means that the document is Left-to-Right.

Practically `\if@rlmain` is set according to the value of `\if@rl` in the beginning of the run.

```
63.238 \AtBeginDocument{% Here we set the main document direction
63.239   \newif\if@rlmain%
63.240   \if@rl% e.g: if the options to babel were [english,hebrew]
63.241     \@rlmaintrue%
63.242   \else% e.g: if the options to babel were [hebrew,english]
63.243     \@rlmainfalse%
63.244   \fi%
63.245 }
```

`\@torl` Switches current direction to Right-to-Left: saves current Left-to-Right encoding in `\lr@encodingdefault`, sets required Right-to-Left language name in `\@rllanguagename` (similar to babel's `\languagename`) and changes direction.

The Right-to-Left language encoding should be defined in `.ldf` file as special macro created by concatenation of the language name and string `encoding`, for example, for Hebrew it will be `\hebrewencoding`.

```
63.246 \DeclareRobustCommand{\@torl}[1]{%
63.247   \if@rl\else%
63.248     \let\lr@encodingdefault=\encodingdefault%
63.249   \fi%
63.250   \def\@rllanguagename{#1}%
63.251   \def\encodingdefault{\csname#1encoding\endcsname}%
63.252   \fontencoding{\encodingdefault}%
63.253   \selectfont%
63.254   \@rltrue}
```

`\@fromrl` Opposite to `\@torl`, switches current direction to Left-to-Right: restores saved Left-to-Right encoding (`\lr@encodingdefault`) and changes direction.

```
63.255 \DeclareRobustCommand{\@fromrl}{%
63.256   \if@rl%
63.257     \let\encodingdefault=\lr@encodingdefault%
63.258   \fi%
63.259   \fontencoding{\encodingdefault}%
63.260   \selectfont%
63.261   \@rlfalse}
```

`\selectlanguage` This standard babel's macro should be redefined to support bidirectional tables. We divide `\selectlanguage` implementation to two parts, and the first part calls the second `\@@selectlanguage`.

```
63.262 \expandafter\def\csname selectlanguage \endcsname#1{%
63.263   \edef\languagename{%
63.264     \ifnum\escapechar=\expandafter'\string#1\@empty
63.265     \else \string#1\@empty\fi}%
63.266   \@@selectlanguage{\languagename}}
```

`\@@selectlanguage` This new internal macro redefines a final part of the standard babel's `\selectlanguage` implementation.

Standard L^AT_EX provides us with 3 tables: Table of Contents (`.toc`), List of Figures (`.lof`), and List of Tables (`.lot`). In multi-lingual texts mixing Left-to-Right languages with Right-to-Left ones, the use of various directions in one table results in very ugly output. Therefore, these 3 standard tables will be used now only for Left-to-Right languages, and we will add 3 Right-to-Left tables (their extensions are simply reversed ones): RL Table of Contents (`.cot`), RL List of Figures (`.fol`), and RL List of Tables (`.lof`).

```

63.267 \def\@@selectlanguage#1{%
63.268   \select@language{#1}%
63.269   \if@filesw
63.270     \protected@write\@auxout{}\string\select@language{#1}%
63.271     \if@rl%
63.272       \addtocontents{cot}{\xstring\select@language{#1}}%
63.273       \addtocontents{fol}{\xstring\select@language{#1}}%
63.274       \addtocontents{tol}{\xstring\select@language{#1}}%
63.275     \else%
63.276       \addtocontents{toc}{\xstring\select@language{#1}}%
63.277       \addtocontents{lof}{\xstring\select@language{#1}}%
63.278       \addtocontents{lot}{\xstring\select@language{#1}}%
63.279     \fi%
63.280 \fi}

```

`\setrllanguage` The `\setrllanguage` and `\unsetrllanguage` pair of macros is proved to very useful in bilingual texts, for example, in Hebrew-English texts. The language-specific commands, for example, `\sethebrew` and `\unsethebrew` use these macros as basis.

Implementation saves and restores other language in `\other@language` variable, and uses internal macro `\@@selectlanguage`, defined above, to switch between languages.

```

63.281 \let\other@language=\language
63.282 \DeclareRobustCommand{\setrllanguage}[1]{%
63.283   \if@rl\else%
63.284     \let\other@language=\language%
63.285   \fi%
63.286   \def\language{#1}%
63.287   \@@selectlanguage{\language}}
63.288 \DeclareRobustCommand{\unsetrllanguage}[1]{%
63.289   \if@rl%
63.290     \let\language=\other@language%
63.291   \fi
63.292   \@@selectlanguage{\language}}

```

`\L` Macros for changing direction, originally taken from TUGboat. Usage: `\L{Left to Right text}` and `\R{Right to Left text}`. Numbers should also be enclosed in `\L{}`, as in `\L{123}`.

Note, that these macros do not receive language name as parameter. Instead, the saved `\@rllanguage` will be used. We assume that each Right-to-Left language defines `\tolanguage` and `\fromlanguage` macros in language definition file, for example, for Hebrew: `\tohebrew` and `\fromhebrew` macros in `hebrew.ldf` file.

The macros `\L` and `\R` include ‘protect’ to make them robust and allow use, for example, in tables.

Due to the fact that some packages have different definitions for `\L` the macro `\HeblatexRedefineL` is provided to override them. This may be required with `hyperref`, for instance.

```

63.293 \let\next=
63.294 \def\HeblatexRedefineL{%

```

```

63.295 \def\L{\protect\pL}%
63.296 }
63.297 \HeblatexRedefineL
63.298 \def\pL{\protect\afterassignment\moreL \let\next= }
63.299 \def\moreL{\bracetext \aftergroup\endL \beginL\csname
63.300 from\@rllanguagename\endcsname}
63.301 \def\R{\protect\pR}
63.302 \def\pR{\protect\afterassignment\moreR \let\next= }
63.303 \def\moreR{\bracetext \aftergroup\endR \beginR\csname
63.304 to\@rllanguagename\endcsname}
63.305 \def\bracetext{\ifcat\next{\else\ifcat\next}\fi
63.306 \errmessage{Missing left brace has been substituted}\fi \bgroup}
63.307 \everydisplay{\if@rl\aftergroup\beginR\fi }

\@ensure@R Two small internal macros, a-la \ensuremath
\@ensure@L63.308 \def\@ensure@R#1{\if@rl#1\else\R{#1}\fi}
63.309 \def\@ensure@L#1{\if@rlL{#1}\else#1\fi}

Take care of Right-to-Left indentation in every paragraph. Originally,
\noindent was redefined for right-to-left by Yaniv Bargury, then the implemen-
tation was rewritten by Alon Ziv using an idea by Chris Rowley: \noindent now
works unmodified.
63.310 \def\rl@everypar{\if@rl{\setbox\z@\lastbox\beginR\usebox\z@}\fi}
63.311 \let\o@everypar=\everypar
63.312 \def\everypar#1{\o@everypar{\rl@everypar#1}}

\hmbbox Useful vbox commands. All text in math formulas is best enclosed in these: LR
\embox text in \embox and RL text in \hmbbox. \mbox{} is useless for both cases, since
it typesets in Left-to-Right even for Right-to-Left languages (additions by Yaniv
Bargury).
63.313 \newcommand{\hmbbox}[1]{\mbox{\R{#1}}}
63.314 \newcommand{\embox}[1]{\mbox{\L{#1}}}

\@brackets When in Right-to-Left mode, brackets should be swapped. This macro receives 3
parameters: left bracket, content, right bracket. Brackets can be square brackets,
braces, or parentheses.
63.315 \def\@brackets#1#2#3{\protect\if@rl #3#2#1\protect\else
63.316 #1#2#3\protect\fi}

\@number \@number preserves numbers direction from Left to Right. \@latin in addition
\@latin switches current encoding to the latin.
63.317 \def\@@number#1{\ifmmode\else\beginL\fi#1\ifmmode\else\endL\fi}
63.318 \def\@@latin#1{\@@number{\@fromrl#1}}
63.319 \def\@number{\protect\@@number}
63.320 \def\@latin{\protect\@@latin}

```

63.4.2 Counters

To make counter references work in Right to Left text, we need to surround their original definitions with an `\@number{...}` or `\@latin{...}`. Note, that language-specific counters, such as `\hebr` or `\gim` are provided with language definition file.

We start with saving the original definitions:

```

63.321 \let\@arabic=\@arabic
63.322 \let\@roman=\@roman
63.323 \let\@Roman=\@Roman
63.324 \let\@alph=\@alph
63.325 \let\@Alph=\@Alph

```

`\@arabic` Arabic and roman numbers should be from Left to Right. In addition, roman
`\@roman` numerals, both lower- and upper-case should be in latin encoding.

```

\@Roman63.326 \def\@arabic#1{\@number{\@@arabic#1}}
63.327 \def\@roman#1{\@latin{\@@roman#1}}
63.328 \def\@Roman#1{\@latin{\@@Roman#1}}

\arabicnorl This macro preserves the original definition of \arabic (overrides the overriding
of \@arabic)
63.329 \def\arabicnorl#1{\expandafter\@@arabic\csname c@#1\endcsname}

\make@lr In Right to Left documents all counters defined in the standard document
classes article, report and book provided with LATEX 2ε, such as \thesection,
\thefigure, \theequation should be typed as numbers from left to right. To
ensure direction, we use the following \make@lr{counter} macro:
63.330 \def\make@lr#1{\begingroup
63.331 \toks@=\expandafter{#1}%
63.332 \edef\x{\endgroup
63.333 \def\noexpand#1{\noexpand\@number{\the\toks@}}}%
63.334 \x}

63.335 \@ifclassloaded{letter}{}{%
63.336 \@ifclassloaded{slides}{}{%
63.337 \make@lr\thesection
63.338 \make@lr\thesubsection
63.339 \make@lr\thesubsubsection
63.340 \make@lr\theparagraph
63.341 \make@lr\thesubparagraph
63.342 \make@lr\thefigure
63.343 \make@lr\thetable
63.344 }
63.345 \make@lr\theequation
63.346 }
```

63.4.3 Preserving logos

Preserve T_EX, L^AT_EX and L^AT_EX 2_ε logos.

```

\TeX
63.347 \let\@@TeX\TeX
63.348 \def\TeX{\@latin{\@@TeX}}

\LaTeX
63.349 \let\@@LaTeX\LaTeX
63.350 \def\LaTeX{\@latin{\@@LaTeX}}

\LaTeXe
63.351 \let\@@LaTeXe\LaTeXe
63.352 \def\LaTeXe{\@latin{\@@LaTeXe}}
```

63.4.4 List environments

List environments in Right-to-Left languages, are ticked and indented from the right instead of from the left. All the definitions that caused indentation are revised for Right-to-Left languages. L^AT_EX keeps track on the indentation with the `\leftmargin` and `\rightmargin` values.

`list` Thus we need to override the definition of the `\list` macro: when in RTL mode, the right margins are the begining of the line.

```

63.353 \def\list#1#2{%
63.354 \ifnum \@listdepth >5\relax
63.355 \toodeep
```

```

63.356 \else
63.357   \global\advance\@listdepth\@ne
63.358 \fi
63.359 \rightmargin\z@
63.360 \listparindent\z@
63.361 \itemindent\z@
63.362 \csname @list\romannumeral\the\@listdepth\endcsname
63.363 \def\@itemlabel{#1}%
63.364 \let\makelabel\@mklab
63.365 \@nmbrlistfalse
63.366 #2\relax
63.367 \@trivlist
63.368 \parskip\parsep
63.369 \parindent\listparindent
63.370 \advance\linewidth -\rightmargin
63.371 \advance\linewidth -\leftmargin

```

The only change in the macro is the `\if@rl` case:

```

63.372 \if@rl
63.373   \advance\@totalleftmargin \rightmargin
63.374 \else
63.375   \advance\@totalleftmargin \leftmargin
63.376 \fi
63.377 \parshape \@ne \@totalleftmargin \linewidth
63.378 \ignorespaces}

```

`\labelenumii` The `\labelenumii` and `\p@enumiii` commands use *parentheses*. They are revised
`\p@enumiii` to work Right-to-Left with the help of `\@brackets` macro defined above.

```

63.379 \def\labelenumii{\@brackets(\thenumii)}
63.380 \def\p@enumiii{\p@enumii\@brackets(\thenumii)}

```

63.4.5 Tables of moving stuff

Tables of moving arguments: table of contents (`toc`), list of figures (`lof`) and list of tables (`lot`) are handles here. These three default \LaTeX tables will be used now exclusively for Left to Right stuff.

Three additional Right-to-Left tables: RL table of contents (`cot`), RL list of figures (`fol`), and RL list of tables (`tol`) are added. These three tables will be used exclusively for Right to Left stuff.

`\@tableofcontents` We define 3 new macros similar to the standard \LaTeX tables, but with one parameter — table file extension. These macros will help us to define our additional
`\@listoffigures` tables below.
`\@listoftables`

```

63.381 \@ifclassloaded{letter}{\% other
63.382 \@ifclassloaded{slides}{\% other
63.383   \@ifclassloaded{article}{\% article
63.384     \newcommand\@tableofcontents[1]{\%
63.385       \section*\contentsname\@mkboth\%
63.386       {\MakeUppercase\contentsname}\%
63.387       {\MakeUppercase\contentsname}}\%
63.388     \@starttoc{#1}}
63.389   \newcommand\@listoffigures[1]{\%
63.390     \section*\listfigurename\@mkboth\%
63.391     {\MakeUppercase\listfigurename}\%
63.392     {\MakeUppercase\listfigurename}}\%
63.393     \@starttoc{#1}}
63.394   \newcommand\@listoftables[1]{\%
63.395     \section*\listtablename\@mkboth\%
63.396     {\MakeUppercase\listtablename}\%
63.397     {\MakeUppercase\listtablename}}\%
63.398     \@starttoc{#1}}\%
63.399   {\% else report or book

```

```

63.400 \newcommand\@tableofcontents[1]{%
63.401 \if@restonecolfalse\if@twocolumn\@restonecoltrue\onecolumn%
63.402 \fi\chapter*{\contentsname\@mkboth%
63.403 {\MakeUppercase\contentsname}%
63.404 {\MakeUppercase\contentsname}}%
63.405 \@starttoc{#1}\if@restonecol\twocolumn\fi}
63.406 \newcommand\@listoffigures[1]{%
63.407 \if@restonecolfalse\if@twocolumn\@restonecoltrue\onecolumn%
63.408 \fi\chapter*{\listfigurename\@mkboth%
63.409 {\MakeUppercase\listfigurename}%
63.410 {\MakeUppercase\listfigurename}}%
63.411 \@starttoc{#1}\if@restonecol\twocolumn\fi}
63.412 \newcommand\@listoftables[1]{%
63.413 \if@twocolumn\@restonecoltrue\onecolumn\else\@restonecolfalse\fi%
63.414 \chapter*{\listtablename\@mkboth%
63.415 {\MakeUppercase\listtablename}%
63.416 {\MakeUppercase\listtablename}}%
63.417 \@starttoc{#1}\if@restonecol\twocolumn\fi}}%

```

`\lrrtableofcontents` Left-to-Right tables are called now `\lrrxxx` and defined with the aid of three macros `\lrrlistoffigures` defined above (extensions `toc`, `lof`, and `lot`).

```

\lrrlistoftables63.418 \newcommand\lrrtableofcontents{\@tableofcontents{toc}}%
63.419 \newcommand\lrrlistoffigures{\@listoffigures{lof}}%
63.420 \newcommand\lrrlistoftables{\@listoftables{lot}}%

```

`\lrrtableofcontents` Right-to-Left tables will be called `\lrrxxx` and defined with the aid of three macros `\lrrlistoffigures` defined above (extensions `cot`, `fol`, and `tol`).

```

\lrrlistoftables63.421 \newcommand\lrrtableofcontents{\@tableofcontents{cot}}%
63.422 \newcommand\lrrlistoffigures{\@listoffigures{fol}}%
63.423 \newcommand\lrrlistoftables{\@listoftables{tol}}%

```

`\tableofcontents` Let `\xxx` be `\lrrxxx` if the current direction is Right-to-Left and `\lrrxxx` if it is Left-to-Right.

```

\listoftables63.424 \renewcommand\tableofcontents{\if@rll\lrrtableofcontents%
63.425 \else\lrrtableofcontents\fi}
63.426 \renewcommand\listoffigures{\if@rll\lrrlistoffigures%
63.427 \else\lrrlistoffigures\fi}
63.428 \renewcommand\listoftables{\if@rll\lrrlistoftables%
63.429 \else\lrrlistoftables\fi}}

```

`\@dottedtocline` The following makes problems when making a Right-to-Left tables, since it uses `\leftskip` and `\rightskip` which are both mode dependent.

```

63.430 \def\@dottedtocline#1#2#3#4#5{%
63.431 \ifnum #1>\c@tocdepth \else
63.432 \vskip \z@ \@plus.2\p@
63.433 {\if@rll\rightskip\else\leftskip\fi #2\relax
63.434 \if@rll\leftskip\else\rightskip\fi \@tocrmarg \parfillskip
63.435 -\if@rll\leftskip\else\rightskip\fi
63.436 \parindent #2\relax\@afterindenttrue
63.437 \interlinepenalty\@M
63.438 \leavevmode
63.439 \@tempdima #3\relax
63.440 \advance\if@rll\rightskip\else\leftskip\fi \@tempdima
63.441 \null\nobreak\hskip -\if@rll\rightskip\else\leftskip\fi
63.442 {#4}\nobreak
63.443 \leaders\hbox{$\m@th
63.444 \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
63.445 mu$}\hfill
63.446 \nobreak
63.447 \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor \beginL#5\endL}%
63.448 \par}%
63.449 \fi}

```

`\l@part` This standard macro was redefined for table of contents since it uses `\rightskip` which is mode dependent.

```

63.450 \ifclassloaded{letter}{}{% other
63.451 \ifclassloaded{slides}{}{% other
63.452 \renewcommand*\l@part[2]{%
63.453   \ifnum \c@tocdepth >-2\relax
63.454     \addpenalty{-\@highpenalty}%
63.455     \addvspace{2.25em \@plus\p@}%
63.456     \begingroup
63.457       \setlength\@tempdima{3em}%
63.458       \parindent \z@ \if@r1\leftskip\else\rightskip\fi \@pnumwidth
63.459       \parfillskip -\@pnumwidth
63.460       {\leavevmode
63.461         \large \bfseries #1\hfil \hb@xt@\@pnumwidth{\hss#2}}\par
63.462         \nobreak
63.463         \global\@nobreaktrue
63.464         \everypar{\global\@nobreakfalse\everypar{}}%
63.465       \endgroup
63.466     \fi}}

```

`\@part` Part is redefined to support new Right-to-Left table of contents (cot) as well as the Left-to-Right one (toc).

```

63.467 \ifclassloaded{article}{% article class
63.468   \def\@part[#1]#2{%
63.469     \ifnum \c@secnumdepth >\m@ne
63.470       \refstepcounter{part}%
63.471       \addcontentsline{toc}{part}{\thepart\hspace{1em}#1}%
63.472       \addcontentsline{cot}{part}{\thepart\hspace{1em}#1}%
63.473     \else
63.474       \addcontentsline{toc}{part}{#1}%
63.475       \addcontentsline{cot}{part}{#1}%
63.476     \fi
63.477     {\parindent \z@ \raggedright
63.478       \interlinepenalty \@M
63.479       \normalfont
63.480       \ifnum \c@secnumdepth >\m@ne
63.481         \Large\bfseries \partname~\thepart
63.482         \par\nobreak
63.483       \fi
63.484       \huge \bfseries #2%
63.485       \markboth{}{}\par}%
63.486     \nobreak
63.487     \vskip 3ex
63.488     \@afterheading}%
63.489 }{% report and book classes
63.490   \def\@part[#1]#2{%
63.491     \ifnum \c@secnumdepth >-2\relax
63.492       \refstepcounter{part}%
63.493       \addcontentsline{toc}{part}{\thepart\hspace{1em}#1}%
63.494       \addcontentsline{cot}{part}{\thepart\hspace{1em}#1}%
63.495     \else
63.496       \addcontentsline{toc}{part}{#1}%
63.497       \addcontentsline{cot}{part}{#1}%
63.498     \fi
63.499     \markboth{}{}%
63.500     {\centering
63.501       \interlinepenalty \@M
63.502       \normalfont
63.503       \ifnum \c@secnumdepth >-2\relax
63.504         \huge\bfseries \partname~\thepart
63.505         \par
63.506         \vskip 20\p@

```

```

63.507 \fi
63.508 \Huge \bfseries #2\par}%
63.509 \@endpart}}

```

`\@sect` Section was redefined from the `latex.ltx` file. It is changed to support both Left-to-Right (toc) and Right-to-Left (cot) table of contents simultaneously.

```

63.510 \def\@sect#1#2#3#4#5#6[#7]#8{%
63.511 \ifnum #2>\c@secnumdepth
63.512 \let\@svsec\@empty
63.513 \else
63.514 \refstepcounter{#1}%
63.515 \protected@edef\@svsec{\@secntformat{#1}\relax}%
63.516 \fi
63.517 \@tempskipa #5\relax
63.518 \ifdim \@tempskipa>\z@
63.519 \begingroup
63.520 #6{%
63.521 \@hangfrom{\hskip #3\relax\@svsec}%
63.522 \interlinepenalty \@M #8\@@par}%
63.523 \endgroup
63.524 \csname #1mark\endcsname{#7}%
63.525 \addcontentsline{toc}{#1}{%
63.526 \ifnum #2>\c@secnumdepth \else
63.527 \protect\numberline{\csname the#1\endcsname}%
63.528 \fi
63.529 #7}%
63.530 \addcontentsline{cot}{#1}{%
63.531 \ifnum #2>\c@secnumdepth \else
63.532 \protect\numberline{\csname the#1\endcsname}%
63.533 \fi
63.534 #7}%
63.535 \else
63.536 \def\@svsechd{%
63.537 #6{\hskip #3\relax
63.538 \@svsec #8}%
63.539 \csname #1mark\endcsname{#7}%
63.540 \addcontentsline{toc}{#1}{%
63.541 \ifnum #2>\c@secnumdepth \else
63.542 \protect\numberline{\csname the#1\endcsname}%
63.543 \fi
63.544 #7}%
63.545 \addcontentsline{cot}{#1}{%
63.546 \ifnum #2>\c@secnumdepth \else
63.547 \protect\numberline{\csname the#1\endcsname}%
63.548 \fi
63.549 #7}}}%
63.550 \fi
63.551 \@xsect{#5}}

```

`\@caption` Caption was redefined from the `latex.ltx` file. It is changed to support Left-to-Right list of figures and list of tables (lof and lot) as well as new Right-to-Left lists (fol and to1) simultaneously.

```

63.552 \long\def\@caption#1[#2]#3{%
63.553 \par
63.554 \addcontentsline{\csname ext@#1\endcsname}{#1}%
63.555 {\protect\numberline{\csname the#1\endcsname}%
63.556 {\ignorespaces #2}}}%
63.557 \def\@fignm{figure}
63.558 \ifx#1\@fignm\addcontentsline{fol}{#1}%
63.559 {\protect\numberline{\csname the#1\endcsname}%
63.560 {\ignorespaces #2}}\fi%
63.561 \def\@tblnm{table}

```



```

63.562 \ifx#1\@tblnm\addcontentsline{tol}{#1}%
63.563 {\protect\numberline{\csname the#1\endcsname}%
63.564 {\ignorespaces #2}}\fi%
63.565 \begingroup
63.566 \@parboxrestore
63.567 \if@minipage
63.568 \@setminipage
63.569 \fi
63.570 \normalsize
63.571 \@makecaption{\csname fnum#1\endcsname}{\ignorespaces #3}\par
63.572 \endgroup

```

`\l@chapter` This standard macro was redefined for table of contents since it uses `\rightskip` which is mode dependent.

```

63.573 \@ifclassloaded{letter}{\}%
63.574 \@ifclassloaded{slides}{\}%
63.575 \@ifclassloaded{article}{\}%
63.576 \renewcommand*\l@chapter[2]{%
63.577 \ifnum \c@tocdepth >\m@ne
63.578 \addpenalty{-\@highpenalty}%
63.579 \vskip 1.0em \@plus\p@
63.580 \setlength\@tempdima{1.5em}%
63.581 \begingroup
63.582 \parindent \z@ \if@rl\leftskip\else\rightskip\fi \@pnumwidth
63.583 \parfillskip -\@pnumwidth
63.584 \leavevmode \bfseries
63.585 \advance\if@rl\rightskip\else\leftskip\fi\@tempdima
63.586 \hskip -\if@rl\rightskip\else\leftskip\fi
63.587 #1\nobreak\hfil \nobreak\hbext@\@pnumwidth{\hss#2}\par
63.588 \penalty\@highpenalty
63.589 \endgroup
63.590 \fi}}

```

`\l@section` The toc entry for section did not work in article style. Also it does not print dots, which is funny when most of your work is divided into sections.

`\l@subsection` It was revised to use `\@dottedtocline` as in `report.sty` (by Yaniv Bargury)

`\l@paragraph` and was updated later for all kinds of sections (by Boris Lavva).

```

\l@subparagraph63.591 \@ifclassloaded{article}{%
63.592 \renewcommand*\l@section{\@dottedtocline{1}{1.5em}{2.3em}}
63.593 \renewcommand*\l@subsection{\@dottedtocline{2}{3.8em}{3.2em}}
63.594 \renewcommand*\l@subsubsection{\@dottedtocline{3}{7.0em}{4.1em}}
63.595 \renewcommand*\l@paragraph{\@dottedtocline{4}{10em}{5em}}
63.596 \renewcommand*\l@subparagraph{\@dottedtocline{5}{12em}{6em}}\}

```

63.4.6 Two-column mode

This is the support of `twocolumn` option for the standard $\text{\LaTeX} 2_{\epsilon}$ classes. The following code was originally borrowed from the `ArabTeX` package, file `latexext.sty`, copyright by Klaus Lagally, Institut fuer Informatik, Universitaet Stuttgart. It was updated for this package by Boris Lavva.

```

\@outputdblcol First column is \leftcolumn will be shown at the right side, Second column is
\set@outputdblcol \@outputbox will be shown at the left side.
rl@outputdblcol \set@outputdblcol IS CURRENTLY DISABLED. TODO: REMOVE IT
[tzafrir]
63.597 \let\@outputdblcol\@outputdblcol
63.598 %\def\set@outputdblcol{%
63.599 % \if@rl\renewcommand{\@outputdblcol}{\rl@outputdblcol}%
63.600 % \else\renewcommand{\@outputdblcol}{\@outputdblcol}\fi}
63.601 \renewcommand{\@outputdblcol}{%
63.602 \if@rlmain%

```

```

63.603 \rl@outputdblcol%
63.604 \else%
63.605 \@@outputdblcol%
63.606 \fi%
63.607 }
63.608 \newcommand{\rl@outputdblcol}{%
63.609 \if@firstcolumn
63.610 \global \@firstcolumnfalse
63.611 \global \setbox\@leftcolumn \box\@outputbox
63.612 \else
63.613 \global \@firstcolumntrue
63.614 \setbox\@outputbox \vbox {\hb@xt@\textwidth {%
63.615 \hskip\columnwidth%
63.616 \hfil\vrule\@width\columnseprule\hfil
63.617 \hb@xt@\columnwidth {%
63.618 \box\@leftcolumn \hss}%
63.619 \hb@xt@\columnwidth {%
63.620 \hskip-\textwidth%
63.621 \box\@outputbox \hss}%
63.622 \hskip\columnsep%
63.623 \hskip\columnwidth}}%
63.624 \@combinedblfloats
63.625 \@outputpage
63.626 \begingroup
63.627 \@dblfloatplacement
63.628 \@startdblcolumn
63.629 \@whiles\if@fcolmade \fi
63.630 {\@outputpage
63.631 \@startdblcolumn}%
63.632 \endgroup
63.633 \fi}

```

63.4.7 Footnotes

\footnoterule The Right-to-Left footnote rule is simply reversed default Left-to-Right one. Footnotes can be used in RL or LR main modes, but changing mode while a footnote is pending is still unsolved.

```

63.634 \let\@@footnoterule=\footnoterule
63.635 \def\footnoterule{\if@rl\hb@xt@\hsize{\hss\vbox{\@@footnoterule}}%
63.636 \else\@@footnoterule\fi}

```

63.4.8 Headings and two-side support

When using **headings** or **myheadings** modes, we have to ensure that the language and direction of heading is the same as the whole chapter/part of the document. This is implementing by setting special variable **\headlanguage** when starting new chapter/part.

In addition, when selecting the **twoside** option (default in **book** document class), the LR and RL modes need to be set properly for things on the heading and footing. This is done here too.

ps@headings First, we will support the standard **letter** class:

```

ps@myheadings63.637 \@ifclassloaded{letter}{%
headeven63.638 \def\headodd{\protect\if@rl\beginR\fi\headtoname{}}
headodd63.639 \ignorespaces\toname
63.640 \hfil \@date
63.641 \hfil \pagename{} \thepage\protect\if@rl\endR\fi}
63.642 \if@twoside
63.643 \def\ps@headings{%
63.644 \let\@oddfoot\@empty\let\@evenfoot\@empty
63.645 \def\@oddhead{\select@language{\headlanguage}\headodd}

```

```

63.646         \let\@evenhead\@oddhead}
63.647 \else
63.648     \def\ps@headings{%
63.649         \let\@oddfoot\@empty
63.650         \def\@oddhead{\select@language{\headlanguage}\headodd}}
63.651 \fi
63.652 \def\headfirst{\protect\if@rl\beginR\fi\fromlocation \hfill %
63.653     \telephonenumber\protect\if@rl\endR\fi}
63.654 \def\ps@firstpage{%
63.655     \let\@oddhead\@empty
63.656     \def\@oddfoot{\raisebox{-45\p@}{\z@}{%
63.657         \hb@xt@{\textwidth}{\hspace*{100\p@}}%
63.658         \ifcase \@ptsize\relax
63.659             \normalsize
63.660         \or
63.661             \small
63.662         \or
63.663             \footnotesize
63.664         \fi
63.665         \select@language{\headlanguage}\headfirst}}\hss}}
63.666 %
63.667 \renewcommand{\opening}[1]{%
63.668     \let\headlanguage=\language%
63.669     \ifx\@empty\fromaddress%
63.670         \thispagestyle{firstpage}%
63.671         {\raggedleft\@date\par}%
63.672     \else % home address
63.673         \thispagestyle{empty}%
63.674         {\raggedleft
63.675             \if@rl\begin{tabular}{@{\beginR\csize}%
63.676                 to\@rllanguage\endcsname}r@{\endR}}\ignorespaces
63.677                 \fromaddress \*[2\parskip]%
63.678                 \@date \end{tabular}\par%
63.679             \else\begin{tabular}{l}\ignorespaces
63.680                 \fromaddress \*[2\parskip]%
63.681                 \@date \end{tabular}\par%
63.682             \fi}%
63.683         \fi
63.684         \vspace{2\parskip}%
63.685         {\raggedright \toname \ \ toaddress \par}%
63.686         \vspace{2\parskip}%
63.687         #1\par\nobreak}
63.688 }

```

Then, the article, report and book document classes are supported. Note, that in one-sided mode \markright was changed to \markboth.

```

63.689 {% article, report, book
63.690     \def\headeven{\protect\if@rl\beginR\thepage\hfil\rightmark\endR
63.691         \protect\else\thepage\hfil{\slshape\leftmark}
63.692         \protect\fi}
63.693     \def\headodd{\protect\if@rl\beginR\leftmark\hfil\thepage\endR
63.694         \protect\else{\slshape\rightmark}\hfil\thepage
63.695         \protect\fi}
63.696     \@ifclassloaded{article}{% article
63.697         \if@twoside % two-sided
63.698             \def\ps@headings{%
63.699                 \let\@oddfoot\@empty\let\@evenfoot\@empty
63.700                 \def\@evenhead{\select@language{\headlanguage}\headeven}%
63.701                 \def\@oddhead{\select@language{\headlanguage}\headodd}%
63.702                 \let\@mkboth\markboth
63.703                 \def\sectionmark##1{%
63.704                     \markboth {\MakeUppercase{#1}}
63.705                     \ifnum \c@secnumdepth >\z@

```

```

63.706         \thesection\quad
63.707     \fi
63.708     ##1}}{}}%
63.709     \def\subsectionmark##1{%
63.710         \markright{%
63.711             \ifnum \c@secnumdepth >\@ne
63.712                 \thesubsection\quad
63.713             \fi
63.714         ##1}}}}
63.715 \else         % one-sided
63.716     \def\ps@headings{%
63.717         \let\@oddfoot\@empty
63.718         \def\@oddhead{\headodd}%
63.719         \let\@mkboth\markboth
63.720         \def\sectionmark##1{%
63.721             \markboth{\MakeUppercase{%
63.722                 \ifnum \c@secnumdepth >\m@ne
63.723                     \thesection\quad
63.724                 \fi
63.725                 ##1}}{\MakeUppercase{%
63.726                     \ifnum \c@secnumdepth >\m@ne
63.727                         \thesection\quad
63.728                     \fi
63.729                     ##1}}}}
63.730     \fi
63.731 %
63.732     \def\ps@myheadings{%
63.733         \let\@oddfoot\@empty\let\@evenfoot\@empty
63.734         \def\@evenhead{\select@language{\headlanguage}\headeven}%
63.735         \def\@oddhead{\select@language{\headlanguage}\headodd}%
63.736         \let\@mkboth\@gobbletwo
63.737         \let\sectionmark\@gobble
63.738         \let\subsectionmark\@gobble
63.739     }}{% report and book
63.740     \if@twoside % two-sided
63.741         \def\ps@headings{%
63.742             \let\@oddfoot\@empty\let\@evenfoot\@empty
63.743             \def\@evenhead{\select@language{\headlanguage}\headeven}
63.744             \def\@oddhead{\select@language{\headlanguage}\headodd}
63.745             \let\@mkboth\markboth
63.746             \def\chaptermark##1{%
63.747                 \markboth{\MakeUppercase{%
63.748                     \ifnum \c@secnumdepth >\m@ne
63.749                         \@chapapp\ \thechapter. \ %
63.750                     \fi
63.751                     ##1}}{}}%
63.752             \def\sectionmark##1{%
63.753                 \markright {\MakeUppercase{%
63.754                     \ifnum \c@secnumdepth >\z@
63.755                         \thesection. \ %
63.756                     \fi
63.757                     ##1}}}}
63.758     \else % one-sided
63.759         \def\ps@headings{%
63.760             \let\@oddfoot\@empty
63.761             \def\@oddhead{\select@language{\headlanguage}\headodd}
63.762             \let\@mkboth\markboth
63.763             \def\chaptermark##1{%
63.764                 \markboth{\MakeUppercase{%
63.765                     \ifnum \c@secnumdepth >\m@ne
63.766                         \@chapapp\ \thechapter. \ %
63.767                     \fi

```

```

63.768      ##1}}{\MakeUppercase{%
63.769      \ifnum \c@secnumdepth >\m@ne
63.770      \@chapapp\ \thechapter. \ %
63.771      \fi
63.772      ##1}}}}
63.773 \fi
63.774 \def\ps@myheadings{%
63.775     \let\@oddfoot\@empty\let\@evenfoot\@empty
63.776     \def\@evenhead{\select@language{\headlanguage}\headeven}%
63.777     \def\@oddhead{\select@language{\headlanguage}\headodd}%
63.778     \let\@mkboth\@gobbletwo
63.779     \let\chaptermark\@gobble
63.780     \let\sectionmark\@gobble
63.781 }}}}

```

63.4.9 Postscript Porblems

Any command that is implemented by PostScript directives, e.g. commands from the `ps-tricks` package, needs to be fixed, because the PostScript directives are being interpreted after the document has been converted by $\text{T}_{\text{E}}\text{X}$ to visual Hebrew (DVI, PostScript and PDF have visual Hebrew).

For instance: Suppose you wrote in your document:

```
\textcolor{cyan}{some ltr text}
```

This would be interpreted by $\text{T}_{\text{E}}\text{X}$ to something like:

```
[postscript:make color cyan]some LTR text[postscript:make color black]
```

However, with the bidirectionality support we get:

```
\textcolor{cyan}{\hebalef\hebbet}
```

Translated to:

```
[postscript:make color black]{bet}{alef}[postscript:make color cyan]
```

While we want:

```
[postscript:make color cyan]{bet}{alef}[postscript:make color black]
```

The following code will probably work at least with code that stays in the same

line:

`@textcolor`

```

63.782 \AtBeginDocument{%
63.783   %I assume that \textcolor is only defined by the package color
63.784   \ifx\@textcolor\undefined\else%
63.785     % If that macro was defined before the beginning of the document,
63.786     % that is: the package was loaded: redefine it with bidi support
63.787     \def\@textcolor#1#2#3{%
63.788       \if@rl%
63.789         \beginL\protect\leavevmode{\color#1{#2}\beginR#3\endR}\endL%
63.790       \else%
63.791         \protect\leavevmode{\color#1{#2}#3}%
63.792       \fi%
63.793     }%
63.794   \fi%
63.795 }
63.796 % \end{macrocode}
63.797 % \end{macro}
63.798 % \begin{macro}{\thetrueSlideCounter}
63.799 %   This macro probably needs to be overridden for when using |prosper|,
63.800 %   (waiting for feedback. Tzafrir)
63.801 %   \begin{macrocode}
63.802 \@ifclassloaded{prosper}{%
63.803   \def\thetrueSlideCounter{\arabicnorl{trueSlideCounter}}
63.804 }{}

```

63.4.10 Miscellaneous internal L^AT_EX macros

`\raggedright` `\raggedright` was changed from `latex.ltx` file to support Right-to-Left mode, because of the bug in its implementation.

```
63.805 \def\raggedright{%
63.806   \let\\\@centercr
63.807   \leftskip\z@skip\rightskip\@flushglue
63.808   \parindent\z@\parfillskip\z@skip}
```

Swap meanings of `\raggedright` and `\raggedleft` in Right-to-Left mode.

```
63.809 \let\@raggedleft=\raggedleft
63.810 \let\@raggedright=\raggedright
63.811 \renewcommand\raggedleft{\if@rl\@raggedright%
63.812                               \else\@raggedleft\fi}
63.813 \renewcommand\raggedright{\if@rl\@raggedleft%
63.814                               \else\@raggedright\fi}
```

`\author` `\author` is inserted with `tabular` environment, and will be used in restricted horizontal mode. Therefore we have to add explicit direction change command when in Right-to-Left mode.

```
63.815 \let\@author=\author
63.816 \renewcommand{\author}[1]{\@author{\if@rl\beginR #1\endR\else #1\fi}}
```

`\MakeUppercase` There are no uppercase and lowercase letters in most Right-to-Left languages, therefore we should redefine `\MakeUppercase` and `\MakeLowercase` L^AT_EX 2_ε commands.

```
63.817 \let\@MakeUppercase=\MakeUppercase
63.818 \def\MakeUppercase#1{\if@rl#1\else\@MakeUppercase{#1}\fi}
63.819 \let\@MakeLowercase=\MakeLowercase
63.820 \def\MakeLowercase#1{\if@rl#1\else\@MakeLowercase{#1}\fi}
```

`\underline` We should explicitly use `\L` and `\R` commands in `\underlined` text.

```
63.821 \let\@@underline=\underline
63.822 \def\underline#1{\@@underline{\if@rl\R{#1}\else #1\fi}}
```

`\undertext` was added for L^AT_EX 2.09 compatibility mode.

```
63.823 \ifcompatibility
63.824   \let\undertext=\underline
63.825 \fi
```

`\@xnthm` The following has been inserted to correct the appearance of the number in `\@opargbegintheorem` `\newtheorem` to reorder theorem number components. A similar correction in the definition of `\@opargbegintheorem` was added too.

```
63.826 \def\@xnthm#1#2[#3]{%
63.827   \expandafter\@ifdefinable\csname #1\endcsname
63.828   {\@definecounter{#1}\@addtoreset{#1}{#3}%
63.829     \expandafter\xdef\csname the#1\endcsname{\noexpand\@number
63.830       {\expandafter\noexpand\csname the#3\endcsname \@thmcountersep
63.831         \@thmcounter{#1}}}%
63.832     \global\@namedef{#1}{\@thm{#1}{#2}}%
63.833     \global\@namedef{end#1}{\@endtheorem}}%
63.834 %
63.835 \def\@opargbegintheorem#1#2#3{%
63.836   \trivlist
63.837     \item[\hskip \labelsep{\bfseries #1\ #2\
63.838       \@brackets{#3}}]{\itshape}
```

`\@chapter` The following was added for pretty printing of the chapter numbers, for supporting Right-to-Left tables (`cot`, `fol`, and `tol`), to save `\headlanguage` for use in running headers, and to start two-column mode depending on chapter's main language.

```
63.839 \ifclassloaded{article}{-}{%
```

```

63.840 % For pretty printing
63.841 \def\@chapapp{Chapter}
63.842 \def\@thechapter{\@arabic\c@chapter}
63.843 \def\@chapter[#1]#2{%
63.844   \let\headlanguage=\language%
63.845   %\set@outputdblcol%
63.846   \ifnum \c@secnumdepth >\m@ne
63.847     \refstepcounter{chapter}%
63.848     \typeout{\@chapapp\space\@thechapter.}%
63.849     \addcontentsline{toc}{chapter}%
63.850     {\protect\numberline{\thechapter}#1}
63.851     \addcontentsline{cot}{chapter}%
63.852     {\protect\numberline{\thechapter}#1}
63.853   \else
63.854     \addcontentsline{toc}{chapter}{#1}%
63.855     \addcontentsline{cot}{chapter}{#1}%
63.856   \fi
63.857   \chaptermark{#1}
63.858   \addtocontents{lof}{\protect\addvspace{10\p@}}%
63.859   \addtocontents{fol}{\protect\addvspace{10\p@}}%
63.860   \addtocontents{lot}{\protect\addvspace{10\p@}}%
63.861   \addtocontents{tol}{\protect\addvspace{10\p@}}%
63.862   \if@twocolumn
63.863     \topnewpage[\@makechapterhead{#2}]%
63.864   \else
63.865     \@makechapterhead{#2}%
63.866     \@afterheading
63.867   \fi}
63.868 %
63.869 \def\@schapter#1{%
63.870   \let\headlanguage=\language%
63.871   %\set@outputdblcol%
63.872   \if@twocolumn
63.873     \topnewpage[\@makeschapterhead{#1}]%
63.874   \else
63.875     \@makeschapterhead{#1}%
63.876     \@afterheading
63.877   \fi}

```

`\appendix` Changed mainly for pretty printing of appendix numbers, and to start two-column mode with the right language (if needed).

```

63.878 \@ifclassloaded{letter}{\other
63.879 \@ifclassloaded{slides}{\other
63.880 \@ifclassloaded{article}{\article
63.881   \renewcommand\appendix{\par
63.882     \setcounter{section}{0}%
63.883     \setcounter{subsection}{0}%
63.884     \renewcommand\thesection{\@Alph\c@section}}
63.885 }{\report and book
63.886   \renewcommand\appendix{\par
63.887     %\set@outputdblcol%
63.888     \setcounter{chapter}{0}%
63.889     \setcounter{section}{0}%
63.890     \renewcommand\@chapapp{\appendixname}%
63.891     % For pretty printing
63.892     \def\@chapapp{Appendix}%
63.893     \def\@thechapter{\@Alph\c@chapter}
63.894     \renewcommand\thechapter{\@Alph\c@chapter}}}}

```

63.4.11 Bibliography and citations

`\cite` Citations are produced by the macro `\cite{LABEL}{NOTE}`. Both the citation label and the note is typeset in the current direction. We have to use `\@brackets`
`\@biblabel`
`\@lbibitem`

macro in `\@cite` and `\@biblabel` macros. In addition, when using *alpha* or similar bibliography style, the `\@lbibitem` is used and have to be update to support bot Right-to-Left and Left-to-Right citations.

```

63.895 \def\@cite#1#2{\@brackets[#1]{\if@tempw , #2\fi}}
63.896 \def\@biblabel#1{\@brackets[#1]}
63.897 \def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
63.898     {\let\protect\noexpand
63.899     \immediate
63.900     \if@rl\write\@auxout{\string\bibcite{#2}{\R{#1}}}%
63.901     \else\write\@auxout{\string\bibcite{#2}{\L{#1}}}\fi%
63.902     }\fi\ignorespaces}

```

`thebibliography` Use `\rightmargin` instead of `\leftmargin` when in RL mode.

```

63.903 \@ifclassloaded{letter}{\% other
63.904 \@ifclassloaded{slides}{\% other
63.905 \@ifclassloaded{article}{\%
63.906 \renewenvironment{thebibliography}[1]
63.907 {\section*{\refname\@mkboth%
63.908     {\MakeUppercase\refname}%
63.909     {\MakeUppercase\refname}}%
63.910 \list{\@biblabel{\@arabic\c@enumiv}}%
63.911 {\settowidth\labelwidth{\@biblabel{#1}}%
63.912 \if@rl\leftmargin\else\rightmargin\fi\labelwidth
63.913 \advance\if@rl\leftmargin\else\rightmargin\fi\labelsep
63.914 \@openbib@code
63.915 \usecounter{enumiv}%
63.916 \let\p@enumiv\@empty
63.917 \renewcommand\theenumiv{\@arabic\c@enumiv}}%
63.918 \sloppy
63.919 \clubpenalty4000
63.920 \@clubpenalty \clubpenalty
63.921 \widowpenalty4000%
63.922 \sfcode'\.\@m}
63.923 {\def\@noitemerr
63.924 {\@latex@warning{Empty 'thebibliography' environment}}}%
63.925 \endlist}}%
63.926 {\renewenvironment{thebibliography}[1]{\%
63.927 \chapter*{\bibname\@mkboth%
63.928     {\MakeUppercase\bibname}%
63.929     {\MakeUppercase\bibname}}%
63.930 \list{\@biblabel{\@arabic\c@enumiv}}%
63.931 {\settowidth\labelwidth{\@biblabel{#1}}%
63.932 \if@rl\leftmargin\else\rightmargin\fi\labelwidth
63.933 \advance\if@rl\leftmargin\else\rightmargin\fi\labelsep
63.934 \@openbib@code
63.935 \usecounter{enumiv}%
63.936 \let\p@enumiv\@empty
63.937 \renewcommand\theenumiv{\@arabic\c@enumiv}}%
63.938 \sloppy
63.939 \clubpenalty4000
63.940 \@clubpenalty \clubpenalty
63.941 \widowpenalty4000%
63.942 \sfcode'\.\@m}
63.943 {\def\@noitemerr
63.944 {\@latex@warning{Empty 'thebibliography' environment}}}%
63.945 \endlist}}}}

```

`\@verbatim` All kinds of verbs (`\verb`, `\verb*`, `verbatim` and `verbatim*`) now can be used in Right-to-Left mode. Errors in latin mode solved too.

```

63.946 \def\@verbatim{%
63.947 \let\do\@makeother \dospecials%
63.948 \obeylines \verbatim@font \@noligs}

```


`\@makecaption` Captions are set always centered. This allows us to use bilingual captions, for example: `\caption{\R{RLtext}} \L{LRtext}`, which will be formatted as:

Right to left caption here (RLtext)
Left to right caption here (LRtext)

See also `\bcaption` command below.

```
63.949 \long\def\@makecaption#1#2{%
63.950   \vskip\abovecaptionskip%
63.951   \begin{center}%
63.952     #1: #2%
63.953   \end{center} \par%
63.954   \vskip\belowcaptionskip}
```

63.4.12 Additional bidirectional commands

- Section headings are typeset with the default global direction.
- Text in section headings in the reverse language *do not* have to be protected for the reflection command, as in: `\protect\L{Latin Text}`, because `\L` and `\R` are robust now.
- Table of contents, list of figures and list of tables should be typeset with the `\tableofcontents`, `\listoffigures` and `\listoftables` commands respectively.
- The above tables will be typeset in the main direction (and language) in effect where the above commands are placed.
- Only 2 tables of each kind are supported: one for Right-to-Left and another for Left-to-Right directions.

How to include line to both tables? One has to use bidirectional sectioning commands as following:

1. Use the `\bxxx` version of the sectioning commands in the text instead of the `\xxx` version (`xxx` is one of: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `caption`).
2. Syntax of the `\bxxx` command is `\bxxx{RL text}{LR text}`. Both arguments are typeset in proper direction by default (no need to change direction for the text inside).
3. The section header inside the document will be typeset in the global direction in effect at the time. i.e. The `{RL text}` will be typeset if Right-to-Left mode is in effect and `{LR text}` otherwise.

`\bpart`

```
63.955 \newcommand{\bpart}[2]{\part{\protect\if@rl%
63.956   #1 \protect\else #2 \protect\fi}}
```

`\bchapter`

```
63.957 \newcommand{\bchapter}[2]{\chapter{\protect\if@rl%
63.958   #1 \protect\else #2 \protect\fi}}
```

`\bsection`

```
63.959 \newcommand{\bsection}[2]{\section{\protect\if@rl%
63.960   #1 \protect\else #2 \protect\fi}}
```

`\bsubsection`

```
63.961 \newcommand{\bsubsection}[2]{\subsection{\protect\if@rl%
63.962   #1 \protect\else #2 \protect\fi}}
```

`\bsubsubsection`

```
63.963 \newcommand{\bsubsubsection}[2]{\subsubsection{\protect\if@rl%
63.964   #1 \protect\else #2 \protect\fi}}
```

`\bcaption`

```
63.965 \newcommand{\bcaption}[2]{%
63.966   \caption[\protect\if@rl \R{#1}\protect\else \L{#2}\protect\fi]{%
63.967     \if@rl\R{#1}\protect\ \L{#2}
63.968     \else\L{#2}\protect\ \R{#1}\fi}}
```

The following definition is a modified version of `\bchapter`, meant as a bilingual twin for `\chapter*` and `\section*` (added by Irina Abramovici).

`\bchapternn`

```
63.969 \newcommand{\bchapternn}[2]{\chapter*{\protect\if@rl%
63.970   #1 \protect\else #2 \protect\fi}}
```

`\bsectionnn`

```
63.971 \newcommand{\bsectionnn}[2]{\section*{\protect\if@rl%
63.972   #1 \protect\else #2 \protect\fi}}
```

Finally, at end of `babel` package, the `\headlanguage` and two-column mode will be initialized according to the current language.

```
63.973 \AtEndOfPackage{\rlAtEndOfPackage}
63.974 %
63.975 \def\rlAtEndOfPackage{%
63.976   \global\let\headlanguage=\language\set@outputdblcol%
63.977 }
63.978 </rightleft>
```

63.5 Hebrew calendar

The original version of the package `hebc`⁷⁵ for $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ and $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2.09$, entitled “ $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ & $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ macros for computing Hebrew date from Gregorian one” was created by Michail Rozman, misha@iop.tartu.ew.su⁷⁶

Released: Tammuz 12, 5751–June 24, 1991
 Corrected: Shebat 10, 5752–January 15, 1992 by Rama Porrat
 Corrected: Adar II 5, 5752–March 10, 1992 by Misha
 Corrected: Tebeth, 5756–January 1996 Dan Haran
 (haran@math.tau.ac.il)

The package was adjusted for `babel` and $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2_{\epsilon}$ by Boris Lavva.

Changes to the printing routine (only) by Ron Artstein, June 1, 2003.

This package should be included *after* the `babel` with `hebrew` option, as following:

```
\documentclass[...]{...}
\usepackage[hebrew,...,other languages,...]{babel}
\usepackage{hebc}
```

Two main user-level commands are provided by this package:

`\Hebrewtoday`

Computes today’s Hebrew date and prints it. If we are presently in Hebrew mode, the date will be printed in Hebrew, otherwise — in English (like Shebat 10, 5752).

`\Hebrewdate`

Computes the Hebrew date from the given Gregorian date and prints it. If we are presently in Hebrew mode, the date will be printed in Hebrew, otherwise — in English (like Shebat 10, 5752). An example of usage is shown below:

⁷⁵The following description of `hebc` package is based on the comments included with original source by the author, Michail Rozman.

⁷⁶Please direct any comments, bug reports, questions, etc. about the package to this address.

```

\newcount\hd \newcount\hm \newcount\hy
\hd=10 \hm=3 \hy=1992
\Hebrewdate{\hd}{\hm}{\hy}

```

full The package option `full` sets the flag `\@full@hebrew@year`, which causes years from the current millenium to be printed with the thousands digit (he-tav-shin-samekh-gimel). Without this option, thousands are not printed for the current millenium. NOTE: should this be a command option rather than a package option? –RA.

63.5.1 Introduction

The Hebrew calendar is inherently complicated: it is lunisolar – each year starts close to the autumn equinox, but each month must strictly start at a new moon. Thus Hebrew calendar must be harmonized simultaneously with both lunar and solar events. In addition, for reasons of the religious practice, the year cannot start on Sunday, Wednesday or Friday.

For the full description of Hebrew calendar and for the list of references see:

Nachum Dershowitz and Edward M. Reingold, “*Calendarical Calculations*”, Software–Pract.Exper., vol. 20 (9), pp.899–928 (September 1990).

C translation of LISP programs from the above article available from Mr. Wayne Geiser, `geiser%pictel@uunet.uu.net`.

The 4th distribution (July 1989) of `hdate/hcal` (Hebrew calendar programs similar to UNIX `date/cal`) by Mr. Amos Shapir, `amos@shum.huji.ac.il`, contains short and very clear description of algorithms.

63.5.2 Registers, Commands, Formatting Macros

The command `\Hebrewtoday` produces today’s date for Hebrew calendar. It is similar to the standard L^AT_EX 2_ε command `\today`. In addition three numerical registers `\Hebrewday`, `\Hebrewmonth` and `\Hebrewyear` are set. For setting this registers without producing of date string command `\Hebrewsetreg` can be used.

The command `\Hebrewdate{Gday}{Gmonth}{Gyear}` produces Hebrew calendar date corresponding to Gregorian date `Gday.Gmonth.Gyear`. Three numerical registers `\Hebrewday`, `\Hebrewmonth` and `\Hebrewyear` are set.

For converting arbitrary Gregorian date `Gday.Gmonth.Gyear` to Hebrew date `Hday.Hmonth.Hyear` without producing date string the command:

```
\HebrewFromGregorian{Gday}{Gmonth}{Gyear}{Hday}{Hmonth}{Hyear}
```

can be used.

```

63.979 (*calendar)
63.980 \newif\if@full@hebrew@year
63.981 \@full@hebrew@yearfalse
63.982 \DeclareOption{full}{\@full@hebrew@yeartrue}
63.983 \ProcessOptions
63.984 \newcount\Hebrewday \newcount\Hebrewmonth \newcount\Hebrewyear

```

\Hebrewdate Hebrew calendar date corresponding to Gregorian date `Gday.Gmonth.Gyear`. If Hebrew (right-to-left) fonts & macros are not loaded, we have to use English format.

```

63.985 \def\Hebrewdate#1#2#3{%
63.986   \HebrewFromGregorian{#1}{#2}{#3}
63.987   {\Hebrewday}{\Hebrewmonth}{\Hebrewyear}%
63.988   \ifundefined{if@rl}%
63.989     \FormatForEnglish{\Hebrewday}{\Hebrewmonth}{\Hebrewyear}%
63.990   \else%
63.991     \FormatDate{\Hebrewday}{\Hebrewmonth}{\Hebrewyear}%
63.992   \fi}

```

`\Hebrewtoday` Today's date in Hebrew calendar.

```
63.993 \def\Hebrewtoday{\Hebrewdate{\day}{\month}{\year}}
63.994 \let\hebrewtoday=\Hebrewtoday
```

`\Hebrewsetreg` Set registers: today's date in hebrew calendar.

```
63.995 \def\Hebrewsetreg{%
63.996     \HebrewFromGregorian{\day}{\month}{\year}
63.997     {\Hebrewday}{\Hebrewmonth}{\Hebrewyear}}
```

`\FormatDate` Prints a Hebrew calendar date `Hebrewday.Hebrewmonth.Hebrewyear`.

```
63.998 \def\FormatDate#1#2#3{%
63.999     \if@r1%
63.1000         \FormatForHebrew{#1}{#2}{#3}%
63.1001     \else%
63.1002         \FormatForEnglish{#1}{#2}{#3}
63.1003     \fi}
```

To prepare another language version of Hebrew calendar commands, one should change or add commands here.

We start with Hebrew language macros.

`\HebrewYearName` Prints Hebrew year as a Hebrew number. Disambiguates strings by adding lamed-pe-gimel to years of the first Jewish millenium and to years divisible by 1000. Suppresses the thousands digit in the current millenium unless the package option `full` is selected. NOTE: should this be provided as a command option rather than a package option? -RA.

```
63.1004 \def\HebrewYearName#1{%
63.1005     \@tempcnta=#1\divide\@tempcnta by 1000\multiply\@tempcnta by 1000
63.1006     \ifnum#1=\@tempcnta\relax % divisible by 1000: disambiguate
63.1007         \Hebrewnumeralfinal{#1}\ )\heblamed\hebpe"\hebgimel(
63.1008     \else % not divisible by 1000
63.1009         \ifnum#1<1000\relax % first millennium: disambiguate
63.1010             \Hebrewnumeralfinal{#1}\ )\heblamed\hebpe"\hebgimel(
63.1011         \else
63.1012             \ifnum#1<5000
63.1013                 \Hebrewnumeralfinal{#1}%
63.1014             \else
63.1015                 \ifnum#1<6000 % current millenium, print without thousands
63.1016                     \@tempcnta=#1\relax
63.1017                     \if@full@hebrew@year\else\advance\@tempcnta by -5000\fi
63.1018                     \Hebrewnumeralfinal{\@tempcnta}%
63.1019                 \else % #1>6000
63.1020                     \Hebrewnumeralfinal{#1}%
63.1021                 \fi
63.1022             \fi
63.1023         \fi
63.1024     \fi}}
```

`\HebrewMonthName` The macro `\HebrewMonthName{month}{year}` returns the name of month in the 'year'.

```
63.1025 \def\HebrewMonthName#1#2{%
63.1026     \ifnum #1 = 7 %
63.1027         \CheckLeapHebrewYear{#2}%
63.1028         \if@HebrewLeap \hebalef\hebdalet\hebresh\ \hebbet'
63.1029         \else \hebalef\hebdalet\hebresh%
63.1030     \fi%
63.1031 \else%
63.1032     \ifcase#1%
63.1033         % nothing for 0
63.1034         \or\hebtav\hebshin\hebresh\hebyod%
63.1035         \or\hebbet\hebshin\hebvav\hebfinalnun%
63.1036         \or\hebkaf\hebsamekh\heblamed\hebvav%
```

```

63.1037      \or\hebtet\hebbet\hebtav%
63.1038      \or\hebshin\hebbet\hebtet%
63.1039      \or\hebalef\hebdalet\hebresh\ \hebalef'%
63.1040      \or\hebalef\hebdalet\hebresh\ \hebbet'%
63.1041      \or\hebnun\hebyod\hebsamekh\hebfinalnun%
63.1042      \or\hebalef\hebyod\hebyod\hebresh%
63.1043      \or\hebsamekh\hebyod\hebvav\hebfinalnun%
63.1044      \or\hebtav\hebmeme\hebvav\hebzayin%
63.1045      \or\hebalef\hebbet%
63.1046      \or\hebalef\heblamed\hebvav\heblamed%
63.1047      \fi%
63.1048      \fi}

```

\HebrewDayName Name of day in Hebrew letters (gimatria).

```
63.1049 \def\HebrewDayName#1{\Hebrewnumeral{#1}}
```

\FormatForHebrew The macro **\FormatForHebrew{*hday*}{*hmonth* }{*hyear*}** returns the formatted Hebrew date in Hebrew language.

```

63.1050 \def\FormatForHebrew#1#2#3{%
63.1051   \HebrewDayName{#1}~\hebbet\HebrewMonthName{#2}{#3},~%
63.1052   \HebrewYearName{#3}}

```

We continue with two English language macros for Hebrew calendar.

\HebrewMonthNameInEnglish The macro **\HebrewMonthNameInEnglish{*month*}{*year*}** is similar to **\HebrewMonthName** described above. It returns the name of month in the Hebrew ‘year’ in English.

```

63.1053 \def\HebrewMonthNameInEnglish#1#2{%
63.1054   \ifnum #1 = 7%
63.1055     \CheckLeapHebrewYear{#2}%
63.1056     \if@HebrewLeap Adar II\else Adar\fi%
63.1057   \else%
63.1058     \ifcase #1%
63.1059       % nothing for 0
63.1060       \or Tishrei%
63.1061       \or Heshvan%
63.1062       \or Kislev%
63.1063       \or Tebeth%
63.1064       \or Shebat%
63.1065       \or Adar I%
63.1066       \or Adar II%
63.1067       \or Nisan%
63.1068       \or Iyar%
63.1069       \or Sivan%
63.1070       \or Tammuz%
63.1071       \or Av%
63.1072       \or Elul%
63.1073     \fi
63.1074   \fi}

```

\FormatForEnglish The macro **\FormatForEnglish{*hday*}{*hmonth* }{*hyear*}** is similar to **\FormatForHebrew** macro described above and returns the formatted Hebrew date in English.

```

63.1075 \def\FormatForEnglish#1#2#3{%
63.1076   \HebrewMonthNameInEnglish{#2}{#3} \number#1,\ \number#3}

```

63.5.3 Auxiliary Macros

```
63.1077 \newcount\@common
```

\Remainder **\Remainder{*a*}{*b*}{*c*}** calculates $c = a \% b == a - b \times \frac{a}{b}$

```
63.1078 \def\Remainder#1#2#3{%
```

```

63.1079      #3 = #1%                % c = a
63.1080      \divide #3 by #2%        % c = a/b
63.1081      \multiply #3 by -#2%      % c = -b(a/b)
63.1082      \advance #3 by #1}%      % c = a - b(a/b)

```

```

63.1083 \newif\if@Divisible

```

`\CheckIfDivisible` `\CheckIfDivisible{a}{b}` sets `\@Divisibletrue` if $a\%b == 0$

```

63.1084 \def\CheckIfDivisible#1#2{%
63.1085   {%
63.1086     \countdef\tmp = 0% \tmp == \count0 - temporary variable
63.1087     \Remainder{#1}{#2}{\tmp}%
63.1088     \ifnum \tmp = 0%
63.1089       \global\@Divisibletrue%
63.1090     \else%
63.1091       \global\@Divisiblefalse%
63.1092     \fi}%

```

`\ifundefined` From the \TeX book, ex. 7.7:

```

\ifundefined{command}<true text>\else<false text>\fi

```

```

63.1093 \def\ifundefined#1{\expandafter\ifx\csname#1\endcsname\relax}

```

63.5.4 Gregorian Part

```

63.1094 \newif\if@GregorianLeap

```

`\IfGregorianLeap` Conditional which is true if Gregorian ‘year’ is a leap year: $((year\%4 == 0) \wedge (year\%100 \neq 0)) \vee (year\%400 == 0)$

```

63.1095 \def\IfGregorianLeap#1{%
63.1096   \CheckIfDivisible{#1}{4}%
63.1097   \if@Divisible%
63.1098     \CheckIfDivisible{#1}{100}%
63.1099     \if@Divisible%
63.1100       \CheckIfDivisible{#1}{400}%
63.1101       \if@Divisible%
63.1102         \@GregorianLeaptrue%
63.1103       \else%
63.1104         \@GregorianLeapfalse%
63.1105       \fi%
63.1106     \else%
63.1107       \@GregorianLeaptrue%
63.1108     \fi%
63.1109   \else%
63.1110     \@GregorianLeapfalse%
63.1111   \fi%
63.1112 \if@GregorianLeap}

```

`\GregorianDaysInPriorMonths` The macro `\GregorianDaysInPriorMonths{month}{year}{days}` calculates the number of days in months prior to ‘month’ in the ‘year’.

```

63.1113 \def\GregorianDaysInPriorMonths#1#2#3{%
63.1114   {%
63.1115     #3 = \ifcase #1%
63.1116       0 \or%                % no month number 0
63.1117       0 \or%
63.1118       31 \or%
63.1119       59 \or%
63.1120       90 \or%
63.1121       120 \or%
63.1122       151 \or%
63.1123       181 \or%
63.1124       212 \or%

```

```

63.1125          243 \or%
63.1126          273 \or%
63.1127          304 \or%
63.1128          334%
63.1129      \fi%
63.1130      \IfGregorianLeap{#2}%
63.1131          \ifnum #1 > 2%          % if month after February
63.1132              \advance #3 by 1% % add leap day
63.1133      \fi%
63.1134      \fi%
63.1135      \global\@common = #3}%
63.1136      #3 = \@common}

```

\GregorianDaysInPriorYears The macro `\GregorianDaysInPriorYears{year}{days}` calculates the number of days in years prior to the ‘year’.

```

63.1137 \def\GregorianDaysInPriorYears#1#2{%
63.1138     {%
63.1139         \countdef\tmpc = 4%          % \tmpc==\count4
63.1140         \countdef\tmpb = 2%          % \tmpb==\count2
63.1141         \tmpb = #1%                  %
63.1142         \advance \tmpb by -1%          %
63.1143         \tmpc = \tmpb%                % \tmpc = \tmpb = year-1
63.1144         \multiply \tmpc by 365%       % Days in prior years =
63.1145         #2 = \tmpc%                  % = 365*(year-1) ...
63.1146         \tmpc = \tmpb%                %
63.1147         \divide \tmpc by 4%           % \tmpc = (year-1)/4
63.1148         \advance #2 by \tmpc%         % ... plus Julian leap days ...
63.1149         \tmpc = \tmpb%                %
63.1150         \divide \tmpc by 100%          % \tmpc = (year-1)/100
63.1151         \advance #2 by -\tmpc%         % ... minus century years ...
63.1152         \tmpc = \tmpb%                %
63.1153         \divide \tmpc by 400%          % \tmpc = (year-1)/400
63.1154         \advance #2 by \tmpc%         % ... plus 4-century years.
63.1155         \global\@common = #2}%
63.1156     #2 = \@common}

```

\AbsoluteFromGregorian The macro `\AbsoluteFromGregorian{day}{month}{year}{absdate}` calculates the absolute date (days since 01.01.0001) from Gregorian date `day.month.year`.

```

63.1157 \def\AbsoluteFromGregorian#1#2#3#4{%
63.1158     {%
63.1159         \countdef\tmpd = 0%          % \tmpd==\count0
63.1160         #4 = #1%                    % days so far this month
63.1161         \GregorianDaysInPriorMonths{#2}{#3}{\tmpd}%
63.1162         \advance #4 by \tmpd%         % add days in prior months
63.1163         \GregorianDaysInPriorYears{#3}{\tmpd}%
63.1164         \advance #4 by \tmpd%         % add days in prior years
63.1165         \global\@common = #4}%
63.1166     #4 = \@common}

```

63.5.5 Hebrew Part

```

63.1167 \newif\if@HebrewLeap

```

\CheckLeapHebrewYear Set `\@HebrewLeaptrue` if Hebrew ‘year’ is a leap year, i.e. if $(1 + 7 \times \text{year}) \% 19 < 7$ then *true* else *false*

```

63.1168 \def\CheckLeapHebrewYear#1{%
63.1169     {%
63.1170         \countdef\tmpa = 0%          % \tmpa==\count0
63.1171         \countdef\tmpb = 1%          % \tmpb==\count1
63.1172     %
63.1173         \tmpa = #1%
63.1174         \multiply \tmpa by 7%

```

```

63.1175      \advance \tmpa by 1%
63.1176      \Remainder{\tmpa}{19}{\tmpb}%
63.1177      \ifnum \tmpb < 7%          % \tmpb = (7*year+1)%19
63.1178          \global\@HebrewLeaptrue%
63.1179      \else%
63.1180          \global\@HebrewLeapfalse%
63.1181      \fi}}

```

\HebrewElapsedMonths The macro `\HebrewElapsedMonths{year}{months}` determines the number of months elapsed from the Sunday prior to the start of the Hebrew calendar to the mean conjunction of Tishri of Hebrew ‘year’.

```

63.1182 \def\HebrewElapsedMonths#1#2{%
63.1183     {%
63.1184         \countdef\tmpa = 0%          % \tmpa==\count0
63.1185         \countdef\tmpb = 1%          % \tmpb==\count1
63.1186         \countdef\tmpc = 2%          % \tmpc==\count2
63.1187 %
63.1188         \tmpa = #1%                  %
63.1189         \advance \tmpa by -1%         %
63.1190         #2 = \tmpa%                  % #2 = \tmpa = year-1
63.1191         \divide #2 by 19%             % Number of complete Meton cycles
63.1192         \multiply #2 by 235%          % #2 = 235*((year-1)/19)
63.1193 %
63.1194         \Remainder{\tmpa}{19}{\tmpb}% \tmpa = years%19-years this cycle
63.1195         \tmpc = \tmpb%                %
63.1196         \multiply \tmpb by 12%         %
63.1197         \advance #2 by \tmpb%         % add regular months this cycle
63.1198 %
63.1199         \multiply \tmpc by 7%          %
63.1200         \advance \tmpc by 1%          %
63.1201         \divide \tmpc by 19%          % \tmpc = (1+7*((year-1)%19))/19 -
63.1202 %                                % number of leap months this cycle
63.1203         \advance #2 by \tmpc%         % add leap months
63.1204 %
63.1205         \global\@common = #2}%
63.1206     #2 = \@common}

```

\HebrewElapsedDays The macro `\HebrewElapsedDays{year}{days}` determines the number of days elapsed from the Sunday prior to the start of the Hebrew calendar to the mean conjunction of Tishri of Hebrew ‘year’.

```

63.1207 \def\HebrewElapsedDays#1#2{%
63.1208     {%
63.1209         \countdef\tmpa = 0%          % \tmpa==\count0
63.1210         \countdef\tmpb = 1%          % \tmpb==\count1
63.1211         \countdef\tmpc = 2%          % \tmpc==\count2
63.1212 %
63.1213         \HebrewElapsedMonths{#1}{#2}%
63.1214         \tmpa = #2%                  %
63.1215         \multiply \tmpa by 13753%     %
63.1216         \advance \tmpa by 5604%       % \tmpa=MonthsElapsed*13758 + 5604
63.1217         \Remainder{\tmpa}{25920}{\tmpc}% \tmpc == ConjunctionParts
63.1218         \divide \tmpa by 25920%
63.1219 %
63.1220         \multiply #2 by 29%
63.1221         \advance #2 by 1%
63.1222         \advance #2 by \tmpa%         % #2 = 1 + MonthsElapsed*29 +
63.1223 %                                % PartsElapsed/25920
63.1224         \Remainder{#2}{7}{\tmpa}%    % \tmpa == DayOfWeek
63.1225         \ifnum \tmpc < 19440%
63.1226             \ifnum \tmpc < 9924%
63.1227                 \else%                % New moon at 9 h. 204 p. or later
63.1228                 \ifnum \tmpa = 2%    % on Tuesday ...

```



```

63.1229             \CheckLeapHebrewYear{#1}% of a common year
63.1230             \if@HebrewLeap%
63.1231             \else%
63.1232                 \advance #2 by 1%
63.1233             \fi%
63.1234         \fi%
63.1235     \fi%
63.1236     \ifnum \tmpc < 16789%
63.1237     \else%
63.1238         \ifnum \tmpa = 1% % New moon at 15 h. 589 p. or later
63.1239             \advance #1 by -1%
63.1240             \CheckLeapHebrewYear{#1}% at the end of leap year
63.1241             \if@HebrewLeap%
63.1242                 \advance #2 by 1%
63.1243             \fi%
63.1244         \fi%
63.1245     \fi%
63.1246 \else%
63.1247     \advance #2 by 1% % new moon at or after midday
63.1248 \fi%
63.1249 %
63.1250 \Remainder{#2}{7}{\tmpa}% % \tmpa == DayOfWeek
63.1251 \ifnum \tmpa = 0% % if Sunday ...
63.1252     \advance #2 by 1%
63.1253 \else%
63.1254     \ifnum \tmpa = 3% % Wednesday ...
63.1255         \advance #2 by 1%
63.1256     \else%
63.1257         \ifnum \tmpa = 5% % or Friday
63.1258             \advance #2 by 1%
63.1259         \fi%
63.1260     \fi%
63.1261 \fi%
63.1262 \global\@common = #2}%
63.1263 #2 = \@common}

```

\DaysInHebrewYear The macro `\DaysInHebrewYear{year}{days}` calculates the number of days in Hebrew ‘year’.

```

63.1264 \def\DaysInHebrewYear#1#2{%
63.1265     {%
63.1266         \countdef\tmpe = 12% % \tmpe==\count12
63.1267 %
63.1268         \HebrewElapsedDays{#1}{\tmpe}%
63.1269         \advance #1 by 1%
63.1270         \HebrewElapsedDays{#1}{#2}%
63.1271         \advance #2 by -\tmpe%
63.1272         \global\@common = #2}%
63.1273     #2 = \@common}

```

\HebrewDaysInPriorMonths The macro `\HebrewDaysInPriorMonths{month}{year}{days}` calculates the number of days in months prior to ‘month’ in the ‘year’.

```

63.1274 \def\HebrewDaysInPriorMonths#1#2#3{%
63.1275     {%
63.1276         \countdef\tmpf= 14% % \tmpf==\count14
63.1277 %
63.1278         #3 = \ifcase #1% % Days in prior month of regular year
63.1279             0 \or% % no month number 0
63.1280             0 \or% % Tishri
63.1281             30 \or% % Heshvan
63.1282             59 \or% % Kislev
63.1283             89 \or% % Tebeth
63.1284             118 \or% % Shebat

```

```

63.1285          148 \or%          % Adar I
63.1286          148 \or%          % Adar II
63.1287          177 \or%          % Nisan
63.1288          207 \or%          % Iyar
63.1289          236 \or%          % Sivan
63.1290          266 \or%          % Tammuz
63.1291          295 \or%          % Av
63.1292          325 \or%          % Elul
63.1293          400%              % Dummy
63.1294          \fi%
63.1295          \CheckLeapHebrewYear{#2}%
63.1296          \if@HebrewLeap%    % in leap year
63.1297              \ifnum #1 > 6%    % if month after Adar I
63.1298                  \advance #3 by 30% % add 30 days
63.1299          \fi%
63.1300          \fi%
63.1301          \DaysInHebrewYear{#2}{\tmpf}%
63.1302          \ifnum #1 > 3%
63.1303              \ifnum \tmpf = 353%    %
63.1304                  \advance #3 by -1% %
63.1305              \fi%                  % Short Kislev
63.1306              \ifnum \tmpf = 383%    %
63.1307                  \advance #3 by -1% %
63.1308              \fi%                  %
63.1309          \fi%
63.1310 %
63.1311          \ifnum #1 > 2%
63.1312              \ifnum \tmpf = 355%    %
63.1313                  \advance #3 by 1% %
63.1314              \fi%                  % Long Heshvan
63.1315              \ifnum \tmpf = 385%    %
63.1316                  \advance #3 by 1% %
63.1317              \fi%                  %
63.1318          \fi%
63.1319          \global\@common = #3}%
63.1320          #3 = \@common}

```

\AbsoluteFromHebrew The macro `\AbsoluteFromHebrew{day}{month}{year}{absdate}` calculates the absolute date of Hebrew date day.month.year.

```

63.1321 \def\AbsoluteFromHebrew#1#2#3#4{%
63.1322     {%
63.1323         #4 = #1%
63.1324         \HebrewDaysInPriorMonths{#2}{#3}{#1}%
63.1325         \advance #4 by #1%          % Add days in prior months this year
63.1326         \HebrewElapsedDays{#3}{#1}%
63.1327         \advance #4 by #1%          % Add days in prior years
63.1328         \advance #4 by -1373429%    % Subtract days before Gregorian
63.1329         \global\@common = #4}%      % 01.01.0001
63.1330     #4 = \@common}

```

\HebrewFromGregorian The macro `\HebrewFromGregorian{Gday}{Gmonth}{Gyear}{Hday}{Hmonth}{Hyear}` evaluates Hebrew date Hday, Hmonth, Hyear from Gregorian date Gday, Gmonth, Gyear.

```

63.1331 \def\HebrewFromGregorian#1#2#3#4#5#6{%
63.1332     {%
63.1333         \countdef\tmpx= 17%          % \tmpx==\count17
63.1334         \countdef\tmpy= 18%          % \tmpy==\count18
63.1335         \countdef\tmpz= 19%          % \tmpz==\count19
63.1336 %
63.1337         #6 = #3%                      %
63.1338         \global\advance #6 by 3761%    % approximation from above
63.1339         \AbsoluteFromGregorian{#1}{#2}{#3}{#4}%

```

```

63.1340      \tmpz = 1 \tmpy = 1%
63.1341      \AbsoluteFromHebrew{\tmpz}{\tmpy}{#6}{\tmpx}%
63.1342      \ifnum \tmpx > #4%           %
63.1343          \global\advance #6 by -1% Hyear = Gyear + 3760
63.1344      \AbsoluteFromHebrew{\tmpz}{\tmpy}{#6}{\tmpx}%
63.1345      \fi%                         %
63.1346      \advance #4 by -\tmpx%       % Days in this year
63.1347      \advance #4 by 1%           %
63.1348      #5 = #4%                   %
63.1349      \divide #5 by 30%          % Approximation for month from below
63.1350      \loop%                     % Search for month
63.1351          \HebrewDaysInPriorMonths{#5}{#6}{\tmpx}%
63.1352          \ifnum \tmpx < #4%
63.1353              \advance #5 by 1%
63.1354              \tmpy = \tmpx%
63.1355      \repeat%
63.1356      \global\advance #5 by -1%
63.1357      \global\advance #4 by -\tmpy}}
63.1358 </calendar>

```

64 Hebrew input encodings

Hebrew input encodings defined in file `hebinp.dtx`⁷⁷ should be used with `inputenc` L^AT_EX 2_ε package. This package allows the user to specify an input encoding from this file (for example, ISO Hebrew/Latin 8859-8, IBM Hebrew codepage 862 or MS Windows Hebrew codepage 1255) by saying:

```
\usepackage[encoding name]{inputenc}
```

The encoding can also be selected in the document with:

```
\inputencoding{encoding name}
```

The only practical use of this command within a document is when using text from several documents to build up a composite work such as a volume of journal articles. Therefore this command will be used only in vertical mode.

The encodings provided by this package are:

- **si960** 7-bit Hebrew encoding for the range 32–127. This encoding also known as “old-code” and defined by Israeli Standard SI-960.
- **8859-8** ISO 8859-8 Hebrew/Latin encoding commonly used in UNIX systems. This encoding also known as “new-code” and includes hebrew letters in positions starting from 224.
- **cp862** IBM 862 code page commonly used by DOS on IBM-compatible personal computers. This encoding also known as “pc-code” and includes hebrew letters in positions starting from 128.
- **cp1255** MS Windows 1255 (hebrew) code page which is similar to 8859-8. In addition to hebrew letters, this encoding contains also hebrew vowels and dots (nikud).

Each encoding has an associated `.def` file, for example `8859-8.def` which defines the behaviour of each input character, using the commands:

```

\DeclareInputText{slot}{text}
\DeclareInputMath{slot}{math}

```

⁷⁷The files described in this section have version number v1.1b and were last revised on 2004/02/20.

This defines the input character *slot* to be the *text* material or *math* material respectively. For example, 8859-8.def defines slots "EA (letter hebraef) and "B5 (μ) by saying:

```
\DeclareInputText{224}{\hebraef}
\DeclareInputMath{181}{\mu}
```

Note that the *commands* should be robust, and should not be dependent on the output encoding. The same *slot* should not have both a text and a math declaration for it. (This restriction may be removed in future releases of inputenc).

The .def file may also define commands using the declarations:

`\providecommand` or `\ProvideTextCommandDefault`. For example, 8859-8.def defines:

```
\ProvideTextCommandDefault{\textonequarter}{\ensuremath{\frac{1}{4}}}
\DeclareInputText{188}{\textonequarter}
```

The use of the ‘provide’ forms here will ensure that a better definition will not be over-written; their use is recommended since, in general, the best definition depends on the fonts available.

See the documentation in `inputenc.dtx` for details of how to declare input definitions for various encodings.

64.1 Default definitions for characters

First, we insert a `\makeatletter` at the beginning of all .def files to use @ symbol in the macros’ names.

64.1 <-driver>\makeatletter

Some input characters map to internal functions which are not in either the T1 or OT1 font encoding. For this reason default definitions are provided in the encoding file: these will be used unless some other output encoding is used which supports those glyphs. In some cases this default definition has to be simply an error message.

Note that this works reasonably well only because the encoding files for both OT1 and T1 are loaded in the standard LaTeX format.

```
64.2 <*8859-8 | cp862 | cp1255>
64.3 \ProvideTextCommandDefault{\textdegree}{\ensuremath{\sim\circ}}
64.4 \ProvideTextCommandDefault{\textonehalf}{\ensuremath{\frac{1}{2}}}
64.5 \ProvideTextCommandDefault{\textonequarter}{\ensuremath{\frac{1}{4}}}
64.6 </8859-8 | cp862 | cp1255>
64.7 <*8859-8 | cp1255>
64.8 \ProvideTextCommandDefault{\textthreequarters}{\ensuremath{\frac{3}{4}}}
64.9 </8859-8 | cp1255>
64.10 <*cp862 | cp1255>
64.11 \ProvideTextCommandDefault{\textflorin}{\textit{f}}
64.12 </cp862 | cp1255>
64.13 <*cp862>
64.14 \ProvideTextCommandDefault{\textpeseta}{Pt}
64.15 </cp862>
```

The name `\textblacksquare` is derived from the AMS symbol name since Adobe seem not to want this symbol. The default definition, as a rule, makes no claim to being a good design.

```
64.16 <*cp862>
64.17 \ProvideTextCommandDefault{\textblacksquare}
64.18 {\vrule \@width .3em \@height .4em \@depth -.1em\relax}
64.19 </cp862>
```

Some commands can’t be faked, so we have them generate an error message.

```
64.20 <*8859-8 | cp862 | cp1255>
64.21 \ProvideTextCommandDefault{\textcent}
```

```

64.22 {\TextSymbolUnavailable\textcent}
64.23 \ProvideTextCommandDefault{\textyen}
64.24 {\TextSymbolUnavailable\textyen}
64.25 </8859-8 | cp862 | cp1255>
64.26 (*8859-8)
64.27 \ProvideTextCommandDefault{\textcurrency}
64.28 {\TextSymbolUnavailable\textcurrency}
64.29 </8859-8>
64.30 (*cp1255)
64.31 \ProvideTextCommandDefault{\newsheqel}
64.32 {\TextSymbolUnavailable\newsheqel}
64.33 </cp1255>
64.34 (*8859-8 | cp1255)
64.35 \ProvideTextCommandDefault{\textbrokenbar}
64.36 {\TextSymbolUnavailable\textbrokenbar}
64.37 </8859-8 | cp1255>
64.38 (*cp1255)
64.39 \ProvideTextCommandDefault{\textperthousand}
64.40 {\TextSymbolUnavailable\textperthousand}
64.41 </cp1255>

```

Characters that are supposed to be used only in math will be defined by `\providecommand` because L^AT_EX 2_ε assumes that the font encoding for math fonts is static.

```

64.42 (*8859-8 | cp1255)
64.43 \providecommand{\mathonesuperior}{\textsup}
64.44 \providecommand{\maththreesuperior}{\textsup}
64.45 </8859-8 | cp1255>
64.46 (*8859-8 | cp862 | cp1255)
64.47 \providecommand{\mathtwosuperior}{\textsup}
64.48 </8859-8 | cp862 | cp1255>
64.49 (*cp862)
64.50 \providecommand{\mathordmasculine}{\textordmasculine}
64.51 \providecommand{\mathordfeminine}{\textordfeminine}
64.52 </cp862>

```

64.2 The SI-960 encoding

The SI-960 or “old-code” encoding only allows characters in the range 32–127, so we only need to provide an empty `si960.def` file.

64.3 The ISO 8859-8 encoding and the MS Windows cp1255 encoding

The `8859-8.def` encoding file defines the characters in the ISO 8859-8 encoding.

The MS Windows Hebrew character set incorporates the Hebrew letter repertoire of ISO 8859-8, and uses the same code points (starting from 224). It has also some important additions in the 128–159 and 190–224 ranges.

```

64.53 (*cp1255)
64.54 \DeclareInputText{130}{\quotesinglbase}
64.55 \DeclareInputText{131}{\textflorin}
64.56 \DeclareInputText{132}{\quotedblbase}
64.57 \DeclareInputText{133}{\dots}
64.58 \DeclareInputText{134}{\dag}
64.59 \DeclareInputText{135}{\ddag}
64.60 \DeclareInputText{136}{\textasciitilde}
64.61 \DeclareInputText{137}{\textperthousand}
64.62 \DeclareInputText{139}{\guilsinglleft}
64.63 \DeclareInputText{145}{\textquoteleft}
64.64 \DeclareInputText{146}{\textquoteright}
64.65 \DeclareInputText{147}{\textquotedblleft}

```

```

64.66 \DeclareInputText{148}{\textquotedblright}
64.67 \DeclareInputText{149}{\textbullet}
64.68 \DeclareInputText{150}{\textendash}
64.69 \DeclareInputText{151}{\textemdash}
64.70 \DeclareInputText{152}{\~{}}
64.71 \DeclareInputText{153}{\texttrademark}
64.72 \DeclareInputText{155}{\guilsinglright}
64.73 </cp1255>
64.74 (*8859-8 | cp1255)
64.75 \DeclareInputText{160}{\nobreakspace}
64.76 \DeclareInputText{162}{\textcent}
64.77 \DeclareInputText{163}{\pounds}
64.78 <+8859-8>\DeclareInputText{164}{\textcurrency}
64.79 <+cp1255>\DeclareInputText{164}{\newsheqel}
64.80 \DeclareInputText{165}{\textyen}
64.81 \DeclareInputText{166}{\textbrokenbar}
64.82 \DeclareInputText{167}{\S}
64.83 \DeclareInputText{168}{\"{}}
64.84 \DeclareInputText{169}{\textcopyright}
64.85 <+8859-8>\DeclareInputMath{170}{\times}
64.86 \DeclareInputText{171}{\guillemotleft}
64.87 \DeclareInputMath{172}{\lnot}
64.88 \DeclareInputText{173}{\~{}}
64.89 \DeclareInputText{174}{\textregistered}
64.90 \DeclareInputText{175}{\@tabacckludge={}}
64.91 \DeclareInputText{176}{\textdegree}
64.92 \DeclareInputMath{177}{\pm}
64.93 \DeclareInputMath{178}{\mathtwosuperior}
64.94 \DeclareInputMath{179}{\maththreesuperior}
64.95 \DeclareInputText{180}{\@tabacckludge'{} }
64.96 \DeclareInputMath{181}{\mu}
64.97 \DeclareInputText{182}{\P}
64.98 \DeclareInputText{183}{\textperiodcentered}
64.99 <+8859-8>\DeclareInputText{184}{\c\ }
64.100 \DeclareInputMath{185}{\mathonesuperior}
64.101 <+8859-8>\DeclareInputMath{186}{\div}
64.102 \DeclareInputText{187}{\guillemotright}
64.103 \DeclareInputText{188}{\textonequarter}
64.104 \DeclareInputText{189}{\textonehalf}
64.105 \DeclareInputText{190}{\textthreequarters}
64.106 </8859-8 | cp1255>

```

Hebrew vowels and dots (nikud) are included only to MS Windows cp1255 page and start from the position 192.

```

64.107 (*cp1255)
64.108 \DeclareInputText{192}{\hebsheva}
64.109 \DeclareInputText{193}{\hebatafsegol}
64.110 \DeclareInputText{194}{\hebatafpatah}
64.111 \DeclareInputText{195}{\hebatafqamats}
64.112 \DeclareInputText{196}{\hebhiriq}
64.113 \DeclareInputText{197}{\hebtseret}
64.114 \DeclareInputText{198}{\hebsegol}
64.115 \DeclareInputText{199}{\hebpatah}
64.116 \DeclareInputText{200}{\he bqamats}
64.117 \DeclareInputText{201}{\hebholam}
64.118 \DeclareInputText{203}{\hebqubuts}
64.119 \DeclareInputText{204}{\hebdagesh}
64.120 \DeclareInputText{205}{\hebmeteg}
64.121 \DeclareInputText{206}{\hebmaqaf}
64.122 \DeclareInputText{207}{\hebrafe}
64.123 \DeclareInputText{208}{\hebpaseq}
64.124 \DeclareInputText{209}{\hebshindot}
64.125 \DeclareInputText{210}{\hebsindot}

```

```

64.126 \DeclareInputText{211}{\hebsofpasuq}
64.127 \DeclareInputText{212}{\hebdoublevav}
64.128 \DeclareInputText{213}{\hebvavyod}
64.129 \DeclareInputText{214}{\hebdoubleyod}
64.130 </cp1255>

```

Hebrew letters start from the position 224 in both encodings.

```

64.131 (*8859-8 | cp1255)
64.132 \DeclareInputText{224}{\hebalef}
64.133 \DeclareInputText{225}{\hebbet}
64.134 \DeclareInputText{226}{\hebgimel}
64.135 \DeclareInputText{227}{\hebdalet}
64.136 \DeclareInputText{228}{\hebhe}
64.137 \DeclareInputText{229}{\hebvav}
64.138 \DeclareInputText{230}{\hebzayin}
64.139 \DeclareInputText{231}{\hebbet}
64.140 \DeclareInputText{232}{\hebtet}
64.141 \DeclareInputText{233}{\hebyod}
64.142 \DeclareInputText{234}{\hebfinalkaf}
64.143 \DeclareInputText{235}{\hebkaf}
64.144 \DeclareInputText{236}{\heblamed}
64.145 \DeclareInputText{237}{\hebfinalmem}
64.146 \DeclareInputText{238}{\hebmeme}
64.147 \DeclareInputText{239}{\hebfinalnun}
64.148 \DeclareInputText{240}{\hebnun}
64.149 \DeclareInputText{241}{\hebsamekh}
64.150 \DeclareInputText{242}{\hebayin}
64.151 \DeclareInputText{243}{\hebfinalpe}
64.152 \DeclareInputText{244}{\hebpe}
64.153 \DeclareInputText{245}{\hebfinaltsadi}
64.154 \DeclareInputText{246}{\hebtsadi}
64.155 \DeclareInputText{247}{\hebqof}
64.156 \DeclareInputText{248}{\hebresh}
64.157 \DeclareInputText{249}{\hebshin}
64.158 \DeclareInputText{250}{\hebtav}
64.159 </8859-8 | cp1255>

```

Special symbols which define the direction of symbols explicitly. Currently, they are not used in L^AT_EX.

```

64.160 (*cp1255)
64.161 \DeclareInputText{253}{\lefttorightmark}
64.162 \DeclareInputText{254}{\righttoleftmark}
64.163 </cp1255>

```

64.4 The IBM code page 862

The cp862.def encoding file defines the characters in the IBM codepage 862 encoding. The DOS graphics ‘letters’ and a few other positions are ignored (left undefined).

Hebrew letters start from the position 128.

```

64.164 (*cp862)
64.165 \DeclareInputText{128}{\hebalef}
64.166 \DeclareInputText{129}{\hebbet}
64.167 \DeclareInputText{130}{\hebgimel}
64.168 \DeclareInputText{131}{\hebdalet}
64.169 \DeclareInputText{132}{\hebhe}
64.170 \DeclareInputText{133}{\hebvav}
64.171 \DeclareInputText{134}{\hebzayin}
64.172 \DeclareInputText{135}{\hebbet}
64.173 \DeclareInputText{136}{\hebtet}
64.174 \DeclareInputText{137}{\hebyod}
64.175 \DeclareInputText{138}{\hebfinalkaf}
64.176 \DeclareInputText{139}{\hebkaf}

```

64.177 \DeclareInputText{140}{\heblamed}
 64.178 \DeclareInputText{141}{\hebfinalmem}
 64.179 \DeclareInputText{142}{\hebmem}
 64.180 \DeclareInputText{143}{\hebfinalnun}
 64.181 \DeclareInputText{144}{\hebnnun}
 64.182 \DeclareInputText{145}{\hebsamekh}
 64.183 \DeclareInputText{146}{\hebayin}
 64.184 \DeclareInputText{147}{\hebfinalpe}
 64.185 \DeclareInputText{148}{\hebpe}
 64.186 \DeclareInputText{149}{\hebfinaltsadi}
 64.187 \DeclareInputText{150}{\hebtsadi}
 64.188 \DeclareInputText{151}{\hebqof}
 64.189 \DeclareInputText{152}{\hebresh}
 64.190 \DeclareInputText{153}{\hebshin}
 64.191 \DeclareInputText{154}{\hebtav}

 64.192 \DeclareInputText{155}{\textcent}
 64.193 \DeclareInputText{156}{\pounds}
 64.194 \DeclareInputText{157}{\textyen}
 64.195 \DeclareInputText{158}{\textpeseta}
 64.196 \DeclareInputText{159}{\textflorin}
 64.197 \DeclareInputText{160}{\@tabacckludge'a}
 64.198 \DeclareInputText{161}{\@tabacckludge'i}
 64.199 \DeclareInputText{162}{\@tabacckludge'o}
 64.200 \DeclareInputText{163}{\@tabacckludge'u}
 64.201 \DeclareInputText{164}{\~n}
 64.202 \DeclareInputText{165}{\~N}
 64.203 \DeclareInputMath{166}{\mathordfeminine}
 64.204 \DeclareInputMath{167}{\mathordmasculine}
 64.205 \DeclareInputText{168}{\textquestiondown}
 64.206 \DeclareInputMath{170}{\lnot}
 64.207 \DeclareInputText{171}{\textonehalf}
 64.208 \DeclareInputText{172}{\textonequarter}
 64.209 \DeclareInputText{173}{\texttexclamdown}
 64.210 \DeclareInputText{174}{\guillemotleft}
 64.211 \DeclareInputText{175}{\guillemotright}

 64.212 \DeclareInputMath{224}{\alpha}
 64.213 \DeclareInputText{225}{\ss}
 64.214 \DeclareInputMath{226}{\Gamma}
 64.215 \DeclareInputMath{227}{\pi}
 64.216 \DeclareInputMath{228}{\Sigma}
 64.217 \DeclareInputMath{229}{\sigma}
 64.218 \DeclareInputMath{230}{\mu}
 64.219 \DeclareInputMath{231}{\tau}
 64.220 \DeclareInputMath{232}{\Phi}
 64.221 \DeclareInputMath{233}{\Theta}
 64.222 \DeclareInputMath{234}{\Omega}
 64.223 \DeclareInputMath{235}{\delta}
 64.224 \DeclareInputMath{236}{\infty}
 64.225 \DeclareInputMath{237}{\phi}
 64.226 \DeclareInputMath{238}{\varepsilon}
 64.227 \DeclareInputMath{239}{\cap}
 64.228 \DeclareInputMath{240}{\equiv}
 64.229 \DeclareInputMath{241}{\pm}
 64.230 \DeclareInputMath{242}{\geq}
 64.231 \DeclareInputMath{243}{\leq}
 64.232 \DeclareInputMath{246}{\div}
 64.233 \DeclareInputMath{247}{\approx}
 64.234 \DeclareInputText{248}{\textdegree}
 64.235 \DeclareInputText{249}{\textperiodcentered}
 64.236 \DeclareInputText{250}{\textbullet}
 64.237 \DeclareInputMath{251}{\surd}
 64.238 \DeclareInputMath{252}{\mathnssuperior}


```

64.239 \DeclareInputMath{253}{\mathtwosuperior}
64.240 \DeclareInputText{254}{\textblacksquare}
64.241 \DeclareInputText{255}{\nobreakspace}
64.242 \end{cp862}

```

`\DisableNikud` A utility macro to ignore any nikud character that may appear in the input. This allows you to ignore cp1255 nikud characters that happened to appear in the input.

```

64.243 (*8859-8)
64.244 \newcommand{\DisableNikud}{%
64.245   \DeclareInputText{192}{}%
64.246   \DeclareInputText{193}{}%
64.247   \DeclareInputText{194}{}%
64.248   \DeclareInputText{195}{}%
64.249   \DeclareInputText{196}{}%
64.250   \DeclareInputText{197}{}%
64.251   \DeclareInputText{198}{}%
64.252   \DeclareInputText{199}{}%
64.253   \DeclareInputText{200}{}%
64.254   \DeclareInputText{201}{}%
64.255   \DeclareInputText{203}{}%
64.256   \DeclareInputText{204}{}%
64.257   \DeclareInputText{205}{}%
64.258   \DeclareInputText{206}{}%
64.259   \DeclareInputText{207}{}%
64.260   \DeclareInputText{208}{}%
64.261   \DeclareInputText{209}{}%
64.262   \DeclareInputText{210}{}%
64.263   \DeclareInputText{211}{}%
64.264   \DeclareInputText{212}{}%
64.265   \DeclareInputText{213}{}%
64.266   \DeclareInputText{214}{}%
64.267 }
64.268 \end{8859-8}

```

Finally, we reset the category code of the `@` sign at the end of all `.def` files.

```

64.269 \end{driver}\makeatother

```

65 Hebrew font encodings

Don't forget to update the docs...

65.1 THIS SECTION IS OUT OF DATE. UPDATE DOCS TO MATCH HE8 ENCODING

The file `hebrew.fdd`⁷⁸ contains the Local Hebrew Encoding (LHE) definition, the external font information needed to use the Hebrew 7-bit fonts (old code fonts) and `hebfont` package that provides Hebrew font switching commands.

Using this file as an input, `lheenc.def` encoding definition file, all `.fd` files (font definition files) and font switching package for available Hebrew fonts are generated. We chose to use 7-bit encoding as default font encoding, because:

1. There are many 7-bit encoded Hebrew fonts available, more than for any other encoding.
2. Available T_EX Hebrew fonts do not include latin alphabet, and we can safely map Hebrew glyphs to the ASCII positions (0 – 127).

⁷⁸The files described in this section have version number v1.2c and were last revised on 2005/05/20.

Current definition of the LHE encoding supports only Hebrew letters (`\hebalef–\hebtav`), but not Hebrew points, such as `\hebdagesh`, `\hebqamats`, `\hebpatah`, `\hebshindot`, etc. We are working now on such addition.

65.2 The DOCSTRIP modules

The following modules are used in the implementation to direct DOCSTRIP in generating external files:

driver	produce a documentation driver file
HE8enc	produce the encoding definition for CodePage 1255 (HE8)
HE8cmr	make Hebrew default font in HE8
HE8cmss	make Hebrew sans-serif font in HE8
HE8cmtt	make Hebrew typewriter font in HE8
HE8OmegaHebrew	Hebrew font from the Omega project (by ???)
HE8aharoni	Hebrew sans-serif font (Culmus)
HE8david	Hebrew serif font (Culmus)
HE8drugulin	Hebrew old serif font (Culmus)
HE8ellinia	Hebrew isans-serif font (Culmus)
HE8frankruehl	Hebrew serif font (Culmus)
HE8KtavYad	Hebrew handwriting font (Culmus)
HE8MiriamMono	Hebrew monospaced font
HE8Nachlieli	Hebrew sans-serif font (Culmus)
HE8CourierShalom	Hebrew Shalom (Courier) font (by IBM)
HE8HelveticaNarkissTam	Hebrew NarkisTam (Helvetica) (by Zvi Narkis)
HE8TimesNarkissim	Hebrew Narkissim (Times) (by Zvi Narkis)
HE8mfdavid	Hebrew David font (by ???)
HE8mffrank	Hebrew Frank-Ruehl font (by ??)
HE8mffrankthick	Hebrew Frank-Ruehl (thick) font (by ??)
HE8mffrankthin	Hebrew Frank-Ruehl (thin) font (by ??)
HE8mfmiriam	Hebrew Miriam font (by ???)
HE8mfmiriamwide	Hebrew Miriam (wide) font (by ???)
HE8mfarnkistam	Hebrew Narkis Tam font (by ???)
LHEenc	produce the encoding definition for Local Hebrew Encoding (LHE)
LHEcmr	make Hebrew default font in LHE
LHEcmss	make Hebrew sans-serif font in LHE
LHEcmtt	make Hebrew typewriter font in LHE
LHEclas	make Hebrew classic font (by Joel M. Hoffman) in LHE
LHEshold	make Hebrew shalom old font (by Jonathan Brecher) in LHE
LHEshscr	make Hebrew shalom script font (by Jonathan Brecher) in LHE
LHEshstk	make Hebrew shalom stick font (by Jonathan Brecher) in LHE
LHEfr	make Hebrew frank-ruehl font in LHE
LHEcrml	make Hebrew carmel font (by Dr. Samy Zafrany) in LHE
LHERedis	make Hebrew redis font (by Prof. Jacques J. Goldberg) in LHE
nowarn	option for font definition files, that used to produce “silent” font substitutions without giving warnings
hebfont	create Hebrew font switching commands package

A typical DOCSTRIP command file would then have entries like:

```
\generateFile[lhecmr.fd]{t}{\from{hebrew.fdd}{LHEcmr,nowarn}}
```

65.3 The LHEencoding definition file

The Hebrew font encoding LHE is based upon the old-code encoding also known as the Israeli Standard SI-960. Many Hebrew T_EX fonts from the Hebrew University

of Jerusalem are encoded in this encoding. It only uses the lower 128 positions of the font table. As local encoding its name start with the letter ‘L’.

First we define the Local Hebrew Encoding; specify a default for the font substitution process for the LHE encoding and supply a font to be used when all else fails.

```
65.1 (*LHEenc)
65.2 \DeclareFontEncoding{LHE}{-}{-}
65.3 \DeclareFontSubstitution{LHE}{cmr}{m}{n}
65.4 \DeclareErrorFont{LHE}{cmr}{m}{n}{10}
65.5 \end{LHEenc}
```

Then we define a few commands in the LHE encoding.

```
65.6 (*LHEenc)
65.7 \ProvideTextCommand{\textcopyright}{LHE}{\textcircled{\@latin{c}}}
65.8 \ProvideTextCommand{\textregistered}{LHE}{\textcircled{\scshape%
65.9 \@latin{r}}}
65.10 \ProvideTextCommand{\texttrademark}{LHE}{\textsuperscript{\@latin{TM}}}
65.11 \end{LHEenc}
```

Because not everyone can input Hebrew input text directly from the keyboard we need to define control sequences for all the Hebrew glyphs in the fonts. In addition, we want to support many input encodings for Hebrew and to keep the language definition file (`hebrew.ldf`) independent of the encoding. Therefore, we exploit the standard $\text{\LaTeX 2}_{\epsilon}$ font encoding mechanism to define control sequences for all the Hebrew glyphs in the fonts in encoding-specific way. The language definition file uses only the control sequences and doesn’t need to check the current font or input encoding.

In the LHE encoding (7-bit encoding) all the Hebrew glyphs reside in the *lower* half of the font. Currently, only the Hebrew letters are supported. They use the same positions as the latin small letters in ASCII encoding and the position of ‘.

The symbol ‘ (glyph 96) is used by Hebrew letter *Alef*, so we need to define its `\lccode` to allow hyphenation. All other letters retain the same `\lccodes` as their latin counterparts.

```
65.12 \end{LHEenc}\lccode‘=‘‘
```

Hebrew letters occupy the positions 96–122 in LHE encoding:

```
65.13 (*LHEenc)
65.14 \DeclareTextSymbol{\hebalef}{LHE}{96}
65.15 \DeclareTextSymbol{\hebbet}{LHE}{97}
65.16 \DeclareTextSymbol{\hebgimel}{LHE}{98}
65.17 \DeclareTextSymbol{\hebdalet}{LHE}{99}
65.18 \DeclareTextSymbol{\hebhe}{LHE}{100}
65.19 \DeclareTextSymbol{\hebvav}{LHE}{101}
65.20 \DeclareTextSymbol{\hebzayin}{LHE}{102}
65.21 \DeclareTextSymbol{\hebhet}{LHE}{103}
65.22 \DeclareTextSymbol{\hebtet}{LHE}{104}
65.23 \DeclareTextSymbol{\hebyod}{LHE}{105}
65.24 \DeclareTextSymbol{\hebfinalkaf}{LHE}{106}
65.25 \DeclareTextSymbol{\hebkafe}{LHE}{107}
65.26 \DeclareTextSymbol{\heblamed}{LHE}{108}
65.27 \DeclareTextSymbol{\hebfinalmem}{LHE}{109}
65.28 \DeclareTextSymbol{\hebmem}{LHE}{110}
65.29 \DeclareTextSymbol{\hebfinalnun}{LHE}{111}
65.30 \DeclareTextSymbol{\hebnun}{LHE}{112}
65.31 \DeclareTextSymbol{\hebsamekh}{LHE}{113}
65.32 \DeclareTextSymbol{\hebayin}{LHE}{114}
65.33 \DeclareTextSymbol{\hebfinalpe}{LHE}{115}
65.34 \DeclareTextSymbol{\hebpe}{LHE}{116}
65.35 \DeclareTextSymbol{\hebfinaltsadi}{LHE}{117}
65.36 \DeclareTextSymbol{\hebtsadi}{LHE}{118}
65.37 \DeclareTextSymbol{\hebqof}{LHE}{119}
65.38 \DeclareTextSymbol{\hebreish}{LHE}{120}
```

```

65.39 \DeclareTextSymbol{\hebshin}{LHE}{121}
65.40 \DeclareTextSymbol{\hebtav}{LHE}{122}
65.41 \LHEenc

```

Letter `\hebsin` is defined as a synonym of `\hebshin`:

```

65.42 \LHEenc\let\hebsin=\hebshin

```

65.4 The font definition files (in LHE encoding)

65.4.1 Hebrew default font

It uses *Jerusalem* font for regular font, *Old Jaffa* font for italic shape and small-caps, *Dead Sea* font for bold face, and *Tel-Aviv* for bold-italic

```

65.43 (*LHEcmr)
65.44 \DeclareFontFamily{LHE}{cmr}{\hyphenchar\font45}
65.45 \DeclareFontShape{LHE}{cmr}{m}{n}
65.46     {<-> jerus10 }{}
65.47 %%%%% Italicized shape
65.48 \DeclareFontShape{LHE}{cmr}{m}{it}
65.49     {<-> oldjaf10 }{}
65.50 \DeclareFontShape{LHE}{cmr}{m}{sl}
65.51     {<-> oldjaf10 }{}
65.52 \DeclareFontShape{LHE}{cmr}{m}{sc}
65.53     {<-> oldjaf10 }{}
65.54 %%%%% Bold extended series
65.55 \DeclareFontShape{LHE}{cmr}{bx}{n}
65.56     {<-> deads10 }{}
65.57 \DeclareFontShape{LHE}{cmr}{b}{n}
65.58     {<-> deads10 }{}
65.59 %%%%% Bold extended (Italic) series
65.60 \DeclareFontShape{LHE}{cmr}{bx}{sl}
65.61     {<-> telav10 }{}
65.62 \DeclareFontShape{LHE}{cmr}{bx}{it}
65.63     {<-> telav10 }{}
65.64 \LHEcmr

```

65.4.2 Hebrew sans-serif font

We use *Tel Aviv* font for the Sans family. *Old Jaffa* font is used for italic shape and *Dead Sea* used for bold face.

```

65.65 (*LHEcmss)
65.66 \DeclareFontFamily{LHE}{cmss}{\hyphenchar\font45}
65.67 \DeclareFontShape{LHE}{cmss}{m}{n}
65.68     {<-> telav10 }{}
65.69 %%%%% Font/shape undefined, therefore substituted
65.70 \DeclareFontShape{LHE}{cmss}{m}{sc}
65.71 (-nowarn) {<->sub * cmss/m/n}{}
65.72 (+nowarn) {<->ssub * cmss/m/n}{}
65.73 %%%%% Italicized shape
65.74 \DeclareFontShape{LHE}{cmss}{m}{it}
65.75     {<-> oldjaf10 }{}
65.76 %%%%% Font/shape undefined, therefore substituted
65.77 \DeclareFontShape{LHE}{cmss}{m}{sl}
65.78 (-nowarn) {<->sub * cmss/m/it}{}
65.79 (+nowarn) {<->ssub * cmss/m/it}{}
65.80 %%%%% Bold extended series
65.81 \DeclareFontShape{LHE}{cmss}{bx}{n}
65.82     {<-> deads10 }{}
65.83 %%%%% Font/shape undefined, therefore substituted
65.84 \DeclareFontShape{LHE}{cmss}{b}{n}
65.85 (-nowarn) {<->sub * cmss/bx/n}{}
65.86 (+nowarn) {<->ssub * cmss/bx/n}{}

```

```

65.87 %%%%%%%%% Font/shape undefined, therefore substituted
65.88 \DeclareFontShape{LHE}{cmss}{bx}{sl}
65.89 <-nowarn> {<->sub * cmss/bx/n}{ }
65.90 <+nowarn> {<->ssub * cmss/bx/n}{ }
65.91 %%%%%%%%% Font/shape undefined, therefore substituted
65.92 \DeclareFontShape{LHE}{cmss}{bx}{it}
65.93 <-nowarn> {<->sub * cmss/bx/n}{ }
65.94 <+nowarn> {<->ssub * cmss/bx/n}{ }
65.95 </LHEcmss>

```

65.4.3 Hebrew typewriter font

We use *Tel Aviv* font as the typewriter font. *Old Jaffa* font is used for italic shape and *Dead Sea* used for bold face.

```

65.96 (*LHEcmtt)
65.97 \DeclareFontFamily{LHE}{cmtt}{\hyphenchar \font\m@ne}
65.98 \DeclareFontShape{LHE}{cmtt}{m}{n}
65.99 {<-> telav10 }{ }
65.100 %%%%%%%%% Font/shape undefined, therefore substituted
65.101 \DeclareFontShape{LHE}{cmtt}{m}{sc}
65.102 <-nowarn> {<->sub * cmtt/m/n}{ }
65.103 <+nowarn> {<->ssub * cmtt/m/n}{ }
65.104 %%%%%%%%% Italicized shape
65.105 \DeclareFontShape{LHE}{cmtt}{m}{it}
65.106 {<-> oldjaf10 }{ }
65.107 %%%%%%%%% Font/shape undefined, therefore substituted
65.108 \DeclareFontShape{LHE}{cmtt}{m}{sl}
65.109 <-nowarn> {<->sub * cmtt/m/it}{ }
65.110 <+nowarn> {<->ssub * cmtt/m/it}{ }
65.111 %%%%%%%%% Bold extended series
65.112 \DeclareFontShape{LHE}{cmtt}{bx}{n}
65.113 {<-> deads10 }{ }
65.114 %%%%%%%%% Font/shape undefined, therefore substituted
65.115 \DeclareFontShape{LHE}{cmtt}{bx}{it}
65.116 <-nowarn> {<->sub * cmtt/bx/n}{ }
65.117 <+nowarn> {<->ssub * cmtt/bx/n}{ }
65.118 </LHEcmtt>

```

65.4.4 Hebrew classic font

Hclassic and *hcaption* fonts are distributed freely from CTAN sites and copyrighted by Joel M. Hoffman, of 19 Hillcrest Lane, Rye, NY 10580 USA, e-mail: 72700.402@compuserve.com.

Hclassic is a modernized Classical Hebrew font (in the same way that Knuth's *cmr* family is a modernized Roman font — but his fonts are much nicer). Hcaption is a slanted version of hclassic font. Both fonts contain all of the Hebrew consonants, the (rarely used) ligature *alef-lamed* and two versions of the letter *ayin* for use with and without vowels. Hclassic also contains all of the vowels found in Hebrew, a symbol for *meteg*, and dots for use as a *dagesh* and for differentiating *shin* and *sin* letters.

Currently, only the Hebrew consonants (*hebalef* – *hebtav*) from these fonts are supported by L^AT_EX 2_ε, however one can use vowels and dots directly with PLAIN T_EX macros. We are working on generic vowels and dots support for L^AT_EX 2_ε.

```

65.119 (*LHEclas)
65.120 \DeclareFontFamily{LHE}{clas}{ }
65.121 \DeclareFontShape{LHE}{clas}{m}{n}
65.122 {<-> s * [0.83345] hclassic }{ }
65.123 %%%%%%%%% Font/shape undefined, therefore substituted
65.124 \DeclareFontShape{LHE}{clas}{m}{sc}
65.125 <-nowarn> {<->sub * clas/m/n}{ }

```

```

65.126 <+nowarn> {<->ssub * clas/m/n}{f}
65.127 %%%%%%%%% Slanted shape
65.128 \DeclareFontShape{LHE}{clas}{m}{sl}{
65.129     {<-> s * [0.69389] hcaption }{f}
65.130 %%%%%%%%% Font/shape undefined, therefore substituted
65.131 \DeclareFontShape{LHE}{clas}{m}{it}{
65.132 <-nowarn> {<->sub * clas/m/sl}{f}
65.133 <+nowarn> {<->ssub * clas/m/sl}{f}
65.134 </LHEclas>

```

65.4.5 Hebrew shalom fonts

All three shalom fonts (*ShalomScript10*, *ShalomStick10* and *ShalomOldStyle10*) have been created by Jonathan Brecher, of 9 Skyview Road, Lexington, MA 02173-1112 USA, e-mail: brecher@husc.harvard.edu.

All shalom fonts have been written in POSTSCRIPT via Fontographer on a Mac. The fonts have been converted to METAFONT by Rama Porrat (e-mail: rama@cc.huji.ac.il), using the utility *typo*, a font editor + converter between font formats (a commercial product). *ShalomScript10.mf* is the METAFONT equivalent of *ShalomScript.ps*, *ShalomStick10.mf* came from *ShalomStick.ps* and *ShalomOldStyle10.mf* originated in *ShalomOldStyle.ps*.

The fonts differ in the letters' style. *ShalomScript10* contains hand writing Hebrew letters; *ShalomStick10* contains sans-serif letters, and *ShalomOldStyle10* contains old style letters. All three fonts contain vowels and dots (nikud). While converting to METAFONT, letters and symbols within the fonts have been arranged so as to get a usable font for writing Hebrew documents in T_EX or L^AT_EX, with as well as without vowels.

Currently, only the Hebrew consonants (*hebalef* – *hebtav*) from these fonts are supported by L^AT_EX 2_ε, however one can use vowels and dots directly with PLAIN T_EX macros. We are working on generic vowels and dots support for L^AT_EX 2_ε.

```

65.135 <*LHEshold>
65.136 \DeclareFontFamily{LHE}{shold}{f}
65.137 \DeclareFontShape{LHE}{shold}{m}{n}{
65.138     {<-> shold10 }{f}
65.139 </LHEshold>
65.140 <*LHEshscr>
65.141 \DeclareFontFamily{LHE}{shscr}{f}
65.142 \DeclareFontShape{LHE}{shscr}{m}{n}{
65.143     {<-> shscr10 }{f}
65.144 </LHEshscr>
65.145 <*LHEshstk>
65.146 \DeclareFontFamily{LHE}{shstk}{f}
65.147 \DeclareFontShape{LHE}{shstk}{m}{n}{
65.148     {<-> shstk10 }{f}
65.149 </LHEshstk>

```

65.4.6 Hebrew frank-ruehl font

Frank Ruehl font was written in METAFONT and includes three shapes: regular, bold extaneded and slanted.

```

65.150 <*LHEfr>
65.151 \DeclareFontFamily{LHE}{fr}{f}
65.152 \DeclareFontShape{LHE}{fr}{m}{n}{
65.153     {<-> fr }{f}
65.154 %%%%%%%%% Font/shape undefined, therefore substituted
65.155 \DeclareFontShape{LHE}{fr}{m}{sc}{
65.156 <-nowarn> {<->sub * fr/m/n}{f}
65.157 <+nowarn> {<->ssub * fr/m/n}{f}
65.158 %%%%%%%%% Slanted shape

```

```

65.159 \DeclareFontShape{LHE}{fr}{m}{sl}
65.160      {<-> frsl }{}
65.161 %%%%%%%%% Font/shape undefined, therefore substituted
65.162 \DeclareFontShape{LHE}{fr}{m}{it}
65.163 (-nowarn) {<->sub * fr/m/sl}{}
65.164 (+nowarn) {<->ssub * fr/m/sl}{}
65.165 %%%%%%%%% Bold extended series
65.166 \DeclareFontShape{LHE}{fr}{bx}{n}
65.167      {<-> frbx }{}
65.168 %%%%%%%%% Font/shape undefined, therefore substituted
65.169 \DeclareFontShape{LHE}{fr}{b}{n}
65.170 (-nowarn) {<->sub * fr/bx/n}{}
65.171 (+nowarn) {<->ssub * fr/bx/n}{}
65.172 %%%%%%%%% Font/shape undefined, therefore substituted
65.173 \DeclareFontShape{LHE}{fr}{bx}{sl}
65.174 (-nowarn) {<->sub * fr/bx/n}{}
65.175 (+nowarn) {<->ssub * fr/bx/n}{}
65.176 %%%%%%%%% Font/shape undefined, therefore substituted
65.177 \DeclareFontShape{LHE}{fr}{bx}{it}
65.178 (-nowarn) {<->sub * fr/bx/n}{}
65.179 (+nowarn) {<->ssub * fr/bx/n}{}
65.180 </LHEfr>

```

65.4.7 Hebrew carmel font

Carmel font includes regular and slanted shapes. It was created by Dr. Samy Zafrany of the Technion, Haifa, Israel with the intention of making nice fonts for headers and emphasized text.

```

65.181 (*LHEcrlm)
65.182 \DeclareFontFamily{LHE}{crlm}{}
65.183 \DeclareFontShape{LHE}{crlm}{m}{n}
65.184      {<-> crml10 }{}
65.185 %%%%%%%%% Font/shape undefined, therefore substituted
65.186 \DeclareFontShape{LHE}{crlm}{m}{sc}
65.187 (-nowarn) {<->sub * crml/m/n}{}
65.188 (+nowarn) {<->ssub * crml/m/n}{}
65.189 %%%%%%%%% Slanted shape
65.190 \DeclareFontShape{LHE}{crlm}{m}{sl}
65.191      {<-> crmlsl10 }{}
65.192 %%%%%%%%% Font/shape undefined, therefore substituted
65.193 \DeclareFontShape{LHE}{crlm}{m}{it}
65.194 (-nowarn) {<->sub * crml/m/sl}{}
65.195 (+nowarn) {<->ssub * crml/m/sl}{}
65.196 </LHEcrlm>

```

65.4.8 Hebrew redis font

Redis font has been created by Prof. Jacques J. Goldberg of the Technion. Haifa, Israel. The font is available in regular, slanted and bold extended shapes. This font contains a full set of Hebrew letters in a “sans-serif vectorized” style, and selected punctuation.

```

65.197 (*LHEredis)
65.198 \DeclareFontFamily{LHE}{redis}{}
65.199 \DeclareFontShape{LHE}{redis}{m}{n}{%
65.200   <5> <6> redis7
65.201   <7> <8> <9> <10> <12> gen * redis
65.202   <10.95> redis10
65.203   <14.4> redis12
65.204   <17.28> <20.74> <24.88> redis17}{}
65.205 %%%%%%%%% Font/shape undefined, therefore substituted
65.206 \DeclareFontShape{LHE}{redis}{m}{sc}

```

```

65.207 <-nowarn> {<->sub * redis/m/n}{ }
65.208 <+nowarn> {<->ssub * redis/m/n}{ }
65.209 %%%%%%%%% Slanted shape
65.210 \DeclareFontShape{LHE}{redis}{m}{sl}{%
65.211 <5> <6> <7> rediss8
65.212 <8> <9> <10> <12> gen * rediss
65.213 <10.95> rediss10
65.214 <14.4> <17.28> <20.74> <24.88> rediss12}{ }
65.215 %%%%%%%%% Font/shape undefined, therefore substituted
65.216 \DeclareFontShape{LHE}{redis}{m}{it}{
65.217 <-nowarn> {<->sub * redis/m/sl}{ }
65.218 <+nowarn> {<->ssub * redis/m/sl}{ }
65.219 %%%%%%%%% Bold extended series
65.220 \DeclareFontShape{LHE}{redis}{bx}{n}{%
65.221 <5> <6> <7> <8> <9> <10> <10.95> <12>
65.222 <14.4> <17.28> <20.74> <24.88> redisb10}{ }
65.223 %%%%%%%%% Font/shape undefined, therefore substituted
65.224 \DeclareFontShape{LHE}{redis}{b}{n}{
65.225 <-nowarn> {<->sub * redis/bx/n}{ }
65.226 <+nowarn> {<->ssub * redis/bx/n}{ }
65.227 %%%%%%%%% Font/shape undefined, therefore substituted
65.228 \DeclareFontShape{LHE}{redis}{bx}{sl}{
65.229 <-nowarn> {<->sub * redis/bx/n}{ }
65.230 <+nowarn> {<->ssub * redis/bx/n}{ }
65.231 %%%%%%%%% Font/shape undefined, therefore substituted
65.232 \DeclareFontShape{LHE}{redis}{bx}{it}{
65.233 <-nowarn> {<->sub * redis/bx/n}{ }
65.234 <+nowarn> {<->ssub * redis/bx/n}{ }
65.235 </LHEredis>

```

65.5 The HE8encoding definition file

The Hebrew font encoding HE8 is based upon an extension by Microsoft to the ISO-8859-8 standard. This is an 8bit encoding. The extensions include hebrew points (“Nikud”).

First we define the Codepage 1255; specify a default for the font substitution process for the HE8 encoding and supply a font to be used when all else fails.

```

65.236 <*HE8enc>
65.237 \DeclareFontEncoding{HE8}{ }{ }
65.238 \DeclareFontSubstitution{HE8}{cmr}{m}{n}
65.239 \DeclareErrorFont{HE8}{cmr}{m}{n}{10}
65.240 </HE8enc>

```

Then we define a few commands in the HE8 encoding.

```

65.241 <*HE8enc>
65.242 \ProvideTextCommand{\textcopyright}{HE8}{\textcircled{\@latin{c}}}
65.243 \ProvideTextCommand{\textregistered}{HE8}{\textcircled{\scshape%
65.244 \@latin{r}}}
65.245 \ProvideTextCommand{\texttrademark}{HE8}{\textsuperscript{\@latin{TM}}}
65.246 </HE8enc>

```

65.5.1 CHECK HERE FOR HE8 UPDATES

Because not everyone can input Hebrew input text directly from the keyboard we need to define control sequences for all the Hebrew glyphs in the fonts. In addition, we want to support many input encodings for Hebrew and to keep the language definition file (`hebrew.ldf`) independent of the encoding. Therefore, we exploit the standard L^AT_EX 2_ε font encoding mechanism to define control sequences for all the Hebrew glyphs in the fonts in encoding-specific way. The language definition file uses only the control sequences and doesn’t need to check the current font or input encoding.

In the LHE encoding (7-bit encoding) all the Hebrew glyphs reside in the *lower* half of the font. Currently, only the Hebrew letters are supported. They use the same positions as the latin small letters in ASCII encoding and the position of ‘.

Some general symbols:

```
65.247 (*HE8enc)
65.248 \ProvideTextCommand{\textcopyright}{HE8}{\textcircled{\@latin{c}}}%
65.249 \ProvideTextCommand{\textregistered}{HE8}{\textcircled{\scshape%
65.250 \@latin{r}}}%
65.251 \ProvideTextCommand{\texttrademark}{HE8}{\textsuperscript{\@latin{TM}}}%
65.252 (/HE8enc)
```

The hebrew points:

```
65.253 (*HE8enc)
65.254 \DeclareTextSymbol{\sheva}{HE8}{192}
65.255 \DeclareTextSymbol{\hatafsegol}{HE8}{193}
65.256 \DeclareTextSymbol{\hatafpatah}{HE8}{194}
65.257 \DeclareTextSymbol{\hatafqamats}{HE8}{195}
65.258 \DeclareTextSymbol{\hiriq}{HE8}{196}
65.259 \DeclareTextSymbol{\tsere}{HE8}{197}
65.260 \DeclareTextSymbol{\segol}{HE8}{198}
65.261 \DeclareTextSymbol{\patah}{HE8}{199}
65.262 \DeclareTextSymbol{\qamats}{HE8}{200}
65.263 \DeclareTextSymbol{\holam}{HE8}{201}
65.264 \DeclareTextSymbol{\qubuts}{HE8}{203}
65.265 \DeclareTextSymbol{\dagesh}{HE8}{204}
65.266 \DeclareTextSymbol{\meteg}{HE8}{205}
65.267 \DeclareTextSymbol{\maqaf}{HE8}{206}
65.268 \DeclareTextSymbol{\rafe}{HE8}{207}
65.269 \DeclareTextSymbol{\paseq}{HE8}{208}
65.270 \DeclareTextSymbol{\shindot}{HE8}{209}
65.271 \DeclareTextSymbol{\sindot}{HE8}{210}
65.272 \DeclareTextSymbol{\sofpasuq}{HE8}{211}
65.273 \DeclareTextSymbol{\doublevav}{HE8}{212}
65.274 \DeclareTextSymbol{\vavyod}{HE8}{213}
65.275 \DeclareTextSymbol{\doubleyod}{HE8}{214}
65.276 (/HE8enc)
```

Hebrew letters occupy the positions 224–250 in HE8 encoding [WHAT ABOUT OTHER MARKS]:

```
65.277 (*HE8enc)
65.278 % \lccode‘=‘ % probably not needed (Tzafrir)
65.279 \DeclareTextSymbol{\hebalef}{HE8}{224}
65.280 \DeclareTextSymbol{\hebbet}{HE8}{225}
65.281 \DeclareTextSymbol{\hebgimel}{HE8}{226}
65.282 \DeclareTextSymbol{\hebdalet}{HE8}{227}
65.283 \DeclareTextSymbol{\hebhe}{HE8}{228}
65.284 \DeclareTextSymbol{\hebvav}{HE8}{229}
65.285 \DeclareTextSymbol{\hebzayin}{HE8}{230}
65.286 \DeclareTextSymbol{\hebbet}{HE8}{231}
65.287 \DeclareTextSymbol{\hebtet}{HE8}{232}
65.288 \DeclareTextSymbol{\hebyod}{HE8}{233}
65.289 \DeclareTextSymbol{\hebfinalkaf}{HE8}{234}
65.290 \DeclareTextSymbol{\hebkaf}{HE8}{235}
65.291 \DeclareTextSymbol{\heblamed}{HE8}{236}
65.292 \DeclareTextSymbol{\hebfinalmem}{HE8}{237}
65.293 \DeclareTextSymbol{\hebmam}{HE8}{238}
65.294 \DeclareTextSymbol{\hebfinalnun}{HE8}{239}
65.295 \DeclareTextSymbol{\hebnun}{HE8}{240}
65.296 \DeclareTextSymbol{\hebsamekh}{HE8}{241}
65.297 \DeclareTextSymbol{\hebayin}{HE8}{242}
65.298 \DeclareTextSymbol{\hebfinalpe}{HE8}{243}
65.299 \DeclareTextSymbol{\hebpe}{HE8}{244}
65.300 \DeclareTextSymbol{\hebfinaltsadi}{HE8}{245}
```

```

65.301 \DeclareTextSymbol{\hebtsadi}{HE8}{246}
65.302 \DeclareTextSymbol{\hebqof}{HE8}{247}
65.303 \DeclareTextSymbol{\hebresh}{HE8}{248}
65.304 \DeclareTextSymbol{\hebshin}{HE8}{249}
65.305 \DeclareTextSymbol{\hebtav}{HE8}{250}
65.306 \end{HE8enc}

```

Letter `\hebsin` is defined as a synonym of `\hebshin`:

```

65.307 \let\hebsin=\hebshin

```

65.6 The font definition files (in HE8 encoding)

65.6.1 Hebrew default font

It uses *OmegaHebrew* font for regular font, *Old Jaffa* font for italic shape and small-caps, *Dead Sea* font for bold face, and *Tel-Aviv* for bold-italic

```

65.308 (*HE8cmr)
65.309 \DeclareFontFamily{HE8}{cmr}{\hyphenchar\font45}
65.310 \DeclareFontShape{HE8}{cmr}{m}{n}
65.311     {<-> david }{}
65.312 %%%%%%%%% Italicized shape
65.313 \DeclareFontShape{HE8}{cmr}{m}{it}
65.314     {<-> davidi }{}
65.315 \DeclareFontShape{HE8}{cmr}{m}{sl}
65.316     {<-> davidi }{}
65.317 \DeclareFontShape{HE8}{cmr}{m}{sc}
65.318     {<-> david }{}
65.319 %%%%%%%%% Bold extended series
65.320 \DeclareFontShape{HE8}{cmr}{bx}{n}
65.321     {<-> davidb }{}
65.322 \DeclareFontShape{HE8}{cmr}{b}{n}
65.323     {<-> davidb }{}
65.324 %%%%%%%%% Bold extended (Italic) series
65.325 \DeclareFontShape{HE8}{cmr}{bx}{sl}
65.326     {<-> davidbi }{}
65.327 \DeclareFontShape{HE8}{cmr}{bx}{it}
65.328     {<-> davidbi }{}
65.329 \end{HE8cmr}

```

65.6.2 Hebrew sans-serif font

Until we have a real sans-serif font in this distribution, this file will remain a copy of the roman fonts definitions above.

```

65.330 (*HE8cmss)
65.331 \DeclareFontFamily{HE8}{cmss}{\hyphenchar\font45}
65.332 \DeclareFontShape{HE8}{cmss}{m}{n}
65.333     {<-> nachlieli }{}
65.334 %%%%%%%%% Italicized shape
65.335 \DeclareFontShape{HE8}{cmss}{m}{it}
65.336     {<-> nachlieli }{}
65.337 \DeclareFontShape{HE8}{cmss}{m}{sl}
65.338     {<-> nachlieli }{}
65.339 \DeclareFontShape{HE8}{cmss}{m}{sc}
65.340     {<-> nachlieli }{}
65.341 %%%%%%%%% Bold extended series
65.342 \DeclareFontShape{HE8}{cmss}{bx}{n}
65.343     {<-> nachlieli }{}
65.344 \DeclareFontShape{HE8}{cmss}{b}{n}
65.345     {<-> nachlieli }{}
65.346 %%%%%%%%% Bold extended (Italic) series
65.347 \DeclareFontShape{HE8}{cmss}{bx}{sl}
65.348     {<-> nachlieli }{}

```

```

65.349 \DeclareFontShape{HE8}{cmss}{bx}{it}
65.350      {<-> nachlieli }{}
65.351 /HE8cmss)

```

65.6.3 Hebrew typewriter font

Until we have a real sans-serif font in this distribution, this file will remain a copy of the roman fonts definitions above.

```

65.352 (*HE8cmtt)
65.353 \DeclareFontFamily{HE8}{cmtt}{\hyphenchar\font45}
65.354 \DeclareFontShape{HE8}{cmtt}{m}{n}
65.355      {<-> miriam }{}
65.356 %%%%%%%%% Italicized shape
65.357 \DeclareFontShape{HE8}{cmtt}{m}{it}
65.358      {<-> miriam }{}
65.359 \DeclareFontShape{HE8}{cmtt}{m}{sl}
65.360      {<-> miriam }{}
65.361 \DeclareFontShape{HE8}{cmtt}{m}{sc}
65.362      {<-> miriam }{}
65.363 %%%%%%%%% Bold extended series
65.364 \DeclareFontShape{HE8}{cmtt}{bx}{n}
65.365      {<-> miriam }{}
65.366 \DeclareFontShape{HE8}{cmtt}{b}{n}
65.367      {<-> miriam }{}
65.368 %%%%%%%%% Bold extended (Italic) series
65.369 \DeclareFontShape{HE8}{cmtt}{bx}{sl}
65.370      {<-> miriam }{}
65.371 \DeclareFontShape{HE8}{cmtt}{bx}{it}
65.372      {<-> miriam }{}
65.373 /HE8cmtt)

```

65.6.4 8Bit OmegaHebrew font

OmegaHebrew is a serif hebrew font created by the omega project [FILL IN CREDITS] [FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```

65.374 (*HE8OmegaHebrew)
65.375 \def\OmegaHebrewscale{0.9}
65.376 \DeclareFontFamily{HE8}{OmegaHebrew}{\hyphenchar\font45}
65.377 \DeclareFontShape{HE8}{OmegaHebrew}{m}{n}{<-> [\OmegaHebrewscale] OmegaHebrew }{}
65.378 %\endinput % is it needed [tzafrir]
65.379 /HE8OmegaHebrew)

```

65.6.5 8Bit Aharoni font

Aharoni is a serif hebrew font created by the omega project [FILL IN CREDITS] [FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```

65.380 (*HE8aharoni)
65.381 \def\Aharoniscale{1.0}
65.382 \DeclareFontFamily{HE8}{aharoni}{\hyphenchar\font45}
65.383 \DeclareFontShape{HE8}{aharoni}{m}{n} {<-> [\Aharoniscale] aharoni}{}
65.384 \DeclareFontShape{HE8}{aharoni}{m}{it} {<-> [\Aharoniscale] aharonii}{}
65.385 \DeclareFontShape{HE8}{aharoni}{m}{sl} {<-> [\Aharoniscale] aharonii}{}
65.386 \DeclareFontShape{HE8}{aharoni}{b}{n} {<-> [\Aharoniscale] aharonib}{}
65.387 \DeclareFontShape{HE8}{aharoni}{bx}{n} {<-> [\Aharoniscale] aharonib}{}
65.388 \DeclareFontShape{HE8}{aharoni}{bx}{it} {<-> [\Aharoniscale] aharonibi}{}
65.389
65.390 %\endinput % is it needed [tzafrir]
65.391 /HE8aharoni)

```

65.6.6 8Bit David font

David is a serif hebrew font created by the omega project [FILL IN CREDITS]
[FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```
65.392 (*HE8david)
65.393 \def\Davidscale{1.0}
65.394 \DeclareFontFamily{HE8}{david}{\hyphenchar\font45}
65.395
65.396 \DeclareFontShape{HE8}{david}{m}{n}    {<-> [\Davidscale] david}{ }
65.397 \DeclareFontShape{HE8}{david}{m}{it}  {<-> [\Davidscale] davidi}{ }
65.398 \DeclareFontShape{HE8}{david}{m}{sl}  {<-> [\Davidscale] davidi}{ }
65.399 \DeclareFontShape{HE8}{david}{b}{n}    {<-> [\Davidscale] davidb}{ }
65.400 \DeclareFontShape{HE8}{david}{b}{bx}{n} {<-> [\Davidscale] davidb}{ }
65.401 \DeclareFontShape{HE8}{david}{b}{it}  {<-> [\Davidscale] davidbi}{ }
65.402
65.403
65.404 %\endinput % is it needed [tzafrir]
65.405 \end{HE8david}
```

65.6.7 8Bit Drugulin font

Drugulin is a serif hebrew font created by the omega project [FILL IN CREDITS]
[FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```
65.406 (*HE8drugulin)
65.407 \def\Drugulinscale{1.0}
65.408 \DeclareFontFamily{HE8}{drugulin}{\hyphenchar\font45}
65.409 \DeclareFontShape{HE8}{drugulin}{m}{n}    {<-> [\Drugulinscale] drugulinb}{ }
65.410 \DeclareFontShape{HE8}{drugulin}{m}{it}  {<-> [\Drugulinscale] drugulinbi}{ }
65.411 \DeclareFontShape{HE8}{drugulin}{m}{sl}  {<-> [\Drugulinscale] drugulinbi}{ }
65.412 \DeclareFontShape{HE8}{drugulin}{b}{n}    {<-> [\Drugulinscale] drugulinb}{ }
65.413 \DeclareFontShape{HE8}{drugulin}{b}{bx}{n} {<-> [\Drugulinscale] drugulinb}{ }
65.414 \DeclareFontShape{HE8}{drugulin}{b}{it}  {<-> [\Drugulinscale] drugulinbi}{ }
65.415 %\endinput % is it needed [tzafrir]
65.416 \end{HE8drugulin}
```

65.6.8 8Bit Ellinia font

Ellinia is a sans-serif hebrew font created by the omega project [FILL IN CREDITS]
[FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```
65.417 (*HE8ellinia)
65.418 \def\Elliniascale{1.0}
65.419 \DeclareFontFamily{HE8}{ellinia}{\hyphenchar\font45}
65.420 \DeclareFontShape{HE8}{ellinia}{m}{n}    {<-> [\Elliniascale] ellinia}{ }
65.421 \DeclareFontShape{HE8}{ellinia}{m}{it}  {<-> [\Elliniascale] elliniaai}{ }
65.422 \DeclareFontShape{HE8}{ellinia}{m}{sl}  {<-> [\Elliniascale] elliniaai}{ }
65.423 \DeclareFontShape{HE8}{ellinia}{b}{n}    {<-> [\Elliniascale] elliniab}{ }
65.424 \DeclareFontShape{HE8}{ellinia}{b}{bx}{n} {<-> [\Elliniascale] elliniab}{ }
65.425 \DeclareFontShape{HE8}{ellinia}{b}{it}  {<-> [\Elliniascale] elliniabi}{ }
65.426 %\endinput % is it needed [tzafrir]
65.427 \end{HE8ellinia}
```

65.6.9 8Bit FrankRuehl font

FrankRuehl is a serif hebrew font created by the omega project [FILL IN CREDITS]
[FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```
65.428 (*HE8frankruehl)
65.429 \def\FrankRuehlscale{1.0}
65.430 \DeclareFontFamily{HE8}{frank}{\hyphenchar\font45}
65.431 \DeclareFontShape{HE8}{frank}{m}{n}    {<-> [\FrankRuehlscale] frank}{ }
65.432 \DeclareFontShape{HE8}{frank}{m}{it}  {<-> [\FrankRuehlscale] franki}{ }
65.433 \DeclareFontShape{HE8}{frank}{m}{sl}  {<-> [\FrankRuehlscale] franki}{ }
```

```

65.434 \DeclareFontShape{HE8}{frank}{b}{n}    {<-> [\FrankRuehlscale] frankb}{ }
65.435 \DeclareFontShape{HE8}{frank}{bx}{n}    {<-> [\FrankRuehlscale] frankb}{ }
65.436 \DeclareFontShape{HE8}{frank}{bx}{it}    {<-> [\FrankRuehlscale] frankbi}{ }
65.437 %\endinput % is it needed [tzafrir]
65.438 </HE8frankruehl>

```

65.6.10 8Bit KtavYad font

KtavYad is a serif hebrew font created by the omega project [FILL IN CREDITS]
[FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```

65.439 (*HE8yad)
65.440 \def\KtavYadscale{1.0}
65.441 \DeclareFontFamily{HE8}{yad}{\hyphenchar\font45}
65.442 \DeclareFontShape{HE8}{yad}{m}{n}    {<-> [\KtavYadscale] yadi}{ }
65.443 \DeclareFontShape{HE8}{yad}{m}{it}    {<-> [\KtavYadscale] yadi}{ }
65.444 \DeclareFontShape{HE8}{yad}{m}{sl}    {<-> [\KtavYadscale] yadi}{ }
65.445 \DeclareFontShape{HE8}{yad}{b}{n}    {<-> [\KtavYadscale] yadbi}{ }
65.446 \DeclareFontShape{HE8}{yad}{bx}{n}    {<-> [\KtavYadscale] yadbi}{ }
65.447 \DeclareFontShape{HE8}{yad}{bx}{it}    {<-> [\KtavYadscale] yadbi}{ }
65.448 %\endinput % is it needed [tzafrir]
65.449 </HE8yad>

```

65.6.11 8Bit MiriamMono font

MiriamMono is a serif hebrew font created by the omega project [FILL IN CREDITS]
[FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```

65.450 (*HE8miriam)
65.451 \def\MiriamMonoscale{1.0}
65.452 \DeclareFontFamily{HE8}{miriam}{\hyphenchar\font45}
65.453 \DeclareFontShape{HE8}{miriam}{m}{n}    {<-> [\MiriamMonoscale] miriam}{ }
65.454 \DeclareFontShape{HE8}{miriam}{m}{it}    {<-> [\MiriamMonoscale] miriami}{ }
65.455 \DeclareFontShape{HE8}{miriam}{m}{sl}    {<-> [\MiriamMonoscale] miriami}{ }
65.456 \DeclareFontShape{HE8}{miriam}{b}{n}    {<-> [\MiriamMonoscale] miriamb}{ }
65.457 \DeclareFontShape{HE8}{miriam}{bx}{n}    {<-> [\MiriamMonoscale] miriamb}{ }
65.458 \DeclareFontShape{HE8}{miriam}{bx}{it}    {<-> [\MiriamMonoscale] miriambi}{ }
65.459
65.460 %\endinput % is it needed [tzafrir]
65.461 </HE8miriam>

```

65.6.12 8Bit Nachlieli font

Nachlieli is a serif hebrew font created by the omega project [FILL IN CREDITS]
[FILL IN GENERAL SHAPE DESCRIPTION] shapes: [FILL IN]

```

65.462 (*HE8nachlieli)
65.463 \def\Nachlieliscscale{1.0}
65.464 \DeclareFontFamily{HE8}{nachlieli}{\hyphenchar\font45}
65.465 \DeclareFontShape{HE8}{nachlieli}{m}{n}    {<-> [\Nachlieliscscale] nachlieli}{ }
65.466 \DeclareFontShape{HE8}{nachlieli}{m}{it}    {<-> [\Nachlieliscscale] nachlielii}{ }
65.467 \DeclareFontShape{HE8}{nachlieli}{m}{sl}    {<-> [\Nachlieliscscale] nachlielii}{ }
65.468 \DeclareFontShape{HE8}{nachlieli}{b}{n}    {<-> [\Nachlieliscscale] nachlielib}{ }
65.469 \DeclareFontShape{HE8}{nachlieli}{bx}{n}    {<-> [\Nachlieliscscale] nachlielib}{ }
65.470 \DeclareFontShape{HE8}{nachlieli}{bx}{it}    {<-> [\Nachlieliscscale] nachlielib}{ }
65.471 %\endinput % is it needed [tzafrir]
65.472 </HE8nachlieli>

```

65.6.13 Hebrew font switching commands

The `hebfont` package defines a number of high-level commands (all starting with `\text..` similar to the standard $\text{\LaTeX 2}_{\epsilon}$ font-change commands, for example `\textbf`) that have one argument and typeset this argument in the requested

<i>Command</i>	<i>Corresponds to</i>	<i>Font family</i>
<code>\textjm{..}</code>	<code>\rmfamily</code>	Jerusalem font
<code>\textds{..}</code>	<code>\bfseries</code>	Dead Sea font
<code>\textoj{..}</code>	<code>\itshape</code> <code>\slshape</code> <code>\emph</code>	Old Jaffa font
<code>\textta{..}</code>	<code>\sffamily</code> <code>\ttfamily</code>	Tel-Aviv font
<code>\textcrml{..}</code>	<code>\fontfamily{crml}</code>	Carmel fonts
<code>\textfr{..}</code>	<code>\fontfamily{fr}</code>	Frank-Ruehl fonts
<code>\textredis{..}</code>	<code>\fontfamily{redis}</code>	Redis fonts
<code>\textclas{..}</code>	<code>\fontfamily{redis}</code>	Classic fonts
<code>\textshold{..}</code>	<code>\fontfamily{shold}</code>	Shalom Old Style font
<code>\textshscr{..}</code>	<code>\fontfamily{shscr}</code>	Shalom Script font
<code>\textshstk{..}</code>	<code>\fontfamily{shstk}</code>	Shalom Stick font

Table 36: Hebrew font-change commands with arguments

The font change commands provided here all start with `\text..` to emphasize that they are for use in normal text and to be easily memorable.

way. These commands are defined for all available Hebrew fonts defined above and change only font parameters but not direction.

For example, to use Hebrew Classic font family, the following sequence of commands should be included in a $\text{\LaTeX 2}_{\epsilon}$ document:

```
\sethebrew
\textclas{Hebrew text printed with Classic fonts}
```

or to use Hebrew with Classic fonts locally:

```
\R{\textclas{Hebrew text printed with Classic fonts}}
```

We declare $\text{\LaTeX 2}_{\epsilon}$ font commands, e.g. `\textjm{...}` for all available fonts. Table 36 shows the meanings of all these new high-level commands.

`\textjm` Switches to *Jerusalem* font which is default regular Hebrew font (“roman” family). Commands `\textrm{...}` and old-style `\rm ...` will produce the same result.

```
65.473 (*hebfont)
65.474 \def\ivritex@tmp{HE8}
65.475 \ifx\ivritex@tmp\HeblatexEncoding %
65.476 % compatibility with hebfnts:
65.477 \DeclareTextFontCommand{\textjm}{\rmfamily\selectfont}
65.478 \DeclareTextFontCommand{\textds}{\bfseries\selectfont}
65.479 \DeclareTextFontCommand{\textoj}{\itshape\selectfont}
65.480 \DeclareTextFontCommand{\textta}{\sffamily\selectfont}
65.481
65.482 % an attempt to give some replacements to the original hebfonts:
65.483 %
65.484 \DeclareTextFontCommand{\textcrml}{\fontfamily{david}\selectfont}
65.485 \DeclareTextFontCommand{\textfr}{\fontfamily{frank}\selectfont}
65.486 \DeclareTextFontCommand{\textredis}{\fontfamily{aharoni}\selectfont}
65.487 \DeclareTextFontCommand{\textclas}{\fontfamily{drugulin}\selectfont}
65.488 \DeclareTextFontCommand{\textshold}{\fontfamily{frank}\selectfont}
65.489 \DeclareTextFontCommand{\textshscr}{\fontfamily{yad}\selectfont}
65.490 \DeclareTextFontCommand{\textshstk}{\fontfamily{aharoni}\selectfont}
65.491 % note that redis is larger than shstk
```

```

65.492
65.493
65.494 \DeclareTextFontCommand{\textaha}{\fontfamily{aharoni}\selectfont}
65.495 \DeclareTextFontCommand{\textdav}{\fontfamily{david}\selectfont}
65.496 \DeclareTextFontCommand{\textdru}{\fontfamily{drugulin}\selectfont}
65.497 \DeclareTextFontCommand{\textel}{\fontfamily{ellinia}\selectfont}
65.498 % \textfr is already declared above
65.499 \DeclareTextFontCommand{\textmir}{\fontfamily{miriam}\selectfont}
65.500 \DeclareTextFontCommand{\textna}{\fontfamily{nachlieli}\selectfont}
65.501 % is this necessary:
65.502 \DeclareTextFontCommand{\textyad}{\fontfamily{yad}\selectfont}
65.503
65.504 \else%
65.505 \DeclareTextFontCommand{\textjm}{\rmfamily\selectfont}

\textds Switches to Dead Sea font which is default bold font in Hebrew. Commands
\textbf{...} and old-style {\bf ...} will produce the same result.
65.506 \DeclareTextFontCommand{\textds}{\bfseries\selectfont}

\textoj Switches to Old Jaffa font which is default italic font in Hebrew. Commands
\textit{...}, \textsl{...}, \emph{...} and old-style {\it ...} or {\em ...}
will produce the same result.
65.507 \DeclareTextFontCommand{\textoj}{\itshape\selectfont}

\textta Switches to Tel-Aviv font which is default sans-serif font in Hebrew. Commands
\textsf{...}, \texttt{...} and old-style {\sf ...} or {\tt ...} will produce
the same result (because sans-serif is used as typewriter font when in Hebrew
mode).
65.508 \DeclareTextFontCommand{\textta}{\sffamily\selectfont}

\textcrl Switches to Carmel font. Regular and slanted variants of carmel font will be used..
65.509 \DeclareTextFontCommand{\textcrl}{\fontfamily{crl}\selectfont}

\textfr Switches to Frank-Ruehl font family. Regular, bold and slanted frank ruehl fonts
will be used.
65.510 \DeclareTextFontCommand{\textfr}{\fontfamily{fr}\selectfont}

\textredis Switches to Redis font family. Regular, bold and slanted redis fonts of various
sizes will be used.
65.511 \DeclareTextFontCommand{\textredis}{\fontfamily{redis}\selectfont}

\textclas Switches to Classic font family. The normal font will be hclassic and slanted —
hcaption.
65.512 \DeclareTextFontCommand{\textclas}{\fontfamily{clas}\selectfont}

\textshold Switches to Shalom Old Style font.
65.513 \DeclareTextFontCommand{\textshold}{\fontfamily{shold}\selectfont}

\textshscr Switches to Shalom Script font.
65.514 \DeclareTextFontCommand{\textshscr}{\fontfamily{shscr}\selectfont}

\textshstk Switches to Shalom Stick font.
65.515 \DeclareTextFontCommand{\textshstk}{\fontfamily{shstk}\selectfont}
65.516 \fi

Finally, for backward compatibility with LATEX2.09, four old font commands,
e.g. {\jm ...} are defined too (see Table 37).
65.517 \if@compatibility
65.518 \DeclareOldFontCommand{\jm}{\normalfont\rmfamily\selectfont}%
65.519 {\@nomath\jm}

```

<i>Old font command</i>	<i>Font name</i>	<i>Comment</i>
<code>\jm ..}</code>	Jerusalem	default regular (roman) font
<code>\ds ..}</code>	Dead Sea	default bold font
<code>\oj ..}</code>	Old Jaffa	default italic and slanted font used also to emphasize text
<code>\ta ..}</code>	Tel-Aviv	default sans-serif and typewriter font

Table 37: Hebrew old font-change commands for compatibility mode

```

65.520 \DeclareOldFontCommand{\ds}{\normalfont\bfseries\selectfont}%
65.521         {\@nomath\ds}
65.522 \DeclareOldFontCommand{\oj}{\normalfont\itshape\selectfont}%
65.523         {\@nomath\oj}
65.524 \DeclareOldFontCommand{\ta}{\normalfont\sffamily\selectfont}%
65.525         {\@nomath\ta}
65.526 \fi
65.527 \end{hebrewfont}

```

66 Hebrew in L^AT_EX 2.09 compatibility mode

`\documentstyle` command in the preamble of L^AT_EX document indicates that it is a L^AT_EX 2.09 document, and should be processed in *compatibility mode*. In such documents, one of the following three Hebrew style options can be included:

1. **hebrew_newcode** indicates that document will use UNIX ISO 8859-8 or Windows cp1255 input encoding, i.e. *Alef* letter will be represented as 224.
2. **hebrew_p** indicates that document is encoded with IBM PC cp862 encoding, i.e. *Alef* letter will be represented as 128.
3. **hebrew_oldcode** indicates that document uses old 7-bit encoding, as defined in Israeli Standard 960, i.e. *Alef* is character number 96.

Note, that other hebrew-related styles, such as **hebcsl** can be included *after* the abovenamed Hebrew style option, for example:

```
\documentstyle[12pt,hebrew_p,hebcsl]{report}.
```

Any Hebrew document which compiled under L^AT_EX 2.09 should compile under compatibility mode, unless it uses low-level commands such as `\tenrm`.

66.1 The DOCSTRIP modules

The following modules are used in the implementation to direct DOCSTRIP in generating the external files:

```

newcode  produce hebrew_newcode.sty
pccode   produce hebrew_p.sty
oldcode  produce hebrew_oldcode.sty

```

66.2 Obsolete style files

For each of the Hebrew L^AT_EX 2.09 Hebrew styles, we produce a file which uses correct input encoding and calls **babel** with Hebrew and English language options. This means that any styles which say `\input hebrew_newcode.sty` or `\documentstyle[...hebrew_newcode...]{...}` should still work.

```

66.1 (*newcode | pccode | oldcode)
66.2 \NeedsTeXFormat{LaTeX2e}
66.3 \end{newcode | pccode | oldcode}

```



```

66.4 \*newcode>
66.5 \@obsoletedefine{hebrew.sty}{hebrew_newcode.sty}
66.6 \RequirePackage[8859-8]{inputenc}
66.7 \*newcode>
66.8 \*pccode>
66.9 \@obsoletedefine{hebrew.sty}{hebrew_p.sty}
66.10 \RequirePackage[cp862]{inputenc}
66.11 \*pccode>
66.12 \*oldcode>
66.13 \@obsoletedefine{hebrew.sty}{hebrew_oldcode.sty}
66.14 \RequirePackage[si960]{inputenc}
66.15 \*oldcode>
66.16 \*newcode | pccode | oldcode>
66.17 \RequirePackage[english,hebrew]{babel}
66.18 \*newcode | pccode | oldcode>

```

67 The Bahasa Indonesian language

The file `bahasa.dtx`⁷⁹ defines all the language definition macros for the Bahasa Indonesia / Bahasa Melayu language. Bahasa just means ‘language’ in Bahasa Indonesia / Bahasa Melayu. Since both national versions of the language use the same writing, although differing in pronunciation, this file can be used for both languages.

For this language currently no special definitions are needed or available.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
67.1 (*code)
67.2 \LdfInit\CurrentOption{date\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `bahasa` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@bahasa` to see whether we have to do something here.

For both Bahasa Indonesia and Bahasa Malaysia the same set of hyphenation patterns can be used which are available in the file `inhyph.tex`. However it could be loaded using any of the possible Babel options for the Indonesian and Malaysian language. So first we try to find out whether this is the case.

```
67.3 \ifx\l@bahasa\@undefined
67.4   \ifx\l@bahasai\@undefined
67.5     \ifx\l@indon\@undefined
67.6       \ifx\l@indonesian\@undefined
67.7         \ifx\l@bahasam\@undefined
67.8           \ifx\l@malay\@undefined
67.9             \ifx\l@meyalu\@undefined
67.10              \nopatterns{Bahasa Indonesia}
67.11              \adddialect\l@bahasa0\relax
67.12            \else
67.13              \let\l@bahasa\l@meyalu
67.14            \fi
67.15          \else
67.16            \let\l@bahasa\l@malay
67.17          \fi
67.18        \else
67.19          \let\l@bahasa\l@bahasam
67.20        \fi
67.21      \else
67.22        \let\l@bahasa\l@indonesian
67.23      \fi
67.24    \else
67.25      \let\l@bahasa\l@indon
67.26    \fi
67.27  \else
67.28    \let\l@bahasa\l@bahasai
67.29  \fi
67.30 \fi
```

Now that we are sure the `\l@bahasa` has some valid definition we need to make sure that a name to access the hyphenation patterns, corresponding to the option used, is available.

```
67.31 \expandafter\expandafter\expandafter\let
67.32   \expandafter\csname
67.33   \expandafter l\expandafter @\CurrentOption\endcsname
67.34   \l@bahasa
```

The next step consists of defining commands to switch to (and from) the Bahasa language.

⁷⁹The file described in this section has version number v1.0l and was last revised on 2008/03/15.

`\captionsbahasa` The macro `\captionsbahasa` defines all strings used in the four standard documentclasses provided with L^AT_EX.

```

67.35 \@namedef{captions\CurrentOption}{%
67.36   \def\prefacename{Pendahuluan}%
67.37   \def\refname{Pustaka}%
67.38   \def\abstractname{Ringkasan}% (sometime it's called 'intisari'
67.39                               % or 'ikhtisar')
67.40   \def\bibname{Bibliografi}%
67.41   \def\chaptername{Bab}%
67.42   \def\appendixname{Lampiran}%
67.43   \def\contentsname{Daftar Isi}%
67.44   \def\listfigurename{Daftar Gambar}%
67.45   \def\listtablename{Daftar Tabel}%
67.46   \def\indexname{Indeks}%
67.47   \def\figurename{Gambar}%
67.48   \def\tablename{Tabel}%
67.49   \def\partname{Bagian}%
67.50 % Subject: Subyek
67.51 % From: Dari
67.52   \def\enclname{Lampiran}%
67.53   \def\ccname{cc}%
67.54   \def\headtoname{Kepada}%
67.55   \def\pagename{Halaman}%
67.56 % Notes (Endnotes): Catatan
67.57   \def\seename{lihat}%
67.58   \def\alsoname{lihat juga}%
67.59   \def\proofname{Bukti}%
67.60   \def\glossaryname{Daftar Istilah}%
67.61 }
```

`\datebahasa` The macro `\datebahasa` redefines the command `\today` to produce Bahasa Indonesian dates.

```

67.62 \@namedef{date\CurrentOption}{%
67.63   \def\today{\number\day~\ifcase\month\or
67.64     Januari\or Pebruari\or Maret\or April\or Mei\or Juni\or
67.65     Juli\or Agustus\or September\or Oktober\or Nopember\or Desember\fi
67.66     \space \number\year}}
```

`\extrasbahasa` The macro `\extrasbahasa` will perform all the extra definitions needed for the Bahasa language. The macro `\extrasbahasa` is used to cancel the actions of `\noextrasbahasa`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```

67.67 \@namedef{extras\CurrentOption}{}
67.68 \@namedef{noextras\CurrentOption}{}

```

`\bahasahyphenmins` The bahasa hyphenation patterns should be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 2.

```

67.69 \providehyphenmins{\CurrentOption}{\tw@\tw@}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

67.70 \ldf@finish{\CurrentOption}
67.71 /code>

```

68 The Bahasa Malaysia language

The file `bahasam.dtx`⁸⁰ defines all the language definition macros for the Bahasa Malaysia language. Bahasa just means ‘language’ in Bahasa Malaysia. A number of terms differ from those used in bahasa indonesia.

For this language currently no special definitions are needed or available.

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
68.1 (*code)
68.2 \LdfInit\CurrentOption{date\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `bahasa` could be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@bahasa` to see whether we have to do something here.

For both Bahasa Malaysia and Bahasa Indonesia the same set of hyphenation patterns can be used which are available in the file `inhyp.tex`. However it could be loaded using any of the possible Babel options for the Malaysian and Indonesian language. So first we try to find out whether this is the case.

```
68.3 \ifx\l@malay\@undefined
68.4   \ifx\l@meyalu\@undefined
68.5     \ifx\l@bahasam\@undefined
68.6       \ifx\l@bahasa\@undefined
68.7         \ifx\l@bahasai\@undefined
68.8           \ifx\l@indon\@undefined
68.9             \ifx\l@indonesian\@undefined
68.10              \nopatterns{Bahasa Malaysia}
68.11              \adddialect\l@malay0\relax
68.12            \else
68.13              \let\l@malay\l@indonesian
68.14            \fi
68.15          \else
68.16            \let\l@malay\l@indon
68.17          \fi
68.18        \else
68.19          \let\l@malay\l@bahasai
68.20        \fi
68.21      \else
68.22        \let\l@malay\l@bahasa
68.23      \fi
68.24    \else
68.25      \let\l@malay\l@bahasam
68.26    \fi
68.27  \else
68.28    \let\l@malay\l@meyalu
68.29  \fi
68.30 \fi
```

Now that we are sure the `\l@malay` has some valid definition we need to make sure that a name to access the hyphenation patterns, corresponding to the option used, is available.

```
68.31 \expandafter\expandafter\expandafter\let
68.32   \expandafter\csname
68.33   \expandafter \l\expandafter @\CurrentOption\endcsname
68.34   \l@malay
```

The next step consists of defining commands to switch to (and from) the Bahasa language.

`\captionsbahasam` The macro `\captionsbahasam` defines all strings used in the four standard documentclasses provided with L^AT_EX.

⁸⁰The file described in this section has version number v1.0k and was last revised on 2008/01/27.

```

68.35 \@namedef{captions\CurrentOption}{%
68.36   \def\prefacename{Prakata}%
68.37   \def\refname{Rujukan}%
68.38   \def\abstractname{Abstrak}% (sometime it's called 'intisari'
68.39                               % or 'ikhtisar')
68.40   \def\bibname{Bibliografi}%
68.41   \def\chaptername{Bab}%
68.42   \def\appendixname{Lampiran}%
68.43   \def\contentsname{Kandungan}%
68.44   \def\listfigurename{Senarai Gambar}%
68.45   \def\listtablename{Senarai Jadual}%
68.46   \def\indexname{Indeks}%
68.47   \def\figurename{Gambar}%
68.48   \def\tablename{Jadual}%
68.49   \def\partname{Bahagian}%
68.50 % Subject: Perkara
68.51 % From: Dari
68.52   \def\enclname{Lampiran}%
68.53   \def\ccname{sk}% (short form for 'Salinan Kepada')
68.54   \def\headtoname{Kepada}%
68.55   \def\pagename{Halaman}%
68.56 % Notes (Endnotes): Catatan
68.57   \def\seename{sila rujuk}%
68.58   \def\alsoname{rujuk juga}%
68.59   \def\proofname{Bukti}%
68.60   \def\glossaryname{Istilah}%
68.61 }

```

`\datebahasam` The macro `\datebahasam` redefines the command `\today` to produce Bahasa Malaysian dates.

```

68.62 \@namedef{date\CurrentOption}{%
68.63   \def\today{\number\day~\ifcase\month\or
68.64     Januari\or Februari\or Mac\or April\or Mei\or Jun\or
68.65     Julai\or Ogos\or September\or Oktober\or November\or Disember\fi
68.66     \space \number\year}}

```

`\extrasbahasam` The macro `\extrasbahasa` will perform all the extra definitions needed for the Bahasa language. The macro `\extrasbahasa` is used to cancel the actions of `\extrasbahasa`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```

68.67 \@namedef{extras\CurrentOption}{}
68.68 \@namedef{noextras\CurrentOption}{}

```

`\bahasamhyphenmins` The bahasam hyphenation patterns should be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 2.

```

68.69 \providehyphenmins{\CurrentOption}{\tw@\tw@}

```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```

68.70 \ldf@finish{\CurrentOption}
68.71 </code>

```

69 Not renaming hyphen.tex

As Don Knuth has declared that the filename `hyphen.tex` may only be used to designate *his* version of the american English hyphenation patterns, a new solution has to be found in order to be able to load hyphenation patterns for other languages in a plain-based T_EX-format. When asked he responded:

That file name is "sacred", and if anybody changes it they will cause severe upward/downward compatibility headaches.

People can have a file `locallyhyphen.tex` or whatever they like, but they mustn't diddle with `hyphen.tex` (or `plain.tex` except to preload additional fonts).

The files `bplain.tex` and `blplain.tex` can be used as replacement wrappers around `plain.tex` and `lplain.tex` to achieve the desired effect, based on the `babel` package. If you load each of them with `iniTEX`, you will get a file called either `bplain.fmt` or `blplain.fmt`, which you can use as replacements for `plain.fmt` and `lplain.fmt`.

As these files are going to be read as the first thing `iniTEX` sees, we need to set some category codes just to be able to change the definition of `\input`

```
69.1 <*bplain | blplain>
69.2 \catcode'\{=1 % left brace is begin-group character
69.3 \catcode'\}=2 % right brace is end-group character
69.4 \catcode'\#=6 % hash mark is macro parameter character
```

Now let's see if a file called `hyphen.cfg` can be found somewhere on T_EX's input path by trying to open it for reading...

```
69.5 \openin 0 hyphen.cfg
```

If the file wasn't found the following test turns out true.

```
69.6 \ifeof0
69.7 \else
```

When `hyphen.cfg` could be opened we make sure that *it* will be read instead of the file `hyphen.tex` which should (according to Don Knuth's ruling) contain the american English hyphenation patterns and nothing else.

We do this by first saving the original meaning of `\input` (and I use a one letter control sequence for that so as not to waste multi-letter control sequence on this in the format).

```
69.8 \let\input
```

Then `\input` is defined to forget about its argument and load `hyphen.cfg` instead.

```
69.9 \def\input #1 {%
69.10 \let\input\input
69.11 \input hyphen.cfg
```

Once that's done the original meaning of `\input` can be restored and the definition of `\input` can be forgotten.

```
69.12 \let\input\input
69.13 }
69.14 \fi
69.15 </bplain | blplain>
```

Now that we have made sure that `hyphen.cfg` will be loaded at the right moment it is time to load `plain.tex`.

```
69.16 <bplain>\input plain.tex
69.17 <blplain>\input lplain.tex
```

Finally we change the contents of `\fmtname` to indicate that this is *not* the plain format, but a format based on plain with the `babel` package preloaded.

```
69.18 <bplain>\def\fmtname{babel-plain}
69.19 <blplain>\def\fmtname{babel-lplain}
```

When you are using a different format, based on `plain.tex` you can make a copy of `blplain.tex`, rename it and replace `plain.tex` with the name of your format file.

70 Support for formats based on PLAIN_{TEX}

The following code duplicates or emulates parts of L^AT_EX 2_ε that are needed for babel.

```
70.1 (*code)
70.2 \ifx\adddialect\@undefined
```

When `\adddialect` is still undefined we are making a format. In that case only the first part of this file is needed.

```
70.3 \def\@empty{}
```

We need to define `\loadlocalcfg` for plain users as the L^AT_EX definition uses `\InputIfFileExists`.

```
70.4 \def\loadlocalcfg#1{%
70.5   \openin0#1.cfg
70.6   \ifeof0
70.7     \closein0
70.8   \else
70.9     \closein0
70.10    {\immediate\write16{*****}%
70.11     \immediate\write16{* Local config file #1.cfg used}%
70.12     \immediate\write16{*}%
70.13    }
70.14    \input #1.cfg\relax
70.15  \fi
```

We have to execute `\@endofldf` in this case

```
70.16 \endofldf
70.17 }
```

We want to add a message to the message L^AT_EX 2.09 puts in the `\everyjob` register. This could be done by the following code:

```
\let\orgeveryjob\everyjob
\def\everyjob#1{%
  \orgeveryjob{#1}%
  \orgeveryjob\expandafter{\the\orgeveryjob\immediate\write16{%
    hyphenation patterns for \the\loaded@patterns loaded.}}%
  \let\everyjob\orgeveryjob\let\orgeveryjob\@undefined}
```

The code above redefines the control sequence `\everyjob` in order to be able to add something to the current contents of the register. This is necessary because the processing of hyphenation patterns happens long before L^AT_EX fills the register.

There are some problems with this approach though.

- When someone wants to use several hyphenation patterns with S_LT_EX the above scheme won't work. The reason is that S_LT_EX overwrites the contents of the `\everyjob` register with its own message.
- Plain T_EX does not use the `\everyjob` register so the message would not be displayed.

To circumvent this a 'dirty trick' can be used. As this code is only processed when creating a new format file there is one command that is sure to be used, `\dump`. Therefore the original `\dump` is saved in `\org@dump` and a new definition is supplied.

```
70.18 \let\org@dump=\dump
70.19 \def\dump{%
```

To make sure that L^AT_EX 2.09 executes the `\@begindocumenthook` we would want to alter `\begin{document}`, but as this done too often already, we add the new code at the front of `\@preamblecmds`. But we can only do that after it has been defined, so we add this piece of code to `\dump`.

```

70.20 \ifx\@ztryfc\@undefined
70.21 \else
70.22 \toks0=\expandafter{\@preamblecmds}
70.23 \edef\@preamblecmds{\noexpand\@begindocumenthook\the\toks0}
70.24 \def\@begindocumenthook{}
70.25 \fi

```

This new definition starts by adding an instruction to write a message on the terminal and in the transcript file to inform the user of the preloaded hyphenation patterns.

```

70.26 \everyjob\expandafter{\the\everyjob%
70.27 \immediate\write16{\the\toks8 loaded.}}%

```

Then everything is restored to the old situation and the format is dumped.

```

70.28 \let\dump\orig@dump\let\orig@dump\@undefined\dump}
70.29 \expandafter\endinput
70.30 \fi

```

The rest of this file is not processed by \LaTeX but during the normal document run. A number of \LaTeX macro's that are needed later on.

```

70.31 \long\def\@firstofone#1{#1}
70.32 \long\def\@firstoftwo#1#2{#1}
70.33 \long\def\@secondoftwo#1#2{#2}
70.34 \def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
70.35 \def\@star@or@long#1{%
70.36 \@ifstar
70.37 {\let\l@ngrel@x\relax#1}%
70.38 {\let\l@ngrel@x\long#1}}
70.39 \let\l@ngrel@x\relax
70.40 \def\@car#1#2\@nil{#1}
70.41 \def\@cdr#1#2\@nil{#2}
70.42 \let\@typeset@protect\relax
70.43 \long\def\@gobble#1{}
70.44 \edef\@backslashchar{\expandafter\@gobble\string\\}
70.45 \def\strip@prefix#1>{}
70.46 \def\g@addto@macro#1#2{{%
70.47 \toks0\expandafter{#1#2}%
70.48 \xdef#1{\the\toks0}}}
70.49 \def\@namedef#1{\expandafter\def\csname #1\endcsname}
70.50 \def\@ifundefined#1{%
70.51 \expandafter\ifx\csname#1\endcsname\relax
70.52 \expandafter\@firstoftwo
70.53 \else
70.54 \expandafter\@secondoftwo
70.55 \fi}

```

$\text{\LaTeX}_{2\epsilon}$ has the command `\@onlypreamble` which adds commands to a list of commands that are no longer needed after `\begin{document}`.

```

70.56 \ifx\@preamblecmds\@undefined
70.57 \def\@preamblecmds{}
70.58 \fi
70.59 \def\@onlypreamble#1{%
70.60 \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
70.61 \@preamblecmds\do#1}}
70.62 \@onlypreamble\@onlypreamble

```

Mimick \LaTeX 's `\AtBeginDocument`; for this to work the user needs to add `\begin{document}` to his file.

```

70.63 \def\begin{document}{%
70.64 \@begindocumenthook
70.65 \global\let\@begindocumenthook\@undefined
70.66 \def\do##1{\global\let ##1\@undefined}%
70.67 \@preamblecmds
70.68 \global\let\do\noexpand
70.69 }

```



```

70.70 \ifx\@begindocumenthook\@undefined
70.71   \def\@begindocumenthook{}
70.72 \fi
70.73 \@onlypreamble\@begindocumenthook
70.74 \def\AtBeginDocument{\g@addto@macro\@begindocumenthook}

```

We also have to mimick L^AT_EX's \AtEndOfPackage. Our replacement macro is much simpler; it stores its argument in \@endoflfd.

```

70.75 \def\AtEndOfPackage#1{\g@addto@macro\@endoflfd{#1}}
70.76 \@onlypreamble\AtEndOfPackage
70.77 \def\@endoflfd{}
70.78 \@onlypreamble\@endoflfd

```

L^AT_EX needs to be able to switch off writing to its auxiliary files; plain doesn't have them by default.

```

70.79 \ifx\if@files\@undefined
70.80   \expandafter\let\csname if@files\expandafter\endcsname
70.81     \csname iffalse\endcsname
70.82 \fi

```

Mimick L^AT_EX's commands to define control sequences.

```

70.83 \def\newcommand{\@star@or@long\newcommand}
70.84 \def\new@command#1{%
70.85   \@testopt{\@newcommand#1}0}
70.86 \def\@newcommand#1[#2]{%
70.87   \@ifnextchar [{\@xargdef#1[#2]}%
70.88     {\@argdef#1[#2]}}
70.89 \long\def\@argdef#1[#2]#3{%
70.90   \@yargdef#1\@ne{#2}{#3}}
70.91 \long\def\@xargdef#1[#2] [#3]#4{%
70.92   \expandafter\def\expandafter#1\expandafter{%
70.93     \expandafter\@protected@testopt\expandafter #1%
70.94     \csname\string#1\expandafter\endcsname{#3}}}%
70.95   \expandafter\@yargdef \csname\string#1\endcsname
70.96   \tw@{#2}{#4}}
70.97 \long\def\@yargdef#1#2#3{%
70.98   \@tempcnta#3\relax
70.99   \advance \@tempcnta \@ne
70.100   \let\@hash@\relax
70.101   \edef\reserved@a{\ifx#2\tw@ [\@hash@1]\fi}%
70.102   \@tempcntb #2%
70.103   \@whilenum\@tempcntb <\@tempcnta
70.104     \do{%
70.105       \edef\reserved@a{\reserved@a\@hash@\the\@tempcntb}%
70.106       \advance\@tempcntb \@ne}%
70.107   \let\@hash@##%
70.108   \l@ngrel@x\expandafter\def\expandafter#1\reserved@a}
70.109 \let\providecommand\newcommand

70.110 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
70.111 \def\declare@robustcommand#1{%
70.112   \edef\reserved@a{\string#1}%
70.113   \def\reserved@b{#1}%
70.114   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
70.115   \edef#1{%
70.116     \ifx\reserved@a\reserved@b
70.117       \noexpand\x@protect
70.118       \noexpand#1%
70.119     \fi
70.120     \noexpand\protect
70.121     \expandafter\@noexpand\csname
70.122       \expandafter\@gobble\string#1 \endcsname
70.123   }%
70.124   \expandafter\new@command\csname

```

```

70.125 \expandafter\@gobble\string#1 \endcsname
70.126 }
70.127 \def\x@protect#1{%
70.128 \ifx\protect\@typeset@protect\else
70.129 \x@protect#1%
70.130 \fi
70.131 }
70.132 \def\x@protect#1\fi#2#3{%
70.133 \fi\protect#1%
70.134 }

```

The following little macro `\in@` is taken from `latex.ltx`; it checks whether its first argument is part of its second argument. It uses the boolean `\in@`; allocating a new boolean inside conditionally executed code is not possible, hence the construct with the temporary definition of `\bbl@tmpa`.

```

70.135 \def\bbl@tmpa{\csname newif\endcsname\in@}
70.136 \ifx\in@\@undefined
70.137 \def\in@#1#2{%
70.138 \def\in@@##1#1##2##3\in@@{%
70.139 \ifx\in@@##2\in@false\else\in@true\fi}%
70.140 \in@@##2#1\in@\in@@}
70.141 \else
70.142 \let\bbl@tmpa\@empty
70.143 \fi
70.144 \bbl@tmpa

```

\LaTeX has a macro to check whether a certain package was loaded with specific options. The command has two extra arguments which are code to be executed in either the true or false case. This is used to detect whether the document needs one of the accents to be activated (`activegrave` and `activeacute`). For plain \TeX we assume that the user wants them to be active by default. Therefore the only thing we do is execute the third argument (the code for the true case).

```

70.145 \def\@ifpackagewith#1#2#3#4{%
70.146 #3}

```

The \LaTeX macro `\@ifloaded` checks whether a file was loaded. This functionality is not needed for plain \TeX but we need the macro to be defined as a no-op.

```

70.147 \def\@ifloaded#1#2#3#4{}

```

For the following code we need to make sure that the commands `\newcommand` and `\providecommand` exist with some sensible definition. They are not fully equivalent to their $\text{\LaTeX} 2_{\epsilon}$ versions; just enough to make things work in plain \TeX environments.

```

70.148 \ifx\@tempcnta\@undefined
70.149 \csname newcount\endcsname\@tempcnta\relax
70.150 \fi
70.151 \ifx\@tempcntb\@undefined
70.152 \csname newcount\endcsname\@tempcntb\relax
70.153 \fi

```

To prevent wasting two counters in $\text{\LaTeX} 2.09$ (because counters with the same name are allocated later by it) we reset the counter that holds the next free counter (`\count10`).

```

70.154 \ifx\bye\@undefined
70.155 \advance\count10 by -2\relax
70.156 \fi
70.157 \ifx\@ifnextchar\@undefined
70.158 \def\@ifnextchar#1#2#3{%
70.159 \let\reserved@d=#1%
70.160 \def\reserved@a{#2}\def\reserved@b{#3}%
70.161 \futurelet\@let@token\@ifnch}
70.162 \def\@ifnch{%
70.163 \ifx\@let@token\@sptoken

```

```

70.164 \let\reserved@c\@xifnch
70.165 \else
70.166 \ifx\@let@token\reserved@d
70.167 \let\reserved@c\reserved@a
70.168 \else
70.169 \let\reserved@c\reserved@b
70.170 \fi
70.171 \fi
70.172 \reserved@c}
70.173 \def\:{\let\@sptoken= } \: % this makes \@sptoken a space token
70.174 \def\:{\@xifnch} \expandafter\def\:{\futurelet\@let@token\@ifnch}
70.175 \fi
70.176 \def\@testopt#1#2{%
70.177 \@ifnextchar[#{#1}{#1[#2]}}
70.178 \def\@protected@testopt#1{%
70.179 \ifx\protect\@typeset@protect
70.180 \expandafter\@testopt
70.181 \else
70.182 \@x@protect#1%
70.183 \fi}
70.184 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
70.185 #2\relax}\fi}
70.186 \long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
70.187 \else\expandafter\@gobble\fi{#1}}

```

Code from ltoutenc.dtx, adapted for use in the plain T_EX environment.

```

70.188 \def\DeclareTextCommand{%
70.189 \@dec@text@cmd\providecommand
70.190 }
70.191 \def\ProvideTextCommand{%
70.192 \@dec@text@cmd\providecommand
70.193 }
70.194 \def\DeclareTextSymbol#1#2#3{%
70.195 \@dec@text@cmd\chardef#1{#2}#3\relax
70.196 }
70.197 \def\@dec@text@cmd#1#2#3{%
70.198 \expandafter\def\expandafter#2%
70.199 \expandafter{%
70.200 \csname#3-cmd\expandafter\endcsname
70.201 \expandafter#2%
70.202 \csname#3-string#2\endcsname
70.203 }%
70.204 % \let\@ifdefinable\@rc@ifdefinable
70.205 \expandafter#1\csname#3-string#2\endcsname
70.206 }
70.207 \def\@current@cmd#1{%
70.208 \ifx\protect\@typeset@protect\else
70.209 \noexpand#1\expandafter\@gobble
70.210 \fi
70.211 }
70.212 \def\@changed@cmd#1#2{%
70.213 \ifx\protect\@typeset@protect
70.214 \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
70.215 \expandafter\ifx\csname ?\string#1\endcsname\relax
70.216 \expandafter\def\csname ?\string#1\endcsname{%
70.217 \@changed@x@err{#1}%
70.218 }%
70.219 \fi
70.220 \global\expandafter\let
70.221 \csname\cf@encoding\string#1\expandafter\endcsname
70.222 \csname ?\string#1\endcsname
70.223 \fi
70.224 \csname\cf@encoding\string#1%

```

```

70.225         \expandafter\endcsname
70.226     \else
70.227         \noexpand#1%
70.228     \fi
70.229 }
70.230 \def\@changed@x@err#1{%
70.231     \errhelp{Your command will be ignored, type <return> to proceed}%
70.232     \errmessage{Command \protect#1 undefined in encoding \cf@encoding}}
70.233 \def\DeclareTextCommandDefault#1{%
70.234     \DeclareTextCommand#1?%
70.235 }
70.236 \def\ProvideTextCommandDefault#1{%
70.237     \ProvideTextCommand#1?%
70.238 }
70.239 \expandafter\let\csname OT1-cmd\endcsname\@current@cmd
70.240 \expandafter\let\csname?-cmd\endcsname\@changed@cmd
70.241 \def\DeclareTextAccent#1#2#3{%
70.242     \DeclareTextCommand#1{#2}[1]{\accent#3 #1}
70.243 }
70.244 \def\DeclareTextCompositeCommand#1#2#3#4{%
70.245     \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
70.246     \edef\reserved@b{\string##1}%
70.247     \edef\reserved@c{%
70.248         \expandafter\@strip@args\meaning\reserved@a:-\@strip@args}%
70.249     \ifx\reserved@b\reserved@c
70.250         \expandafter\expandafter\expandafter\ifx
70.251             \expandafter\@car\reserved@a\relax\relax\@nil
70.252             \@text@composite
70.253         \else
70.254             \edef\reserved@b##1{%
70.255                 \def\expandafter\noexpand
70.256                     \csname#2\string#1\endcsname###1{%
70.257                     \noexpand\@text@composite
70.258                         \expandafter\noexpand\csname#2\string#1\endcsname
70.259                         ###1\noexpand\@empty\noexpand\@text@composite
70.260                         {##1}%
70.261                     }%
70.262                 }%
70.263             \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
70.264         \fi
70.265         \expandafter\def\csname\expandafter\string\csname
70.266             #2\endcsname\string#1-\string#3\endcsname{#4}
70.267     \else
70.268         \errhelp{Your command will be ignored, type <return> to proceed}%
70.269         \errmessage{\string\DeclareTextCompositeCommand\space used on
70.270             inappropriate command \protect#1}
70.271     \fi
70.272 }
70.273 \def\@text@composite#1#2#3\@text@composite{%
70.274     \expandafter\@text@composite@x
70.275     \csname\string#1-\string#2\endcsname
70.276 }
70.277 \def\@text@composite@x#1#2{%
70.278     \ifx#1\relax
70.279         #2%
70.280     \else
70.281         #1%
70.282     \fi
70.283 }
70.284 %
70.285 \def\@strip@args#1:#2-#3\@strip@args{#2}
70.286 \def\DeclareTextComposite#1#2#3#4{%

```

```

70.287 \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
70.288 \bgroup
70.289 \lccode'\@=#4%
70.290 \lowercase{%
70.291 \egroup
70.292 \reserved@a \@%
70.293 }%
70.294 }
70.295 %
70.296 \def\UseTextSymbol#1#2{%
70.297 % \let\@curr@enc\cf@encoding
70.298 % \@use@text@encoding{#1}%
70.299 #2%
70.300 % \@use@text@encoding\@curr@enc
70.301 }
70.302 \def\UseTextAccent#1#2#3{%
70.303 % \let\@curr@enc\cf@encoding
70.304 % \@use@text@encoding{#1}%
70.305 % #2{\@use@text@encoding\@curr@enc\selectfont#3}%
70.306 % \@use@text@encoding\@curr@enc
70.307 }
70.308 \def\@use@text@encoding#1{%
70.309 % \edef\f@encoding{#1}%
70.310 % \xdef\font@name{%
70.311 % \csname\curr@fontshape/\f@size\endcsname
70.312 % }%
70.313 % \pickup@font
70.314 % \font@name
70.315 % \@@enc@update
70.316 }
70.317 \def\DeclareTextSymbolDefault#1#2{%
70.318 \DeclareTextCommandDefault#1{\UseTextSymbol{#2}{#1}}%
70.319 }
70.320 \def\DeclareTextAccentDefault#1#2{%
70.321 \DeclareTextCommandDefault#1{\UseTextAccent{#2}{#1}}%
70.322 }
70.323 \def\cf@encoding{OT1}

```

Currently we only use the L^AT_EX_{2 ϵ} method for accents for those that are known to be made active in *some* language definition file.

```

70.324 \DeclareTextAccent{"}{OT1}{127}
70.325 \DeclareTextAccent{'}{OT1}{19}
70.326 \DeclareTextAccent{^}{OT1}{94}
70.327 \DeclareTextAccent{'}{OT1}{18}
70.328 \DeclareTextAccent{~}{OT1}{126}

```

The following control sequences are used in `babel.def` but are not defined for PLAIN T_EX.

```

70.329 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
70.330 \DeclareTextSymbol{\textquotedblright}{OT1}{'"}
70.331 \DeclareTextSymbol{\textquoteleft}{OT1}{'`'}
70.332 \DeclareTextSymbol{\textquoteright}{OT1}{'\''}
70.333 \DeclareTextSymbol{\i}{OT1}{16}
70.334 \DeclareTextSymbol{\ss}{OT1}{25}

```

For a couple of languages we need the L^AT_EX-control sequence `\scriptsize` to be available. Because plain T_EX doesn't have such a sophisticated font mechanism as L^AT_EX has, we just `\let` it to `\sevenrm`.

```

70.335 \ifx\scriptsize\undefined
70.336 \let\scriptsize\sevenrm
70.337 \fi
70.338 \code>

```

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols		
\- ... 20.123 , 35.110 , 37.144 , 42.57 , 45.92 , 47.63 , 52.177 , 55.242	\@opargbegintheorem ... 48.147 , 63.826	\az 48.374
\@az 48.493	\@outputdblcol . 63.597	\az@ 48.376
\@az@string ... 48.457	\@part ... 48.192 , 63.467	\Azc 48.434 , 233
\@azc 48.440	\@roman 63.326	\azc 48.433
\@enc@update 52.365 , 55.430	\@schapter ... 63.839	\azc@ 48.435
\@selectlanguage 63.267	\@secntformat .. 48.96	\azc@@ 48.436
\@vpageref 12.1342	\@sect 48.99 , 63.510	\Azp 48.421 , 232
\@Alph 63.102	\@ssect 48.133	\azp 48.420
\@Alph@bul 58.254	\@tableofcontents 63.381	\azp@ 48.422
\@Alphfinal 63.102	\@testdef 12.1225	\Azr 48.406 , 232
\@Roman 63.326	\@textcolor 63.782	\azr 48.405
\@acute 37.77	\@tilde 40.52	\azr@ 48.407
\@alph 63.102	\@torl 63.246	\azr@@ 48.408
\@alph@bul 58.273	\@trema 20.102	\azr@@@ 48.409
\@arabic 63.326	\@umlaut .. 37.77 , 40.52	
\@az 48.377	\@verbatim 63.946	B
\@azc 48.439	\@xnthm 63.826	\babel@beginsave 12.800
\@azp 48.423		\babel@save .. 12.803 , 13
\@azr 48.410	A	\babel@savecnt . 12.800
\@begintheorem . 48.147	\AbsoluteFromGregorian 63.1157	\babel@savevariable 12.812 , 13
\@bibitem 12.1277	\AbsoluteFromHebrew 63.1321	\bahasahyphenmins 67.69
\@biblabel 63.895	\Acite 48.432 , 233	\bahasamhyphenmins 68.69
\@brackets 63.315	\acite 48.431	\basquehyphenmins 40.49
\@caption . 48.81 , 63.552	\active@prefix . 12.584	\bbl@activate 12 , 12.596
\@chapter .. 48.212 , 48.286 , 63.839	\addialect ... 11 , 12.57	\bbl@add@list .. 12.768
\@cite 63.895	\addlanguage .. 11 , 12.45	\bbl@add@special 12.402 , 13
\@citex 12.1249	\addto 12.825 , 13	\bbl@afterelse . 12.423
\@dottedtocline 63.430	\afrikaanshyphenmins 20.101	\bbl@afterfi ... 12.423
\@ensure@L 63.308	\aliasshorthand 6 , 12.638	\bbl@bibcite ... 12.1268
\@ensure@R 63.308	\allowhyphens 12.837 , 13	\bbl@cite@choice 12.1270
\@fromrml 63.255	\Alph 63.102	\bbl@clear@ttrbts 12.789
\@grave 37.77	\alph 63.102	\bbl@deactivate 12 , 12.602
\@hebrew 63.92	\Alphfinal 63.102	\bbl@declare@ttribute 12 , 12.750
\@hebrew@numeral 63.141	\anw@false 28.84	\bbl@disc 12.849
\@latin 63.317	\anw@print 28.84	\bbl@firstcs ... 12.608
\@lbibitem 63.895	\anw@true 28.84	\bbl@frenchindent . 30.512 , 50.54 , 54.58
\@listoffigures 63.381	\Aob 53.51	\bbl@frenchlabelitems 30.453
\@listoftables . 63.381	\aob 53.51	\bbl@frenchspacing 12.817 , 13
\@makecaption 48.69 , 63.949	\ap 31.84	\bbl@get@enc ... 12.356
\@makechapterhead 48.235 , 48.305	\Apageref .. 48.419 , 232	\bbl@hyph@enc .. 12.356
\@mkboth 12.1293	\apageref 48.418	\bbl@ifattributeset 12.754
\@newl@bel 12.1212	\appendix 63.129 , 63.878	\bbl@ifknown@ttrib 12.779
\@nolanerr 12.262	\arabicnorl 63.329	\bbl@language@stack 12.80
\@noopterr 12.262	\Aref 48.404 , 232	\bbl@main@language 12.247
\@nopatterns ... 12.262	\aref 48.403	
\@notshorthand . 12.651	\Asbuk ... 57.340 , 59.319	
\@number 63.317	\asbuk ... 57.347 , 59.326	
	\author 63.815	
	\AutoSpaceBeforeFDP 30.111 , 57.268 , 59.247	
	\Az 48.375 , 232	

\datebahasam	68.62	57.287 , 58.193 ,	\extrassamin	46.46
\datebasque	40.30	59.266 , 61.68	\extrasscottish	27.37
\datebrazil	35.89	\dutchhyphenmins 20.101	\extrasserbian	54.34
\datebreton	24.29		\extrasslovak	55.63
\datebulgarian	58.161	E	\extrasslovene	56.34
\datecatalan	37.31	\EEob	\extrasswedish	45.43
\datecroatian	51.29	\eeob	\extrasturkish	62.36
\dateczech	52.57	\embox	\extrasukrainian	59.177
\datedanish	42.30	\englishhyphenmins	\extrasusorbian	61.51
\datedutch	20.49	\extraswelsh	25.38
\dateenglish	21.89	environments:		
\dateesperanto	18.30	hyphenrules 6 , 12.180	F	
\dateestonian	49.34	otherlanguage	\FBprocess@options	30.779
\datefinnish	47.29	\FBtextellipsis	30.585
\datefrench	30.195	otherlanguage*	\FDP@thinspace	57.268 , 59.247
\dategerman	22.46	\FDPoff	57.272 , 59.251
\dategreek	28.64	thebibliography	\FDPon	57.272 , 59.251
\datehebrew	63.77	\fg	30.158
\dateicelandic	43.48	\Eob	\finishhyphenmins	47.66
\dateinterlingua	19.30	\eob	\flq	12.957
\dateirish	26.30	\Esper	\flqq	12.967
\dateitalian	31.29	\esper	\fnum@figure	48.63
\datelang	12	\estonianhyphenmins	\fnum@table	48.63
\datelatin	32.39	\footnoterule	63.634
\datemagyar	48.37	\et@gentilde	\foreign@language	12.157
\datengerman	23.35	\et@newtilde	\FOREIGNLANGUAGE	12.1371
\datenorsk	44.60	\extrasafrikaans	\foreignlanguage	6 , 12.148
\datepolish	53.29	\extrasalbanian	\FormatDate	63.998
\dateportuges	35.57	\extrasaustrian	\FormatForEnglish	63.1075
\dateromanian	41.30	\extrabahasa	\FormatForHebrew	63.1050
\daterussian	57.185	\extrabahasam	\fprimo)	30.249
\datesamin	46.30	\extrabasque	\frenchbsetup	30.628
\datescottish	27.29	\extrabreton	\FrenchLayout	30.566
\datesdmy	45.39	\extrabulgarian	\fromhebrew	63.90
\dateserbian	54.29	\extracatalan	\frq	12.957
\dateslovak	55.57	\extracroatian	\frqq	12.967
\dateslovene	56.29	\extrasczech	\full	363
\dateswedish	45.29	\extrasdanish	\fup	30.202
\datesymd	45.36	\extrasdutch		
\dateturkish	62.30	\extrasenglish	G	
\dateukrainian	59.162	\extrasesperanto	\gim	63.200
\datewelsh	25.31	\extrasestonian	\gim@nomil	63.151
\DaysInHebrewYear	63.1264	\extrasfinnish	\gim@print	63.193
\DecimalMathComma	30.311	\extrasfrench	\glq	12.933
\decimalsep	43.106	\extrasgerman	\glqq	12.945
\declare@shorthand	12 , 12.610	\extragreek	\gr@c@greek	28.71
\defineshorthand	6 , 12.636	\extrashebrew	\gr@i@ll@value	28.82
\degres	30.289	\extrasicelandic	\gr@month	28.64
\dieresia	40.50	\extrasinterlingua	\gr@num@i	28.139
\dieresis	37.74	\extrasirish	\gr@num@ii	28.139
\DisableNikud	64.243	\extrasitalian	\gr@num@iii	28.139
\DJ	12.907	\extraslang	\gr@num@iv	28.148
\dj	12.907	\gr@num@v	28.148
\doc@style	12.1023	\extralatin	\gr@num@vi	28.148
\dq	22.70 , 23.59 ,	\extraslorsorbian	\gradur	43.193
	43.54 , 44.75 ,	\extrasmagyar	\greek@Alph	28.115
	45.51 , 52.70 ,	\extrasmaustrian	\greek@alph	28.115
	53.112 , 55.83 ,	\extrasnagerman	\greek@amp	28.135
		\extrasnorsk	\greek@Roman	28.125
		\extrasnynorsk		
		\extraspkish		
		\extrasportuges		
		\extrasromanian		
		\extrarussian		

U				V	
<code>\ukrainian@sh@:</code>	59.210	<code>\unsethebrew</code>	63.84	<code>\v</code>	52.76 , 55.147
<code>\ukrainian@sh@;</code>	59.210	<code>\unsetrllanguage</code>	63.281	<code>\verbatim@font</code>	57.202 , 59.181
<code>\ukrainian@sh@?</code>	59.210	<code>\up</code>	30.202 , 37.194	<code>\vrefpagenum</code>	12.1342
<code>\ukrainian@sh@@</code>	59.210	<code>\upp</code>	43.155		
<code>\umlauthigh</code>	12.977	<code>\upp@size</code>	43.155		
<code>\umlautlow</code>	12.977	<code>\user@group</code>	12.626		
<code>\underline</code>	63.821	<code>\usesorthands</code>	6 , 12.629		
<code>\unit</code>	31.84	<code>\uv</code>	52.159 , 55.224		
				W	
				<code>\welshhyphenmins</code>	25.6

Change History

3.0	
<code>\c1qq</code> : Added <code>\leavevmode</code> to allow an opening quote at the beginning of a paragraph	260, 282
albanian-1.0a	
General: Started first version of the file	251
albanian-1.0b	
General: A number of corrections in the translations from Adi Zaimi	251
albanian-1.0c	
General: Small documentation fix	251
babel 2.0a	
General: Added text about <code>german.sty</code>	1
babel 2.0b	
General: Changed order of code to prevent plain TeX from seeing all of it	1
babel 2.1	
General: Modified user interface, <code>\langTeX</code> no longer necessary	1
babel 2.1a	
General: Incorporated Nico's comments	1
babel 2.1b	
General: rename <code>\language</code> to <code>\current@language</code>	1
babel 2.1c	
General: abstract for report fixed, missing <code>}</code> , found by Nicolas Brouard	1
babel 2.1d	
General: Missing right brace in definition of abstract environment, found by Werenfried Spit	1
babel 2.1e	
General: Incorporated more comments from Nico	1
babel 2.2	
General: Renamed <code>\newlanguage</code> to <code>\addlanguage</code>	1
babel 2.2a	
General: Modified the documentation somewhat	1
babel 3.0	
General: Moved part of the code to <code>hyphen.doc</code> in preparation for TeX 3.0	1
babel 3.0a	
General: Updated comments in various places	1
<code>\iflanguage</code> : Added <code>\@bsphack</code> and <code>\@esphack</code>	21
<code>\selectlanguage</code> : Added <code>\@bsphack</code> and <code>\@esphack</code>	23
Replaced <code>\gdef</code> with <code>\def</code>	23
babel 3.0b	
General: Removed some problems in change log	1
babel 3.0c	
General: Renamed <code>babel.sty</code> and <code>latexhax.sty</code> to <code>.com</code>	1
<code>\iflanguage</code> : Added comment character after <code>#2</code>	21
<code>\selectlanguage</code> : Made <code>\selectlanguage</code> robust	21
babel 3.0d	
<code>\@noopterr</code> : Added a percent sign to remove unwanted white space	28
General: Removed use of <code>\@ifundefined</code>	18
<code>\doc@style</code> : Removed use of <code>\@ifundefined</code>	51
<code>\iflanguage</code> : Removed space hacks and use of <code>\@ifundefined</code>	21
Removed superfluous <code>\expandafter</code>	21
<code>\process@language</code> : Added the collection of pattern names.	29
Reinserted <code>\expandafter</code>	29
Removed superfluous <code>\expandafter</code>	29
<code>\selectlanguage</code> : Removed space hacks and use of <code>\@ifundefined</code>	23
Removed superfluous <code>\expandafter</code>	23
babel 3.1	
General: Added the support for active characters and for extending a macro	1
Removed definition of <code>\if@restonecol</code>	51
Removed the need for <code>latexhax</code>	1
<code>\addto</code> : Added macro	45

\readconfigfile: Removed the extra if control sequence	32
Removed use of \toks0	31
\selectlanguage: \originalTeX should only be executed once	23
babel 3.2	
General: Some Changes by br	1
\adddialect: Added \relax	21
\addlanguage: Added a %, removed by	20
\babel@beginsave: Added macro	44
\babel@save: Added macro	44
\babel@savecnt: Added macro	44
\babel@savevariable: Added macro	45
\bbl@add@special: Added macro	32
\bbl@remove@special: Added macro	33
\iflanguage: Rephrased \ifnum test	21
\selectlanguage: Modified to allow arguments that start with an escape character	21
babel 3.2a	
General: Fixups of the code and documentation	1
\originalTeX: Set \originalTeX to \empty, because it should be expandable.	28
\readconfigfile: Free macro space for \process@language	32
\selectlanguage: Added \relax as first command to stop an expansion if	
\protect is empty	23
Added three \expandafters to save macro space for \originalTeX	23
Moved definition of \originalTeX before \extras<lang>	23
Set \originalTeX to \empty, because it should be expandable.	23
Simplified the modification to allow the use in a \write command	21
babel 3.2b	
\allowhyphens: Moved macro from language definition files	45
\save@sf@q: Moved macro from language definition files	46
\set@low@box: Moved macro from language definition files	46
babel 3.2c	
\babel@save: missing backslash led to errors when executing \originalTeX ..	44
babel 3.2d	
\babel@save: saving in \babel@i and restoring from \@babel@i doesn't work	
very well...	44
babel 3.2e	
General: Added slovak	64
babel 3.3	
General: \headpagename should be \pagename	54
Added catalan and galician	64
Added turkish	64
Included driver file, and prepared for distribution	1
babel 3.4	
General: Added bahasa	64
Added language definition file for bahasa	1
Updated for L ^A T _E X 2 _ε	1
\addto: Changed to use toks register	45
babel 3.4b	
General: Added a small driver to be able to process just this file	1
Use the ltxdoc class instead of article	63
babel 3.4c	
General: lhyphen.cfg has become lthyphen.cfg	15
babel 3.4e	
\@nolanerr: Use \PackageError in L ^A T _E X 2 _ε mode	28
\@nopatterns: Macro added	28
\process@language: Added code to detect assignments to left- and righthyphen-	
min in the patternfile.	30
\ProvidesLanguage: Redid the identification code, provided dummy definition of	
\ProvidesFile for plain T _E X	14
babel 3.4g	
\@testdef: Moved the \def inside the macrocode environment	56

babel 3.5a	
\@noopterr: Added \@activated to log active characters	28
General: Added a system shorthand for the right quote	41
Added breton, irish, scottish	64
Changed extension of language definition files to ldf	15
Provided common code to handle the active double quote	1
Replaced 16 system shorthands to deal with hex numbers by one	41
\bb1@activate: Added macro	38
\bb1@deactivate: Added macro	38
\bb1@main@language: Macro added	27
\bb1@pr@ms: Added macro	41
\bb1@set@language: write the language change to the auxiliary files	23
\initiate@active@char: Added a check for right quote and adapt \@pr@ms if necessary	34
Added macro	33
\main@language: Macro added	27
babel 3.5b	
General: Added brazilian as alternative for brazil	16
Added lsorbian, usorbian	64
Added the estonian option	16
lthyphen.cfg has become hyphen.cfg	15
\initiate@active@char: Renamed macro	33
\pageref: Made \@ref and \@pageref robust (PR1353)	57
\process@language: Added optional reading of file with hyphenation exceptions	30
\process@line: added macro	29
\process@synonym: added macro	29
\readconfigfile: Now add a \@space and a / character	31
\selectlanguage: Added an extra level of expansion to separate the switching mechanism from writing to aux files	22
Added default setting of hyphenmin parameters	24
Changed the name of the internal macro to \@selectlanguage	22
Separated the setting of the hyphenmin values	24
Store the name of the current language in a control sequence instead of passing the whole macro construct to strip the escape character in the argument of \@selectlanguage	21
babel 3.5c	
\@noopterr: Added missing closing brace	28
General: Changed the order of including the language files somewhat (PR1652)	64
corrected a few typos (PR1652)	1
Repaired a typo (itlaic, PR1652)	46
babel 3.5d	
General: Added british as an alternative for english with a preference for british hyphenation	16
Added options to influence behaviour of active acute and grave accents	17
Load french.ldf when it is found instead of frenchb.ldf	16
Load language definition files <i>after</i> the check for the hyphenation patterns	15
Merged glyphs.dtx into this file	1
\active@prefix: \@protected@cmd has vanished from ltoutenc.dtx	38
\declare@shorthand: Make a ‘note’ when a shorthand with an argument is defined.	39
\foreignlanguage: Macro added	25
\initiate@active@char: Skip the language-level active char with argument if no shorthands with arguments were defined	36
Skip the user-level active char with argument if no shorthands with arguments were defined	36
\loadlocalcfg: Added macro	62
\pageref: use a different control sequence while making \@ref and \@pageref robust	57
otherlanguage: environment added	24
babel 3.5e	
otherlanguage: changed name	24

babel 3.5f	
\@bibitem: Now use \bbl@redefine	58
\@citex: Now use \bbl@redefine	57
\@testdef: Complete rewrite of this macro as the same character ended up with different category codes in the labels that are being compared. Now use \meaning	56
Now use \bbl@redefine	56
Use \strip@prefix only on \bbl@tempa when it is not \relax	56
General: Added a system shorthand for the left quote	41
Added the greek option	16
No need to reset the category code of the tilde as \initiate@active@char now correctly deals with active characters	41
Now use the file frenchb.ldf from Daniel Flipo for french support	16
repaired a typo	1
replaced \tmp, \bbl@tmp and \bbl@temp with \bbl@tempa	1
\aliasshorthand: New command	39
\bbl@disc: Macro moved from language definition files	46
\bbl@redefine: Macro added	55
\bbl@redefineroobust: Define *foo instead of \foo	55
Macro added	55
\bbl@set@language: Now also define \language name at this level	23
\bbl@test@token: macro added	34
\bibcite: Now use \bbl@redefine	57
\DJ: New definition of \dj, see PR 2058	47
\frq: corrected spelling of \quilsingl...	49
now use \textormath in these definitions	49
\frqq: corrected spelling of \quillemot...	49
now use \textormath in these definitions	49
\initiate@active@char: Deal correctly with already active characters, provide top level expansion and define all lower level expansion macros outside of the \else branch.	34
restore the \lccode of the tie	35
Restore the category code of a shorthand char at end of package	35
store the \lccode of the tie before changing it	35
use \peek@token to check whether it is safe to proceed	36, 37
\lower@umlaut: Added a \allowhyphens	50
removed \allowhyphens	50
\newlabel: Now use \bbl@redefine	56
\nocite: Now use \bbl@redefine	57
\pageref: Now use \bbl@redefineroobust	57
redefine *ref if it exists instead of \ref	57
redefine \setref instead of \ref and \pageref in L ^A T _E X 2 _ε .	57
Reverse the previous change as it inhibits the use of active characters in labels	57
\peek@token: macro added	33
\process@language: Use \empty instead of \@empty as the latter is unknown in plain	30
\ProvidesLanguage: Need to temporarily change the definition of \ProvidesFile for December 1995 release	14
Store version in \fileversion	14
\readconfigfile: Moved the fiddling with \dump to bbplain.dtx as it is no longer needed for L ^A T _E X	32
\selectlanguage: Added a missing percent character	21
Moved check for escape character one level down in the expansion	21
otherlanguage*: environment added	24
babel 3.5g	
General: Added definition of \Babel	64
Added greek	64
Added option afrikaans	15
Removed the use of \patterns@loaded altogether	29
replaced \undefined with \@undefined to be consistent with L ^A T _E X	1
\ifthenelse: Redefinition of \ifthenelse added to circumvent problems with \pageref in the argument of \isodd	59

\initiate@active@char: Top level expansion of \normal@char char where char is already active, should be the expansion of the active character, not the active character itself as this causes an endless loop	34
\nfss@catcodes: Need to add the double quote and acute characters to \nfss@catcodes to prevent problems when reading in .fd files	61
\process@line: Simplified code, removing \bbl@eq@	29
\ProvidesLanguage: Save a few csnames; use \bbl@tempa instead of \ProvidesFile and store message in \toks8	14
babel 3.6a	
\@@vpageref: Redefinition of \@@vpageref added to circumvent problems with active : in the argument of \vref when varioref is used	60
General: Added welsh	64
Removed \babel@core@loaded, no longer needed with the advent of \LdfInit	19
\ldf@finish: Macro added	27
\ldf@quit: Macro added	27
\LdfInit: Macro added	26
\main@language: \main@language now also sets \language and \l@language for use by other packages in the preamble of a document	27
\selectlanguage: Check for the existence of \date... instead of \l@...	23
babel 3.6b	
\addto: Also check if control sequence expands to \relax	45
babel 3.6c	
General: When \LdfInit is undefined we need to load babel.def from babel.sty	15
\bbl@main@language: When hyphen.cfg is not loaded in the format \l@english might not be defined; assume english is language 0	28
babel 3.6d	
\foreign@language: Added \relax to prevent disappearance of the first token after this command.	25
New macro	25
set the language shorthands to ‘none’ before switching on the extras	25
\foreignlanguage: Introduced \foreign@language	25
\selectlanguage: set the language shorthands to ‘none’ before switching on the extras	23
otherlanguage*: Introduced \foreign@language	24
babel 3.6e	
General: Added option frenchb an alias for francais	16
Added options UKenglish and USenglish	17
babel 3.6f	
General: Added option KeepShorthandsActive	17
\bbl@redefine@long: Macro added	55
\ifthenelse: \ifthenelse needs to be long	60
\initiate@active@char: Made restoring of the category code of shorthand characters optional	35
babel 3.6h	
\readconfigfile: Added a couple of \expandafters to copy the contents of \toks8 into \everyjob instead of the reference	32
babel 3.6i	
\@newl@bel: Now redefine \@newl@bel instead of \@lbibitem and \newlabel	56
\@testdef: \@safe@activesfalse is now part of the label definition	56
Make sure that shorthands don’t get expanded at the wrong moment.	56
General: Added basque	64
Added default option	17
Added the Basque option	15
Added the ukrainian option	17
Added the possibility to have a bblopts.cfg file with option declarations.	17
\bbl@afterfi: Made \bbl@afterelse and \bbl@afterfi \long	33
\bbl@test@token: renamed \bbl@token to \bbl@test@token to prevent a clash with ArabTeX	34
\declare@shorthand: Make it possible to distinguish the constructed control sequences for the case with argument	39
\ifthenelse: Now reset the @safe@actives switch inside the 2nd and 3rd arguments of \ifthenelse	60

\initiate@active@char: Make shorthands active during .aux file processing .	35
Remove the use of \peek@token again	37
Remove the use of \peek@token again and make the \...active@arg... com- mands \long	36
\latinencoding: Macro added, moved from .ldf files	19
\latintext: Macro added, moved from .ldf files	19
\markright: Added redefinition of \mark... commands	58
\peek@token: Renamed \test@token to \bbl@test@token to prevent a clash with ArabTeX	33
\system@group: Have a user group called ‘user’ by default	39
\textlatin: Macro added, moved from .ldf files	19
babel 3.6k	
\latinencoding: Use T1 encoding when it is a known encoding	19
babel 3.6l	
General: Don’t load babel.def now, but rather define \LdfInit temporarily in order to load babel.def at the right time, preventing problems with the temporary definition of \bbl@redefine	15
babel 3.6m	
\latinencoding: Can’t use \ifpackageloaded need to parse \@filelist . . .	19
babel 3.6n	
\latinencoding: Added a check for ‘manual’ selection of T1 encoding, without loading fontenc	19
moved checking for fontenc right to the top of babel.sty	19
babel 3.6o	
General: Moved the rest of the font encoding related definitions to their original place	19
babel 3.6p	
General: Added the ngerman and naustrian options	16
babel 3.6q	
\latinencoding: Better solution then parsing \@filelist, use \ifl@aded . .	19
babel 3.6r	
General: We do need to load babel.def right now as \ProvidesLanguage needs to be defined before the .ldf files are read and the reason for for 3.6l has been removed	15
babel 3.6s	
\bibcite: Need to determine ‘online’ which definition of \bibcite is needed .	57
babel 3.6u	
\latinencoding: Moved this code to babel.def	19
babel 3.6v	
\bbl@bibcite: Macro \bbl@bibcite added	58
\bbl@cite@choice: Macro \bbl@cite@choice added	58
\bibcite: Also check for cite it can’t handle \@safe@activesfalse in its second argument	57
babel 3.7a	
\newl@bel: Call \@safe@activestruer directly	56
\@testdef: Removed \@safe@activesfalse from the label definition	56
General: Added icelandic	64
Added the hebrew option	16
Added the icelandic option	16
Added the polutonikogreek option	16
No longer define the control sequence \KeepShorthandsActive	17
Now need packages t1enc and supertabular to be loaded; the documentation for icelandic needs its .ldf file to be present	63
\allowhyphens: Make \allowhyphens a no-op for T1 fontencoding	45
\bbl@switch@sh: Added command	40
\foreignlanguage: Added executing \originalTeX	25
\grq: Make the definition of \grq dependent on the font encoding	48
\grqq: Make the definition of \grqq dependent on the font encoding	48
\iflanguage: Now evaluate the \ifnum test <i>after</i> the \fi from the \ifx test and use \@firstoftwo and \@secondoftwo	21
\initiate@active@char: Commented out peek@token and \test@token as short- hands are made expandable again	33

Use <code>\@ifpackagewith</code> to determine whether shorthand characters need to remain active	35
<code>\LaTeX</code> : Make <code>T_EX</code> and <code>L^AT_EX</code> logos encoding-independent	59
<code>\process@language</code> : Read pattern files in a group	30
<code>\ProvidesLanguage</code> : Added macro to prevent problems with unexpected <code>\ProvidesFile</code> in plain formats because of <code>babel</code> .	14
Removed superfluous braces	15
<code>\shorthandoff</code> : Added command	40
<code>\shorthandon</code> : Added command	40
babel 3.7b	
General: Added Latin	64
Added the <code>latin</code> option	16
<code>\iflanguage</code> : Slight enhancement: added braces around first argument of <code>\bbl@afterfi</code>	21
babel 3.7c	
General: Added an error message for when no language option was specified	17
Added hebrew and serbian	64
Added support for language attributes	42
Added ukrainian	64
define <code>\adddialect</code> before loading <code>plain.def</code> here	18
No longer use a redefinition of an internal macro, just check <code>\bbl@main@language</code> and load <code>babel.def</code>	17
Removed redefinition of <code>\@roman</code> and <code>\@Roman</code>	59
set the correct language attribute for polutoniko greek	16
<code>\initiate@active@char</code> : Only execute <code>\initiate@active@char</code> once for each character	34
<code>\markright</code> : Avoid expanding the arguments by storing them in token registers	58
Removed the use of <code>\head@lang</code> (PR 2990)	58
<code>\process@language</code> : Added the execution of the contents of <code>\toks@</code>	30
Also store <code>\language</code> for possible later use in <code>\process@synonym</code>	29
need to set hyphenmin values globally	30
Only set hyphenmin values when the pattern file changed them	30
Set <code>\lefthyphenmin</code> to <code>\m@ne</code> <i>inside</i> the group; explicitly set the hyphenmin parameters for language 0	30
<code>\process@line</code> : added an extra argument in order to prevent a trailing space from becoming part of the control sequence when defining a synonym (PR 2851)	29
<code>\process@synonym</code> : Now also store hyphenmin parameters for language synonyms	29
Use a token register to temporarily store a command to set hyphenmin parameters for the synonym which is defined <i>before</i> the first pattern file is processed	29
babel 3.7d	
General: Fixed a few typos in <code>\changes</code> entries which made typesetting the code impossible	1
<code>\initiate@active@char</code> : Make sure the active character doesn't get expanded more than once by the <code>\edef</code> by adding <code>\expandafter\strip@prefix\meaning</code>	34
babel 3.7e	
General: Added missing hebrew files	64
<code>\bbl@clear@ttribs</code> : When <code>\bbl@attributes</code> is undefined this should be a no-op	44
<code>\initiate@active@char</code> : pass the argument on with braces in order to prevent it from breaking up	36, 37
previous change was rubbish; use <code>\let</code> instead of <code>\edef</code>	34
<code>hyphenrules</code> : Added environment <code>hyphenrules</code>	25
babel 3.7f	
General: Added bulgarian	64
Added <code>samin</code>	64
Added the <code>bulgarian</code> option	16
Added the <code>samin</code> option	16
<code>\bbl@get@enc</code> : Added macro	31
<code>\bbl@ifattributeset</code> : macro added	43

\bb1@prim@s: Need to redefine \prim@s as well as plain T _E X's definition uses	
\next	41
\bb1@set@language: Macro \bb1@set@language introduced	23
\ifthenelse: \pageref needs to have its babel definition reinstated in the second and third arguments	60
\initiate@active@char: Added an extra shorthand combination on user level to catch an interfering \protect	37
Insert a check for math mode in the definition of \normal@char'	35
Introduced an extra level of expansion in the definition of an active caret ..	35
Make an exception for the active caret which needs an extra level of expansion	36
remove the braces again	36, 37
The redefinition needs to take place one level higher, \prim@s needs to be redefined.	34
\process@language: Allow for the encoding to be used as part of the language name	30
\providehyphenmins: added macro	26
\save@sf@q: PR3119, don't start a paragraph in a local group	46
\selectlanguage: Use \aftergroup to keep the language grouping correct in auxiliary files PR3091	22
babel 3.7g	
\@citex: The shorthands need to be deactivated for the second argument of \@citex only.	57
General: Added option acadian	15
Added option canadian	16
Added option canadien	16
\initiate@active@char: use \textormath to get rid of the \fi (PR 3266) ...	35
babel 3.7h	
General: Added a number of missing comment characters which caused spurious white space	1
babel 3.7j	
General: <i>only</i> load frenchb.1df	16
\bb1@pop@language: Introduce the language stack mechanism	22
Now use the language stack mechanism	23
\FOREIGNLANGUAGE: Define \FOREIGNLANGUAGE unconditionally	61
\substitutefontfamily: create file with lowercase name	18
\textlatin: added \leavevmode to prevent a paragraph starting <i>inside</i> the group	19
otherlanguage: rely on \selectlanguage to keep track of the nesting	24
\usesshorthands: The change from 11/112001 was incomplete	39
When T _E X has seen a character its category code is fixed; need to use a 'stand-in' for the call of \bb1@activate	39
babel 3.7k	
\textlatin: Use \DeclareTextFontCommand	19
babel 3.7m	
\@mkboth: added \bb1@restore@actives to the mark	59
\@nooperr: Macro added	28
General: Added the interlingua option	16
\bb1@pop@language: Removed the superfluous empty definition of \bb1@pop@language	23
\bb1@restore@actives: New macro added	38
\markright: added \bb1@restore@actives to the mark	58
\selectlanguage: Check for the existence of both \l@... and \date...	23
babel 3.7o	
\@active@prefix: Added handling of the situation where \protect is set to \@unexpandable@protect	38
\ldf@finish: Also restore the category code of the equals sign	27
\ldf@quit: Also restore the category code of the equals sign	27
\LdfInit: make sure the equals sign has its default category code	26
\vreftpagenum: Added redefinition of \vreftpagenum which deals with ranges of pages	60
babel 3.8a	
General: Added interlingua	64
Also load package url	63

babel 3.8b	
\bb1@sh@select: Added command	37
\declare@shorthand: We need to support shorthands with and without argument in different groups; added the name of the group to the storage macro	39
\frq: Made \flq and \frq fontencoding dependent	49
\frqq: Made \flqq and \frqq fontencoding dependent	49
\grq: Made \glq fontencoding dependent as well	48
\grqq: Made \grqq fontencoding dependent as well	48
\hhline: added \string to prevent unwanted expansion of the colon	61
\initiate@active@char: Now use \bb1@sh@select	36, 37
babel 3.8c	
\mkboth: No need to add <i>anything</i> to an empty mark, prevented this by checking the contents of the arguments	59
General: Added option australian	15
Added the newzealand option	16
\markright: No need to add <i>anything</i> to an empty mark; prevented this by checking the contents of the argument	58
babel 3.8e	
General: Many enhancements to the text by Andrew Young	1
babel 3.8f	
\mkboth: Make the definition independent of the original definition; expand \language before passing it into the token registers	59
\markright: Make the definition independent of the original definition; expand \language before passing it into the token registers	58
babel 3.8g	
\Ref: We also need to adapt \Ref which needs to be able to uppercase the first letter of the expansion of \ref	60
babel 3.8h	
General: added malay , meyaluy and bahasam for the Bahasa Malaysia support	15
Added albanian and bahasam	64
Added option albanian	15
added synonyms indonesian , indon and bahasai for the original bahasa (indone- sia) support	15
babel 3.8j	
\mkboth: Added setting of \mkboth (PR 3826)	59
\bb1@switch@sh@on: Added a group in order to protect the current lowercase code of the tilde (PR 3851)	40
\pdfstringdefDisableCommands: Inform hyperref to use shorthands at system level (PR4006)	61
hyphenrules : Also set the hyphenmin paramters to the correct value (PR3997)	25
babel 3.8l	
\bb1@main@language: Use \bb1@patterns	27
\bb1@patterns: Macro added	25
\foreign@language: use \bb1@patterns	25
\selectlanguage: Use \bb1@patterns	23
hyphenrules : Use \bb1@patterns	25
bahasa-0.9c	
General: Now use \@patterns to produce the warning	394, 396
Removed the use of \filedate and moved identification after the loading of babel.def	394, 396
bahasa-1.0b	
\captionsbahasa: Added \proofname for AMS- \LaTeX	395
\captionsbahasam: Added \proofname for AMS- \LaTeX	396
bahasa-1.0d	
General: Replaced \undefined with \@undefined and \empty with \@empty for consistency with \LaTeX	394, 396
\captionsbahasa: Replaced ‘Proof’ by ‘Bukti’ (PR2214)	395
\captionsbahasam: Replaced ‘Proof’ by ‘Bukti’ (PR2214)	396
bahasa-1.0e	
General: Moved the definition of \atcatcode right to the beginning.	394, 396
Now use \ldf@finish to wrap up	395, 397
Now use \LdfInit to perform initial checks	394, 396

\bahasahyphenmins: use \bahasahyphenmins to store the correct values	395
\bahasamhyphenmins: use \bahasamhyphenmins to store the correct values ..	397
bahasa-1.0f	
\datebahasa: Use \edef to define \today	395
\datebahasam: Use \edef to define \today	397
bahasa-1.0g	
\datebahasa: Februari should be spelled as Pebruari	395
\datebahasam: Februari should be spelled as Pebruari	397
bahasa-1.0h	
\bahasahyphenmins: Now use \providehyphenmins to provide a default value	395
\bahasamhyphenmins: Now use \providehyphenmins to provide a default value	397
\captionsbahasa: Added \glossaryname	395
\captionsbahasam: Added \glossaryname	396
bahasa-1.0i	
\captionsbahasa: Inserted translation for Glossary	395
\captionsbahasam: Inserted translation for Glossary	396
bahasa-v1.0k	
General: Make it possible that this file is loaded by variuos options	394
bahasam-0.9f	
General: A number of changes to make this specific to Bahasa Mayasia	396
bahasam-1.0k	
\captionsbahasam: Inserted changes from Awangku Merali	396
\datebahasam: Februari restored to BM spelling; see Collins Kamus Dwibahasa	
2005	397
bahasam-v1.0j	
General: Make it possible that this file is loaded by variuos options	396
bahasa 1.0f	
\datebahasa: use \def instead of \edef to save memory	395
\datebahasam: use \def instead of \edef to save memory	397
basque-1.0b	
General: Removed empty groups after guillemot characters	205
\datebasque: use \def instead of \edef	204
Use \edef to define \today to save memory	204
basque-1.0c	
\datebasque: fixed typo in April's name	204
basque-1.0d	
\noextrabasque: Deactivate shorthands ouside of Basque	204
basque-1.0e	
\basquehyphenmins: Now use \providehyphenmins to provide a default value	204
\captionsbasque: Added \glossaryname	203
basque-1.0f	
General: Changed url's for the patterns file	203
\captionsbasque: Added translation for Glossary	203
bbplain-0.1	
General: Added redefinition of \dump to add a message to \everyjob	399
bbplain-1.0c	
General: Add execution of \@begindocumenthook to \@preamblecmds	399
Added definition of \loadlocalcfg	399
Moved the \dump code here from babel.dtx	399
bbplain-1.0d	
General: Also reset category codes after loading the configuration file as \AtEndOfPackage	
is undefined in this case	399
bbplain-1.0e	
General: Added the \newcommand code	402
Provide a more complete emulation of \DeclareRobustCommand and \newcommand	
.	401
bbplain-1.0f	
General: added \@empty	399
Added \textquotedblright and \textquoteright	405
Added definition of \scriptsize	405
Consistently use \@undefined instead of \undefined	399
Use \toks8 instead of \patterns@loaded	400

bbplain-1.0g	
General: Added <code>\ss</code> and <code>\i</code>	405
bbplain-1.0h	
General: Only load the necessary parts into the format, let this file be read again by <code>babel.def</code>	399
bbplain-1.0i	
General: <code>\document</code> is not a L ^A T _E X2.09-only command; AMST _E X defines it too; now use <code>\@ztryfc</code> to detect L ^A T _E X2.09	399
bbplain-1.0j	
General: <code>\@begindocumenthook</code> might already be defined	401
Add the definition of <code>\@begindocumenthook</code> to the L ^A T _E X2.09 format	399
bbplain-1.0k	
General: <code>\newcount</code> is an <code>\outer</code> command, can't use it inside an <code>\if</code> construct	402
missing <code>\@undefined</code> added	401
bbplain-1.0l	
General: Mixed up the definition of <code>\@tempcntb</code>	402
bbplain-1.0m	
General: Set <code>\if@files</code> to <code>\iffalse</code> only for plain T _E X	401
bbplain-1.0n	
General: Added <code>\@secondoftwo</code>	400
Added the source for the format wrapper files	398
Repaired typo and added missing <code>\endcsname</code>	401
bbplain-1.0o	
General: Added definition of <code>\in@</code>	402
bbplain-1.0p	
General: Added <code>\@ifl@aded</code> as a no-op	402
bbplain-1.0q	
General: Added <code>\@ifundefined</code>	400
bbplain-1.0r	
General: Added <code>\textquotedblleft</code> and <code>\textquoteleft</code>	405
breton-1.0	
General: First release	89
breton-1.0b	
<code>\captionsbreton</code> : Added <code>\proofname</code> for AMS-L ^A T _E X	89
<code>\noextrasbreton</code> : Use the new mechanism for dealing with active chars	90
breton-1.0c	
General: Postpone the <code>\DeclareTextCompositeCommands</code> untill <code>\AtBeginDocument</code>	91
breton-1.0e	
General: Now use <code>\ldf@finish</code> to wrap up	91
Now use <code>\LdfInit</code> to perform initial checks	89
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consis- tency with L ^A T _E X, moved the definition of <code>\atcatcode</code> right to the beginning.	89
breton-1.0f	
<code>\datebreton</code> : use <code>\def</code> instead of <code>\edef</code>	89
Use <code>\edef</code> to define <code>\today</code> to save memory	89
breton-1.0g	
<code>\noextrasbreton</code> : Deactivate shorthands outside of Breton	90
breton-1.0h	
<code>\captionsbreton</code> : Added <code>\glossaryname</code>	89
bulgarian-0.99	
General: This is a prerelease version of this file. Features needing further testing are removed.	303
bulgarian-1.0b	
<code>\extrasbulgarian</code> : Now use <code>\providehyphenmins</code> to provide a default value	310
bulgarian-1.0c	
General: Added missing closing brace	305
<code>\dq</code> : repaired typo	309
bulgarian-1.0d	
General: Change definition of <code>\th</code> only for this language	312
bulgarian-1.0e	
<code>\cdash</code> : Two occurrences of <code>\emp</code> were changed into <code>tab</code> followed by <code>emp</code>	310

catalan-1.1	
<code>\captionscatalan: \headpagename</code> should be <code>\pagename</code>	178
catalan-2.0	
General: Removed code to load <code>latexhax.com</code>	177
<code>\captionscatalan</code> : Added some names	178
<code>\extrascatalan</code> : Macro completely rewritten	178
<code>\noextrascatalan</code> : Macro completely rewritten	178
catalan-2.0b	
General: Incorporated the changes from <code>spanish.sty</code>	177
catalan-2.1	
General: Update for L ^A T _E X 2 _ε	177
catalan-2.1d	
General: Now use <code>\@nopatterns</code> to produce the warning	178
Removed the use of <code>\filedate</code> and moved identification after the loading of	
<code>babel.def</code>	177
<code>\captionscatalan</code> : Added a few missing translations	178
<code>\textacute</code> : Renamed from <code>\acute</code> as that is a <code>\mathaccent</code>	179
catalan-2.2a	
General: All the code to deal with active characters is now in <code>babel.def</code> . . .	180
<code>\extrascatalan</code> : Handling of active characters completely rewritten	178
<code>\noextrascatalan</code> : All the code for handling active characters is now moved to	
<code>babel.def</code>	179
catalan-2.2b	
General: Changed mathmode definition of acute shorthands to expand to a single	
prime followed by the next character in the input	181
Made the activation of the grave and acute accents optional	177
<code>\captionscatalan</code> : Added <code>\proofname</code> for AMS-L ^A T _E X	178
<code>\datecatalan</code> : Month names in lowercase	178
<code>\Lgem</code> : Added support for typing the catalan “ <i>ela geminada</i> ” with the macros	
<code>\lgem</code> and <code>\Lgem</code>	182
<code>\noextrascatalan</code> : Make activating the accent characters optional	179
<code>\up</code> : Added definition of macro <code>\up</code> , which can be used to type ordinals	183
catalan-2.2c	
General: Added ’ as an axtra shorthand, removed ’n as a shorthand	181
Added shorthands for guillemets	181
cedile now produced by double quote shorthand	181
Removed the use of the tilde for catalan	177
catalan-2.2d	
<code>\captionscatalan</code> : added translation of Proof	178
Translations revised	178
catalan-2.2e	
General: Added “ as an axtra shorthand	181
Added vertical bar as argument to active acute	181
<code>\L.L</code> : Added redefinition of <code>\l</code> and <code>\L</code>	182
<code>\noextrascatalan</code> : Need to split up the <code>\@ifpackagewith</code> statements	179
Now give the apostrophe a lowercase code	178
<code>\up</code> : Now use <code>\textsuperscript</code> and make <code>\up</code> robust	183
catalan-2.2f	
General: Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for	
consistency with L ^A T _E X	177
<code>\Lgem</code> : Added a check for math mode as the use of <code>\lgem</code> and <code>\Lgem</code> in math	
mode is not sensible.	182
catalan-2.2g	
General: Moved the definition of <code>\atcatcode</code> right to the beginning.	177
Now use <code>\ldf@finish</code> to wrap up	183
Now use <code>\LdfInit</code> to perform initial checks	177
catalan-2.2h	
<code>\noextrascatalan</code> : Added some comment signs to prevent unwanted spaces in	
the output	179
catalan-2.2i	
General: Removed empty groups after guillemot characters	181
<code>\datecatalan</code> : use <code>\def</code> instead of <code>\edef</code>	178

Use <code>\edef</code> to define <code>\today</code> to save memory	178
catalan-2.2k	
General: A wrong <code>\changes</code> entry made typesetting impossible	177
catalan-2.2l	
<code>\noextrascatalan</code> : Don't forget to deactivate the shorthands	179
Make sure that the grave accent has catcode 12 <i>before</i> it is made <code>\active</code>	179
catalan-2.2m	
<code>\captionscatalan</code> : Added <code>\glossaryname</code>	178
catalan-2.2n	
<code>\catalanhyphenmins</code> : Set the hyphenation parameters both to two as required by <code>cahyph.tex</code>	178
catalan-2.2o	
<code>\L.L</code> : Postpone the redefinition of <code>\l</code> and <code>\L</code> until begin document to prevent overwriting by fontenc	182
catalan-2.2p	
<code>\captionscatalan</code> : Inserted translation for Glossary	178
changes-1.0f	
General: The hyphen char needs to appear at the beginning of the line as well.	205
croatian-1.0a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	254
croatian-1.0b	
General: Removed use of <code>\@ifundefined</code>	254
croatian-1.0c	
General: Removed some typos	254
croatian-1.1	
General: Added a warning when no hyphenation patterns were loaded.	254
Brought up-to-date with babel 3.2a	254
<code>\captionscroatian</code> : Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	254
croatian-1.2	
<code>\captionscroatian</code> : <code>\headpagename</code> should be <code>\pagename</code>	254
croatian-1.3	
General: Update for L ^A T _E X 2 _ε	254
croatian-1.3d	
<code>\captionscroatian</code> : Added a few translations	254
croatian-1.3e	
<code>\captionscroatian</code> : Added <code>\proofname</code> for AMS-L ^A T _E X	254
croatian-1.3f	
<code>\captionscroatian</code> : Added translation of Proof	254
<code>\datecroatian</code> : in croatian language, the genitive for the name of the month has to be used	254
croatian-1.3g	
General: Now use <code>\ldf@finish</code> to wrap up	255
Now use <code>\LdfInit</code> to perform initial checks	254
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consis- tency with L ^A T _E X, moved the definition of <code>\atcatcode</code> right to the beginning.	254
croatian-1.3h	
<code>\datecroatian</code> : <code>sijev{c}nja</code> should be <code>seij\v{c}nja</code> and there should be a period after the year	254
croatian-1.3i	
<code>\captionscroatian</code> : Replaced some of the translations with 'better' words	254
<code>\datecroatian</code> : use <code>\def</code> instead of <code>\edef</code>	254
Use <code>\edef</code> to define <code>\today</code> to save memory	254
croatian-1.3j	
<code>\datecroatian</code> : changed <code>\od</code> into <code>\or</code>	254
croatian-1.3k	
<code>\captionscroatian</code> : Added <code>\glossaryname</code>	254
croatian-1.3l	
<code>\captionscroatian</code> : Inserted translation for Glossary	254
czech-1.0a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	256
czech-1.0b	
General: Removed use of <code>\@ifundefined</code>	257

czech-1.1	
General: Added a warning when no hyphenation patterns were loaded.	257
Brought up-to-date with babel 3.2a	256
\captionsczech: Added \seename, \alsoname and \prefacename	258
czech-1.1a	
\noextrasczech: Removed typo, \q was restored twice, once too many.	258
czech-1.2	
General: Included some features from Kasal's czech.sty	256
czech-1.3	
General: Update for L ^A T _E X 2 _ε	256
czech-1.3d	
General: Now use \@nopatterns to produce the warning	257
Removed the use of \filedate and moved identification after the loading of	
babel.def	256
czech-1.3e	
\noextrasczech: now use \bbl@frenchspacing and \bbl@nonfrenchspacing	258
Use L ^A T _E X's \v and \r accent commands	258
czech-1.3f	
\captionsczech: Added \proofname for AMS-L ^A T _E X	258
czech-1.3g	
\captionsczech: Fixed two errors and provided translation for 'proof'	258
czech-1.3h	
General: Now use \ldf@finish to wrap up	267
Now use \LdfInit to perform initial checks	257
Replaced \undefined with \@undefined and \empty with \@empty for consis-	
tency with L ^A T _E X, moved the definition of \atcatcode right to the beginning.	256
czech-1.3i	
\dateczech: Use \def instead of \edef	258
Use \edef to define \today to save memory	258
czech-1.3j	
\captionsczech: Added \glossaryname	258
czech-1.3k	
\captionsczech: Added translation for Glossary	258
czech-3.0	
General: Added default for setting hyphenmin parameters. Values taken from	
C _S L ^A T _E X.	259
Implemented the functionality of C _S L ^A T _E X's czech.sty. The version number was	
bumped to 3.0 to minimize confusion by being higher than the last version of	
C _S L ^A T _E X.	256
\captionsczech: Updated some translations. Former translations were: 'Do-	
datek' for \appendixname and 'Index' for \indexname. Also removed spurious	
colon at the end of \ccname.	258
czech-3.1	
\cs@emdash: ensure correct catcode for the saved hyphen	261
\cs@splitattr: attribute added	264
\noextrasczech: move \languageshorthands here, so that the language group	
is always initialized correctly	258
\splithyphens: activate with split hyphens and deactivate with standard hy-	
phens, not vice versa	264
danish-1.0a	
General: Renamed babel.sty in babel.com	208
danish-1.0b	
General: Removed use of \@ifundefined	208
danish-1.1	
General: Added a warning when no hyphenation patterns were loaded.	208
Brought up-to-date with babel 3.2a	208
\captionsdanish: Added \seename, \alsoname and \prefacename	208
danish-1.2	
\captionsdanish: \headpagename should be \pagename	208
danish-1.2b	
\captionsdanish: Added a few translations	208

danish-1.3	
General: Update for L ^A T _E X 2 _ε	208
danish-1.3a	
\datedanish: Added ‘.’ to definition of \today	209
danish-1.3c	
\captionsdanish: Included some revisions from Peter Busk Larsen	208
danish-1.3f	
General: Now use \@nopatterns to produce the warning	208
Removed the use of \filedate and moved identification after the loading of	
babel.def	208
danish-1.3g	
General: Added the active double quote character as suggested by Peter Busk	
Laursen	208
danish-1.3h	
\captionsdanish: Added \proofname for AMS-L ^A T _E X	208
\extrasdanish: Added \bbl@frenchspacing	209
\noextrasdanish: Added \bbl@nonfrenchspacing	209
danish-1.3i	
\captionsdanish: Added translation of ‘Proof’	208
danish-1.3j	
General: Now use \ldf@finish to wrap up	210
Now use \LdfInit to perform initial checks	208
Replaced \undefined with \@undefined and \empty with \@empty for consis-	
tency with L ^A T _E X, moved the definition of \atcatcode right to the beginning.	208
\extrasdanish: Added definition of "~ and ="	209
Changed definition of " to print “ instead of ”	209
danish-1.3k	
\datedanish: use \def instead of \edef	209
Use \edef to define \today to save memory	209
\extrasdanish: Removed empty groups after double quote and guillemot char-	
acters	209
danish-1.3m	
\extrasdanish: Deactivate shorthands outside of Danish	209
danish-1.3n	
\captionsdanish: Added \glossaryname	208
danish-1.3o	
\captionsdanish: Added translation of ‘Glossary’	208
danish-1.3p	
\englishhyphenmins: Added default for setting of hyphenmin parameters	208
danish-1.3q	
\:-: Added redefinition of \- from dutch.ldf	209
\englishhyphenmins: Set lefthyphenmin to two	208
\extrasdanish: Added definition of "/" from dutch.ldf	209
danish-1.3r	
\extrasdanish: Made "/" a real Danish shorthand	209
dutch-2.0a	
General: Added checking of format	73
dutch-2.0b	
General: Added extrasdutch	73
dutch-2.0c	
General: Added grqq macros	73
dutch-2.1	
General: reflect change to version 2.1 in babel and changes in german v2.3	73
dutch-2.1a	
General: Incorporated Nico’s comments	73
dutch-2.1b	
General: Incorporated more comments by Nico	73
dutch-2.1c	
General: Fixed some typos	73
dutch-2.2	
General: Fixed problem with the use of " in moving arguments while " is active	73

dutch-2.3	
\@trema: \dieresis instead of \accent127	76
General: \dieresis instead of \accent127	76
When using PostScript fonts with the Adobe font-encoding, the dieresis-accent	
is located elsewhere, modified code	73
\noextrasafrikaans: Added \dieresis	75
dutch-2.3a	
General: Modified the documentation somewhat	73
dutch-3.0	
General: Modified for babel 3.0	73
Now use \adddialect if language undefined	73
dutch-3.0a	
General: Removed some problems in change log	73
dutch-3.0b	
\extrasafrikaans: added some comment chars to prevent white space	75
\noextrasafrikaans: added some comment chars to prevent white space	75
dutch-3.1	
General: Removed bug found by van der Meer	73
dutch-3.1a	
\captionsdutch: \pagename should be \headpagename	74
Removed \global definitions	74
\datedutch: Removed \global definitions	74
\extrasafrikaans: Removed \global definitions	75
\noextrasafrikaans: Removed \global definitions	75
dutch-3.2	
General: added case for "y and "Y	76
\extrasafrikaans: Save all redefined macros	75
\noextrasafrikaans: Try to restore everything to its former state	75
dutch-3.2a	
General: Added reset of catcode of @ before \endinput.	73
Renamed babel.sty in babel.com	73
dutch-3.2b	
General: removed typo (allowhpyhens)	76
dutch-3.2c	
General: Removed code to load latexhax.com	73
removed use of \@ifundefined	73
dutch-3.3	
General: Rewritten parts of the code to use the new features of babel version 3.1	73
\extrasafrikaans: Macro complete rewritten	75
\noextrasafrikaans: Macro complete rewritten	75
dutch-3.3a	
\@trema: renamed \@umlaut to \@trema	76
General: Added \save@sf@q macro from germanb and rewrote all quote macros	
to use it	75
Moved code to the beginning of the file and added \selectlanguage call	73
\captionsdutch: added \seename and \alsoname	74
dutch-3.3b	
General: Added warning, if no dutch patterns loaded	73
\captionsdutch: added \prefacename	74
\extrasafrikaans: modified handling of \dospecials and \@sanitize	75
\noextrasafrikaans: modified handling of \dospecials and \@sanitize	75
dutch-3.4b	
General: moved definition of \allowhyphens, \set@low@box and \save@sf@q to	
babel.com	75
dutch-3.5	
\captionsdutch: \headpagename should be \pagename	74
dutch-3.6	
General: Update or LaTeX2e	73
dutch-3.6c	
General: Now use \@nopatterns to produce the warning	73
Removed the use of \filedate, moved identification after the loading of ba-	
bel.def	73

dutch-3.7a	
General: Moved identification code to the top of the file	73
Moved the definition of \ij and \IJ to <code>glyphs.def</code>	76
moved the definition of the double quote character at the baseline to <code>glyphs.def</code>	75
Now use <code>\Declaredq dutch</code> to define the functions of the active double quote	76
Removed <code>\dlqq</code> , <code>\@dlqq</code> , <code>\drqq</code> , <code>\@drqq</code> and <code>\dieresis</code>	75
Rewrote the code with respect to the active double quote character	73
The support macros for the active double quote have been moved to <code>babel.def</code>	76
Use <code>\ddot</code> instead of <code>\@MATHUMLAUT</code>	76
Use more general mechanism of <code>\declare@shorthand</code>	76
<code>\afrikaanshyphenmins</code> : use <code>\dutchhyphenmins</code> to store the correct values	75
<code>\IJ</code> : Changed the kerning in the faked ij to match the dc-version of it	47
dutch-3.7b	
General: Added <code>"</code> shorthand	76
dutch-3.7c	
<code>\captionsdutch</code> : We need the <code>"</code> to be active while defining <code>\captionsdutch</code>	74
dutch-3.7d	
<code>\captionsdutch</code> : Added <code>\proofname</code> for AMS- \LaTeX	74
dutch-3.7f	
General: Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX	73
dutch-3.8a	
General: Merged in the definitions for ‘afrikaans’	73
Now use <code>\ldf@finish</code> to wrap up	76
this needs a more complicated check as ‘afrikaans’ may or may not have its own hyphenation patterns	73
<code>\noextrasafrikaans</code> : Made all definitions dependant on <code>\CurrentOption</code>	75
dutch-3.8b	
<code>\captionsdutch</code> : Use <code>Bew"ys</code> instead of <code>Bewijs</code>	74
dutch-3.8c	
General: Added the <code>"~</code> shorthand	76
dutch-3.8e	
General: Added a shorthand with the slash character	76
Forgot to replace ‘german’ by ‘dutch’ when copying definition for <code>"~</code>	76
Removed empty groups after double quote characters	76
<code>\dateafrikaans</code> : use <code>\def</code> instead of <code>\edef</code>	75
Use <code>\edef</code> to define <code>\today</code> to save memory	75
<code>\datedutch</code> : use <code>\def</code> instead of <code>\edef</code>	74
Use <code>\edef</code> to define <code>\today</code> to save memory	74
dutch-3.8h	
<code>\afrikaanshyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	75
<code>\captionsdutch</code> : Added <code>\glossaryname</code>	74
dutch-3.8i	
<code>\-:</code> <code>\-</code> should use <code>\bbl@allowhyphens</code>	76
General: <code>" /</code> should use <code>\bbl@allowhyphens</code>	76
english-2.0a	
General: Added checking of format	77
english-2.1	
General: Reflect changes in babel 2.1	77
english-2.1a	
General: Incorporated Nico’s comments	77
english-2.1b	
General: merged <code>USenglish.sty</code> into this file	77
english-2.1c	
General: fixed typo in definition for american language found by Werenfried Spit (<code>nspit@fys.ruu.nl</code>)	77
english-2.1d	
General: Fixed some typos	77
english-3.0	
General: Modified for babel 3.0	77
Now use <code>\adddialect</code> for american	78

Now use <code>\adddialect</code> if language undefined	77
english-3.0a	
General: Removed bug found by van der Meer	77
english-3.0b	
General: Removed <code>\global</code> definitions	78
<code>\captionenglish</code> : <code>\pagename</code> should be <code>\headpagename</code>	78
Removed <code>\global</code> definitions	78
<code>\dateamerican</code> : Removed <code>\global</code> definitions	80
<code>\dateenglish</code> : Removed <code>\global</code> definitions	79
english-3.0c	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	77
english-3.0d	
General: removed use of <code>\@ifundefined</code>	77
english-3.1	
General: Rewrote parts of the code to use the new features of babel version 3.1	77
english-3.1a	
<code>\captionenglish</code> : added <code>\seename</code> and <code>\alsoname</code>	78
english-3.1b	
<code>\captionenglish</code> : added <code>\prefacename</code>	78
english-3.2	
<code>\captionenglish</code> : <code>\headpagename</code> should be <code>\pagename</code>	78
english-3.3	
General: Update or $\text{\LaTeX 2}\epsilon$	77
english-3.3c	
General: Now use <code>\@nopatterns</code> to produce the warning	77
Removed the use of <code>\filedate</code> and moved the identification after the loading of <code>babel.def</code>	77
english-3.3d	
General: Only define <code>american</code> as a dialect when no separate patterns have been loaded	78
english-3.3e	
<code>\captionenglish</code> : Added <code>\proofname</code> for AMS- \LaTeX	78
english-3.3g	
General: Allow <code>british</code> as the name of the UK patterns	77
Allow <code>USenglish</code> as the name of the american patterns	78
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX	77
<code>\captionenglish</code> : Construct control sequence on the fly	78
<code>\dateenglish</code> : Construct control sequence on the fly	79
<code>\noextrasenglish</code> : Construct control sequences on the fly	80
english-3.3h	
General: Moved the definition of <code>\atcatcode</code> right to the beginning.	77
Now use <code>\ldf@finish</code> to wrap up	80
Now use <code>\LdfInit</code> to perform initial checks	77
english-3.3i	
<code>\dateamerican</code> : use <code>\def</code> instead of <code>\edef</code>	80
Use <code>\edef</code> to define <code>\today</code> to save memory	80
<code>\dateenglish</code> : use <code>\def</code> instead of <code>\edef</code>	79
Use <code>\edef</code> to define <code>\today</code> to save memory	79
english-3.3j	
General: Also allow american english hyphenation patterns to be used for ‘english’	77
Ensure that <code>\l@USenglish</code> is always defined	78
<code>\captionenglish</code> : Added <code>\glossaryname</code>	78
<code>\dateenglish</code> : Make sure that the value of <code>\today</code> is correct for both options ‘american’ and ‘USenglish’	79
english-3.3k	
General: Added support for canadian	77, 78
english-3.3l	
General: Added missing backslash	78
english-3.3m	
<code>\englishhyphenmins</code> : Added default for setting of <code>hyphenmin</code> parameters	78

english-3.3n	
General: Added support for australian and newzealand	77, 78
<code>\dateaustralian</code> : Add australian date	79
<code>\dateenglish</code> : Added support for ‘Australian’ and ‘Newzealand’	79
english-3.3o	
General: Make sure that british patterns are used if they were loaded	77
<code>\dateenglish</code> : Explicitly choose the UK form of date	79
esperanto0-1.4o	
<code>\noextrasesperanto</code> : Moved the check for math to babel.def	69
esperanto-1.0a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	68
esperanto-1.0b	
General: Removed use of <code>\makeatletter</code>	68
esperanto-1.1	
General: Added a warning when no hyphenation patterns were loaded.	68
Brought up-to-date with babel 3.2a	68
<code>\captionesperanto</code> : Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	68
esperanto-1.2	
General: Included code from <code>esperant.sty</code>	68
esperanto-1.3	
<code>\captionesperanto</code> : <code>\headpagename</code> should be <code>\pagename</code>	68
Repaired a number of mistakes, indicated by D. Ederveen	68
<code>\dateesperanto</code> : Removed the capitals from <code>\today</code>	69
esperanto-1.4a	
General: Updated for L ^A T _E X 2 _ε	68
<code>\captionesperanto</code> : added missing closing brace	68
esperanto-1.4d	
General: Removed the use of <code>\filedate</code> , moved Identification after loading of	
<code>babel.def</code>	68
Use <code>\@nopatterns</code> for the warning	68
esperanto-1.4e	
General: Moved identification code to the top of the file	68
esperanto-1.4f	
General: Corrected typos (PR1652)	68
esperanto-1.4g	
<code>\captionesperanto</code> : Added <code>\proofname</code> for AMS-L ^A T _E X	68
esperanto-1.4h	
General: Added a few shorthands	69
esperanto-1.4i	
General: Moved the definition of <code>\atcatcode</code> right to the beginning.	68
Now use <code>\ldf@finish</code> to wrap up	70
Now use <code>\LdfInit</code> to perform initial checks	68
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with L ^A T _E X	68
<code>\captionesperanto</code> : Replaced ‘Proof’ by ‘Pruvo’ PR 2207	68
esperanto-1.4j	
General: fixed typo in table caption (funtion instead of function)	68
esperanto-1.4k	
<code>\dateesperanto</code> : Removed Rthe use of <code>\edef</code> again	69
Use <code>\edef</code> to define <code>\today</code> to save memory	69
esperanto-1.4l	
General: Added a shorthand definition on system level	69
esperanto-1.4n	
<code>\noextrasesperanto</code> : Added a check for math mode to the definition of the shorthand character	69
esperanto-1.4p	
<code>\captionesperanto</code> : Added <code>\glossaryname</code>	68
esperanto-1.4q	
<code>\captionesperanto</code> : Added translation for Glossary	68
<code>\esper</code> : Removed the extra level of expansion for more than five items, as was done in L ^A T _E X	69

esperanto-1.4t	
<code>\esper</code> : Added the missing ‘r’ in these macros	69
estonian-1.0b	
General: corrected typos	247
estonian-1.0c	
<code>\captionsestonian</code> : Added <code>\proofname</code> for AMS- \LaTeX	248
estonian-1.0d	
General: The second argument was missing in the definition of some of the double- quote shorthands	250
<code>\captionsestonian</code> : Added translation of ‘Proof’	248
<code>\noextrasestonian</code> : Removed the code that changes category, lower case, uper case and space factor codes	249
estonian-1.0e	
General: Now use <code>\ldf@finish</code> to wrap up	250
Now use <code>\LdfInit</code> to perform initial checks	247
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consis- tency with \LaTeX , moved the definition of <code>\atcatcode</code> right to the beginning.	247
estonian-1.0f	
General: Removed empty groups after double quote and guillemot characters	250
<code>\dateestonian</code> : use <code>\def</code> instead of <code>\edef</code>	248
Use <code>\edef</code> to define <code>\today</code> to save memory	248
estonian-1.0g	
General: use <code>\bbl@t@one</code> instead of <code>\bbl@next</code>	249
estonian-1.0h	
<code>\captionsestonian</code> : Added <code>\glossaryname</code>	248
<code>\estonianhyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	249
estonian-1.0j	
<code>\captionsestonian</code> : Replaced the translation of ‘Proof’	248
estonian-1.0k	
<code>\captionsestonian</code> : Added translation of ‘Glossary’	248
<code>\et@gentilde</code> : do not redefine caron any more because the default one looks good enough	249
renamed macros <code>\gentilde</code> and <code>\newtilde</code> to <code>\et@gentilde</code> and <code>\et@newtilde</code> 	249
use tilde for all letters except s and z (instead of using caron for all letters except o), like other <code>babel</code> language packages do (this fixes the display of ñ when using the <code>utf8</code> package)	249
<code>\et@newtilde</code> : merged updates in the definition <code>\lower@umlaut</code> into <code>\et@newtilde</code> : removed <code>\allowhyphens</code> and added <code>\bgroup</code>	249
finnish-1.0a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	229
finnish-1.0b	
General: Removed use of <code>\@ifundefined</code>	229
finnish-1.1	
General: Added a warning when no hyphenation patterns were loaded.	229
Brought up-to-date with <code>babel 3.2a</code>	229
<code>\captionsestfinnish</code> : <code>\headpagename</code> should be <code>\pagename</code>	229
Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	229
finnish-1.1.1	
<code>\captionsestfinnish</code> : Added translations	229
finnish-1.2	
General: Update for \LaTeX 2_ϵ	229
finnish-1.3c	
General: Now use <code>\@nopatterns</code> to produce the warning	229
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code>	229
finnish-1.3d	
General: Removed a few references to <code>babel.com</code>	229
finnish-1.3e	
<code>\datefinnish</code> : Added a ‘.’ after the number of the day	230
finnish-1.3f	
<code>\finnishhyphenmins</code> : use <code>\finnishhyphenmins</code> to store the correct values	231

\noextrasfinnish: Added the setting of \frenchspacing	230
Added the setting of more hyphenation parameters, according to PR1027	230
finnish-1.3g	
\:-: Added change of \-	231
\captionsfinnish: Added \proofname for AMS-L ^A T _E X	229
\noextrasfinnish: Added the active double quote	230
finnish-1.3h	
\captionsfinnish: Added finnish word for ‘Proof’	229
finnish-1.3i	
General: Now use \ldf@finish to wrap up	231
Now use \LdfInit to perform initial checks	229
Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X, moved the definition of \atcatcode right to the beginning.	229
finnish-1.3k	
\datefinnish: use \def instead of \edef	230
Use \edef to define \today to save memory	230
\noextrasfinnish: Removed empty groups after double quote and guillemot characters	230
finnish-1.3m	
\noextrasfinnish: Added mising closing brace	230
finnish-1.3n	
\:-: \allowhyphens should have been \bbl@allowhyphens	231
\captionsfinnish: Added \glossaryname	229
\finishhyphenmins: Now use \providehyphenmins to provide a default value	231
\noextrasfinnish: Deactive shorthands outside of Finnish	230
finnish-1.3o	
\captionsfinnish: Provided translation for Glossary	229
finnish-1.3p	
\noextrasfinnish: “= should also use \bbl@allowhyphens	231
finnish-1.3q	
General: Small documentation fix	229
galician 4.3	
General: \dots is removed and instead \dots and \dots are changed, by redefining \ldotc, \dotc and \textellipsis or \dots	189
\sin, \arcsin and \sinh are set to produce the same as \sen, \arcsen and \senh	191
Added \msc	188
cosec and senh moved from \galicianoperators to the main group	191
Removed the shorthand for ç. It existed in medieval galician-portuguese, but that does not seem a reason to be included here.	186
Removed the shorthands "er and "ER, they don’t exist in galician.	195
Removed the Spanish et sign.	186
Set the default to \unspacedoperators	191
Set the default to do	187
german-2.6l	
General: Making germanb behave like german needs some more work besides defining \CurrentOption	81
germanb-1.0a	
General: Incorporated Nico’s comments	81
germanb-1.0b	
General: fixed typo in definition for austrian language found by Werenfried Spit	
nspit@fys.ruu.nl	81
germanb-1.0c	
General: Fixed some typos	81
germanb-1.1	
General: When using PostScript fonts with the Adobe fontencoding, the dieresis-accent is located elsewhere, modified code	81
\noextrasaustrian: Added \dieresis	83
germanb-1.1a	
General: Modified the documentation somewhat	81
germanb-2.0	
General: Modified for babel 3.0	81

Now use <code>\adddialect</code> for austrian	82
Now use <code>\adddialect</code> if language undefined	82
germanb-2.0a	
General: Removed some problems in change log	81
germanb-2.0b	
<code>\extrasaustrian</code> : added some comment chars to prevent white space	83
<code>\noextrasaustrian</code> : added some comment chars to prevent white space	83
germanb-2.1	
General: Removed bug found by van der Meer	81
germanb-2.2	
General: Removed global assignments, brought uptodate with <code>german.tex</code> v2.3d	81
<code>\captionsaustrian</code> : <code>\pagename</code> should be <code>\headpagename</code>	82
Removed <code>\global</code> definitions	82
<code>\extrasaustrian</code> : Save all redefined macros	83
<code>\noextrasaustrian</code> : Try to restore everything to its former state	83
germanb-2.2a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	81
germanb-2.2d	
General: Removed use of <code>\@ifundefined</code>	82
germanb-2.3	
General: Rewritten parts of the code to use the new features of babel version 3.1	81
germanb-2.3e	
General: Added <code>\save@sf@q</code> macro and rewrote all quote macros to use it	83
Added warning, if no german patterns loaded	82
Brought up-to-date with <code>german.tex</code> v2.3e (plus some bug fixes) [br]	81
<code>\captionsaustrian</code> : Added <code>\prefacename</code> , <code>\seename</code> and <code>\alsoname</code>	82
<code>\dategerman</code> : Added <code>\month@german</code>	82
germanb-2.3h	
General: moved definition of <code>\allowhyphens</code> , <code>\set@low@box</code> and <code>\save@sf@q</code> to <code>babel.com</code>	83
germanb-2.4	
<code>\captionsaustrian</code> : <code>\headpagename</code> should be <code>\pagename</code>	82
germanb-2.5	
General: Update or L ^A T _E X 2 _ε	81
germanb-2.5c	
General: Now use <code>\@nopatterns</code> to produce the warning	82
Removed the use of <code>\filedate</code> and moved the identification after the loading of <code>babel.def</code>	81
germanb-2.6a	
General: <code>\umlautlow</code> and <code>\umlauthigh</code> moved to <code>glyphs.dtx</code> , as well as <code>\newumlaut</code> (now <code>\lower@umlaut</code>)	83
Moved all quotation characters to <code>glyphs.dtx</code>	83
Moved the identification to the top of the file	81
Rewrote the code that handles the active double quote character	81
Use <code>\ddot</code> instead of <code>\@MATHUMLAUT</code>	83
<code>\noextrasaustrian</code> : All the code to handle the active double quote has been moved to <code>babel.def</code>	83
Removed <code>\3</code> as it is no longer in <code>german.ldf</code>	83
use <code>\germanhyphenmins</code> to store the correct values	83
germanb-2.6b	
<code>\captionsaustrian</code> : Added <code>\proofname</code> for AMS-L ^A T _E X	82
germanb-2.6c	
General: added the <code>\allowhyphens</code>	83
Moved <code>\german@dq@disc</code> to <code>babel.def</code> , calling it <code>\bbl@disc</code>	83
<code>\noextrasaustrian</code> : Use decimal number instead of hat-notation as the hat may be activated	83
germanb-2.6d	
General: Moved the definition of <code>\atcatcode</code> right to the beginning.	81
Now use <code>\ldf@finish</code> to wrap up	84
Now use <code>\LdfInit</code> to perform initial checks	82
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with L ^A T _E X	81

\captionsaustrian: Construct control sequence on the fly	82
\noextrasaustrian: Construct control sequence \extrasgerman or \extrasaustrian on the fly	83
germanb-2.6f	
General: Copied the coding for "f from german.dtx version 2.5d	84
use \SS instead of SS, removed braces after \ss	84
\ck: Now use \shorthandon and \shorthandoff	84
\dateaustrian: use \def instead of \edef	83
Use \edef to define \today to save memory	83
\dategerman: use \def instead of \edef	82
Use \edef to define \today to save memory	82
germanb-2.6i	
\noextrasaustrian: Deactivate shorthands outside of German	83
germanb-2.6j	
\captionsaustrian: Added \glossaryname	82
\noextrasaustrian: Now use \providehyphenmins to provide a default value	83
germanb-2.6k	
\noextrasaustrian: Turn frenchspacing on, as in german.sty	83
germanb-2.6m	
General: Correted a typo	81
greek-1.0b	
General: Moved the definition of \atcatcode right to the beginning	98
Now use \ldf@finish to wrap up	105
Now use \LdfInit to perform initial checks	99
Replaced \undefined with \@undefined and \empty with \@empty for consis- tency with L ^A T _E X	98
\textgreek: Added a level of braces to keep encoding change local	100
greek-1.0c	
\greek@tilde: Added command	103
greek-1.1	
\Grtoday: Added macro \Grtoday	101
greek-1.1a	
\dategreek: Fixed typo, Oktwbr'iou instead of Oktobr'iou	101
\greek@Alph: removed two superfluous @'s which made \@alph undefined ..	102
greek-1.1b	
\noextragreek: Added setting of \uccodes (after kdgreek.sty)	104
Added shorthand for \char255	104
Made tilde expand to a tilde with \catcode 12	104
greek-1.1c	
General: Added a couple of symbols, needed for \greeknumeral	104
\noextragreek: fixed two typos	104
greek-1.1d	
\dategreek: Macro \gr@month now produces the name of the month	101
greek-1.1e	
General: Added caption name for proof	100
Most symbols are removed and are now defined in package grsymb	104
\gr@month: Macro added	101
\noextragreek: Added lowercase code for v	104
Added uppercase code for special letter "v". Uppercase code for accents is now 9f, instead of ff	104
Shorthand is changed. Active character is now \char159	104
greek-1.2	
General: Added caption names for \polutonikogreek	100
Classical Greek is now a dialect	98
\gr@cc@greek: Added macro \datepolutonikogreek	101
Added macro \gr@cl@month	101
\noextragreek: Added lowercase codes for "modern" greek	104
Added uppercase codes for "modern" Greek. The old codes are now for "Polu- toniko" Greek	104
Definitions for "modern" Greek are now the definitions of "Polutoniko" Greek	104
greek-1.2a	
General: filename lgrenc.def now lowercase	99

<code>\dategreek</code> : Use <code>\edef</code> to define <code>\today</code>	101
<code>\noextragreek</code> : Need shorthand to exist for “monotoniko” Greek, not “polutoniko” Greek	104
greek-1.2b	
General: Classical Greek is now called “Polutoniko” Greek. The previous name was at least misleading	98
<code>\dategreek</code> : use <code>\def</code> instead of <code>\edef</code>	101
<code>\gr@num@iii</code> : No longer use <code>\</code> in the expansion of the <code>\gr@num@x</code> macros as they need to be expandable	103
greek-1.2c	
General: Package <code>grsymb</code> has been eliminated because the CB fonts v2.0 do not include certain symbols and so the remaining symbol definitions have been moved here	104
This version conforms to version 2.0 of the CB fonts and consequently we added a few new symbol-producing commands	98
greek-1.2e	
<code>\greek@Roman</code> : Moved redefinition of <code>\@roman</code> back to the language specific file	102
greek-1.2f	
<code>\ltx@amp</code> : Now switch the definition of <code>\&</code> from <code>\extragreek</code>	103
greek-1.3a	
General: polutoniko is now an attribute to Greek, no longer a ‘dialect’	98
<code>\gr@cc@greek</code> : removed macro <code>\datepolutonikogreek</code>	101
greek-1.3b	
<code>\greeknumeral</code> : Added <code>\expandafter</code> and <code>\number</code> (PR3000) in order to make a counter an acceptable argument	102
greek-1.3c	
<code>\ltx@amp</code> : Added a missing opening brace	103
greek-1.3d	
General: Fixed typo, <code>bl’epe ep’ishc</code> instead of <code>bl’pe ep’ishc</code>	100
<code>\greek@Roman</code> : <code>\@roman</code> and <code>\@Roman</code> need to be added to <code>\extraspolutonikogreek</code>	102
greek-1.3e	
<code>\greek@Roman</code> : <code>\@roman</code> and <code>\@Roman</code> need <i>not</i> be in <code>\extraspolutonikogreek</code> when they are already in <code>\extragreek</code>	102
<code>\noextragreek</code> : <code>\extragreek</code> and <code>\extraspolutonikogreek</code> should be complementary	104
greek-1.3f	
General: Added some code to make older documents work	99
greek-1.3g	
General: <code>\noextraspolutonikogreek</code> was missing	99
greek-1.3h	
<code>\captionsgreek</code> : Added <code>\glossaryname</code>	100
<code>\greekhyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	100
greek-1.3i	
<code>\captionsgreek</code> : The final sigma in all names appears as ‘s’ instead of ‘c’. . .	100
<code>\noextragreek</code> : uc code of ‘v’ is switched to V so that mixed text appears correctly in headers.	104
greek-1.3j	
<code>\noextragreek</code> : Because other languages might make the caret active, we can’t use the double caret notation here	104
Use the tilde as an alias for character 159	104
greek-1.3k	
<code>\greek@tilde</code> : Make sure the character “ is not active during the definition of <code>\greek@tilde</code>	103
<code>\textgreek</code> : Added <code>\leavevmode</code> as was done with <code>\latintext</code>	100
greek-1.3l	
General: Commented these lines out as this change has made it into L ^A T _E X itself.	105
heb209 1.0a	
General: Initial version. Provides <code>hebrew_newcode</code> , <code>hebrew_oldcode</code> and <code>hebrew_p</code> style files for L ^A T _E X 2.09 (by Boris Lavva)	392

hebfdd-1.0a	
General: Initial version. Supports only 7-bit LHE font encoding and all available Hebrew T _E X fonts (by Boris Lavva)	377
hebfdd-1.0b	
General: fixed lhecmr.fd to use oldjaf10 for a slanted font available Hebrew T _E X fonts (by Tzafrir Cohen)	377
hebfdd-1.1a	
General: Adding 8-bit HE8 fonts. Note that most of them cannot be distributed with hebl ^A T _E X (by Tzafrir Cohen)	377
hebfdd-1.2a	
General: Adding configurations for the Culmus fonts, currently 0.90 (by Tzafrir Cohen)	377
hebfdd-1.2b	
General: Reinstated the test whether LHE or HE8 is to be used	377
hebinp 1.0a	
General: Initial version. Provides 8859-8, cp862, cp1255, and old 7-bit input encodings (by Boris Lavva)	371
hebinp 1.1	
General: Renamed hebrew letters: <code>\alef</code> to <code>\hebalef</code> etc. (by Tzafrir Cohen)	371
hebinp 1.1a	
General: Renamed CP1255 nikud <code>\patah</code> to <code>\hebpatah</code> etc. Added the macro <code>\DisableNikud</code> - may not be a good idea (by Tzafrir Cohen)	371
hebrew-1.2c	
General: Typo's in the docstrip guards made HE8nachlieli.fd unusable	377
hebrew-2.0b	
<code>\captionshbrew</code> : Added <code>\glossaryname</code>	336
hebrew-2.3h	
<code>\hebrewencoding</code> : Make LHE the default encoding for compatibility reasons	335
hebrew 0.1	
General: Preliminary L ^A T _E X Hebrew option (by Sergio Fogel)	334
hebrew 0.2	
General: Corrections and additions (by Rama Porrat)	334
hebrew 0.6	
General: Additions (by Yael Dubinsky)	334
hebrew 1.2	
General: Bilingual tables, penalties, documentation and more changes (by Yaniv Bargury)	334
hebrew 1.30	
General: Font selection, various (by Alon Ziv)	334
hebrew 1.31	
General: Bug fixes (by Alon Ziv)	334
hebrew 1.32	
General: Made font-change command for numbers ' <code>\protect</code> 'ed (by Alon Ziv)	334
hebrew 1.33	
General: Made <code>\refstepcounter</code> work using <code>\@l_{tor}</code> (by Alon Ziv)	334
hebrew 1.34	
General: Moved font loading to another file. Added <code>\mainsec</code> . Made all text strings be produced by control codes (similar to L ^A T _E X2.09 Mar '92). Fixed <code>\noindent</code> (by Alon Ziv)	334
hebrew 1.35	
General: Moved the texts to a file selected by the current encoding (by Alon Ziv)	334
hebrew 1.36	
General: Use T _E X tricks to redefine <code>\theXXXX</code> without keeping old definitions. Use only <code>\@eng</code> for direction/font change (removed <code>\@l_{tor}</code>). Switched from use of <code>\mainsec</code> to code taken from <code>babel</code> system (by Alon Ziv)	334
hebrew 1.37	
General: Use <code>\add@around</code> in defining font size commands. Small bug fixes (by Alon Ziv)	334
hebrew 1.38	
General: <code>\everypar</code> changed so that <code>\noindent</code> works unmodified (by Alon Ziv, thanks to Chris Rowley)	334

hebrew 1.39	
General: Redefined primitive sectioning commands. Changed <code>\include</code> so it finds <code>.h</code> , <code>.xet</code> , and <code>.ltx</code> files (no extension needed). Reinstated use of <code>\@ltor</code> (by Alon Ziv)	334
hebrew 1.40	
General: Added the <code>\@brackets</code> hack (by Alon Ziv)	334
hebrew 1.41	
General: Reworked towards using NFSS2. Changed some macro names to be more logical: renamed <code>\@ltor</code> to <code>\@number</code> , <code>\@eng</code> to <code>\@latin</code> , and (in <code>hebrew.ldf</code>) <code>\@heb</code> to <code>\@hebrew</code> (by Alon Ziv)	334
hebrew 1.42	
General: Made list environments work better. Fixed thebibliography environment (by Alon Ziv)	334
hebrew 2.0a	
General: Completely rewritten for L ^A T _E X 2 _ε and <code>babel</code> support. Various input and font encodings (with NFSS2) are supported too. The original <code>hebrew.sty</code> is divided to a number of packages and definition files for better readability and extensibility. Added some user- and code-level documentation inside the <code>.dtx</code> and <code>.fdd</code> files, and L ^A T _E X-driven installation with <code>hebrew.ins</code> (by Boris Lavva)	334
hebrew 2.0b	
<code>\hebrewhyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	336
hebrew 2.1	
General: corrections from Sivan Toledo: sender name in letter, and section name in headings. (by Tzafrir Cohen)	334
hebrew 2.2	
General: renamed hebrew letters to <code>heb*</code> (e.g.: alef renamed to <code>hebalef</code>) (by Tzafrir Cohen)	334
hebrew 2.3	
General: added several <code>\@ifclassloaded{slides}</code> to allow the use of the slides class. (by Tzafrir Cohen)	334
hebrew 2.3a	
General: The documentation should now be built fine (broken since at least 2.1, and probably 2.0) (by Tzafrir Cohen)	334
hebrew 2.3b	
General: minor clean-ups. The documentation builds now with no warnings. Also removed <code>\R</code> from the caption macro (added in 2.1) Added internal <code>\@ensure@L</code> and <code>\@ensure@R</code> (Is there a real need for them? Maybe should they be exposed?) (by Tzafrir Cohen)	334
hebrew 2.3c	
General: a temporary fix to the <code>\gim</code> macro. Should be replaced by stuff from <code>hebcsl</code> . (by Tzafrir Cohen)	334
hebrew 2.3d	
General: Initial support for the prosper class. Added <code>\arabicnorl</code> . (by Tzafrir Cohen)	334
hebrew 2.3e	
General: Removing <code>hebtech</code> from this distribution (not relevant to <code>babel</code>), added <code>\HeblatexEncoding</code> . some docs cleanup (by Tzafrir Cohen)	334
hebrew 2.3f	
General: redefined <code>\list</code> instead of redefining every environment that uses it. some <code>pscolor</code> handling, removed <code>HeblatexEncoding</code> (don't use 2.3e) (by Tzafrir Cohen)	334
icelandic-1.1	
General: Added definitions for old icelandic.	214
icelandic-1.1a	
General: Added definitions for formatting numbers in Icelandic and some extra utilities.	215
icelandic-1.1b	
General: Added references to various publications used	215
icelandic-1.1c	
General: Removed code already present in <code>babel.def</code>	211
<code>\dateicelandic</code> : use <code>\def</code> instead of <code>\edef</code>	213

Use <code>\edef</code> to define <code>\today</code> to save memory	213
icelandic-1.1e	
<code>\noextrasicelandic</code> : Deactivate shorthands outside of Icelandic	213
icelandic-1.1f	
<code>\captionsicelandic</code> : Added <code>\glossaryname</code>	213
<code>\noextrasicelandic</code> : Now use <code>\providehyphenmins</code> to provide a default value	213
icelandic-1.1g	
<code>\captionsicelandic</code> : Added translation for Glossary	213
Only use 7-bit ASCII characters in order to keep the texts input encoding independent	213
irish-1.0b	
General: Corrected typo (PR1652)	94
irish-1.0c	
<code>\captionsirish</code> : Added <code>\proofname</code> for AMS- \LaTeX	94
irish-1.0e	
General: Now use <code>\ldf@finish</code> to wrap up	95
Now use <code>\LdfInit</code> to perform initial checks	94
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX , moved the definition of <code>\atcatcode</code> right to the beginning.	94
irish-1.0f	
<code>\captionsirish</code> : Added missing translations provided in PR 2719	94
<code>\dateirish</code> : use <code>\def</code> instead of <code>\edef</code>	94
Use <code>\edef</code> to define <code>\today</code> to save memory	94
<code>\irishhyphenmins</code> : Added definition of <code>\hyphenmins</code>	94
irish-1.0h	
<code>\captionsirish</code> : Added <code>\glossaryname</code>	94
<code>\irishhyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	94
italian-0.99	
General: First version, from english.doc	137
italian-1.0	
General: Modified for babel 3.0	137
Now use <code>\adddialect</code> if language undefined	138
italian-1.0a	
General: removed typo	137
italian-1.0b	
General: Removed bug found by van der Meer	137
italian-1.0c	
<code>\captionsitalian</code> : <code>\pagename</code> should be <code>\headpagename</code>	138
Removed <code>\global</code> definitions	138
<code>\dateitalian</code> : Removed <code>\global</code> definitions	139
italian-1.0d	
<code>\captionsitalian</code> : ‘contiene’ substituted by ‘Allegati’ as suggested by Marco Bozzo (BOZZO@CERNVM).	138
italian-1.0e	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	137
italian-1.0h	
General: Removed use of <code>\@ifundefined</code>	138
italian-1.1	
General: Added a warning when no hyphenation patterns were loaded.	138
Brought up-to-date with babel 3.2a	137
<code>\captionsitalian</code> : <code>\headpagename</code> should be <code>\pagename</code>	138
Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	138
italian-1.2	
General: Update for \LaTeX	137
italian-1.2b	
<code>\captionsitalian</code> : Changed some of the words following suggestions from Claudio Beccari	138
<code>\italianhyphenmins</code> : Added setting of left and righthyphenmin according to Claudio Beccari’s suggestion	139
<code>\noextrasitalian</code> : Added setting of club- and widowpenalty	139
Added setting of finallyphendemerits	139

italian-1.2e	
General: Now use <code>\@nopatterns</code> to produce the warning	138
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code>	137
italian-1.2f	
General: Updated for babel 3.5	137
italian-1.2g	
<code>\captionsitalian</code> : Added <code>\proofname</code> for AMS- \LaTeX	138
italian-1.2h	
<code>\captionsitalian</code> : Added translation of ‘Proof’	138
<code>\noextrasitalian</code> : Now give the apostrophe a lowercase code	139
italian-1.2i	
General: Now use <code>\ldf@finish</code> to wrap up	145
Now use <code>\LdfInit</code> to perform initial checks	138
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX , moved the definition of <code>\atcatcode</code> right to the beginning.	137
italian-1.2j	
General: Added braces around second arg of <code>\LdfInit</code>	138
italian-1.2l	
General: Added <code>\unit</code> , <code>\ap</code> , and <code>\ped</code> macros	137
Added useful macros for fulfilling ISO 31/XI regulations	141
<code>\noextrasitalian</code> : Changed example “begl’italiani” (obsolete spelling) with another, “nell’altezza”, that behaves the same way	139
italian-1.2m	
General: Added support for etymological hyphenation	137
Support for etymological hyphenation	139
<code>\italianhyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	139
italian-1.2n	
General: Added several commands for the caporali double quotes and for simplifying the accented vowel input	137
Completely modified etymological hyphenation facility	137
Completely new etymological hyphenation facility	140
italian-1.2o	
General: Added <code>\glossaryname</code>	137
italian-1.2p	
General: Removed redefinition of <code>\add@acc</code> since its functionality has been introduced into the kernel of \LaTeX 2001/06/01	137, 142
italian-1.2q	
General: Added test for avoiding conflict with package <code>units.sty</code> ; adjusted caporali functionality, since the previous one did not work with the standard (although obsolete) slides class file	137
Redefined the caporali machinery so as to avoid incompatibilities with the slides class, as there are no Cyrillic slides fonts as there are for Latin script	143
<code>\ped</code> : Added testing for avoiding conflicts with the <code>units.sty</code> package	142
italian-1.2r	
<code>\it@cwm</code> : Added <code>\nobreak</code> to <code>\it@cwm</code> and corrected <code>\it@next</code>	140
italian-1.2s	
General: Corrected email of CB	137
italian-1.2t	
<code>\noextrasitalian</code> : Added <code>\@clubpenalty</code> to the italian specific settings, otherwise any sectioning command restores it to the default value valid for english and most other languages	139
latin-0.99	
General: Added shortcuts for breve, macron, and etymological hyphenation (CB)	146
First version, from <code>italian.dtx</code> (CB)	146
latin-1.2	
General: Added suggestions from Krzysztof Konrad Żelechowski (CB)	146
latin-2.0	
General: Completely new etymological hyphenation (CB)	146
latin-2.0a	
General: Revised by JB	146
<code>\latinhyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	148

latin-2.0b	
General: Simplified shortcuts for etymological hyphenation; modified breve and macro shortcuts; language attribute medieval declared	146
latin-2.0c	
General: Restored caret and equals sign category codes before exiting	146
latin-2.0d	
General: Restored caret and equals sign category codes before exiting	146
latin-2.0e	
General: Introduced the language attribute ‘withprosodicmarks’; modified use of breve and macron shortcuts in order to avoid possible conflicts with other packages	146
latin-2.0f	
<code>\datelatin</code> : Added a comment character to prevent unwanted space	148
latin-2.0g	
General: Added a <code>\nobreak</code>	152
<code>\LatinMarksOff</code> : Added two commands	151
<code>\ProsodicMarks</code> : changed <code>\allowhyphens</code> to <code>\bbl@allowhyphens</code>	150
latin-2.0h	
<code>\LatinMarksOff</code> : Added missing backslash	151
Removed the setting of <code>\LatinMarksOff</code> from <code>\extraslatin</code>	151
latin-2.0i	
General: Corrected the <code>\@clubpenalty</code> problem	146
latin-2.0j	
General: Added a missing comment char and a missing closing brace	146
latin-2.0k	
<code>\LatinMarksOff</code> : Set the <code>\LatinMarks...</code> commands equal to the <code>\ProsodicMarks...</code> commands for compatibility	151
<code>\ProsodicMarks</code> : Restore category codes rather than return them to their \TeX default values. <i>And</i> do that outside of the command definition	151
Use <code>\active</code> instead of 13	150
<code>\ProsodicMarksOff</code> : Save current category codes of equals sign and caret in order to restore them later	150
latin-2.0l	
General: Added two missing <code>\end</code> macro’s	146
lsorbian-0.1	
General: First version	326
lsorbian-1.0b	
<code>\captionlsorbian</code> : Added <code>\proofname</code> for AMS- \LaTeX	326
lsorbian-1.0d	
General: Now use <code>\ldf@finish</code> to wrap up	327
Now use <code>\LdfInit</code> to perform initial checks	326
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX , moved the definition of <code>\atcatcode</code> right to the beginning.	326
lsorbian-1.0e	
<code>\newdatelsorbian</code> : use <code>\def</code> instead of <code>\edef</code>	326
Use <code>\edef</code> to define <code>\today</code> to save memory	326
lsorbian-1.0f	
<code>\captionlsorbian</code> : Added <code>\glossaryname</code>	326
lsorbian-1.0g	
General: Make this work for more than one option name	327
Make this work for more than one option name.	327
This file can be loaded under more than one name.	326
<code>\captionlsorbian</code> : Make this work for more than one option name.	326
<code>\newdatelsorbian</code> : Make this work for more than one option name.	326
<code>\olddatelsorbian</code> : Make this work for more than one option name.	327
magyar-1.0a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	232
magyar-1.0b	
General: Removed use of <code>\@ifundefined</code>	233
magyar-1.1	
General: Added a warning when no hyphenation patterns were loaded.	233
Brought up-to-date with babel 3.2a	232

\captionsmagyar: \headpagename should be \pagename	233
Added \seename, \alsoname and \prefacename	233
magyar-1.1c	
\captionsmagyar: Added translations, fixed typos	233
\ondatemagyar: The date number should not be followed by a dot.	234
magyar-1.1d	
General: Further spelling corrections	232
\datemagyar: Rewritten to produce the correct date format	234
\ondatemagyar: Renamed from \datemagyar; no longer redefines \today.	234
magyar-1.1e	
General: Still more spelling corrections	232
magyar-1.2	
General: Update for L ^A T _E X 2 _ε	232
magyar-1.3c	
General: Now use \@nopatterns to produce the warning	233
Removed the use of \filedate and moved identification after the loading of babel.def	232
magyar-1.3e	
\captionsmagyar: Added \proofname for AMS-L ^A T _E X	233
magyar-1.3f	
\captionsmagyar: translated Proof and replaced some translations	233
magyar-1.3g	
General: Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X	232
magyar-1.3h	
General: Now use \ldf@finish to wrap up	246
Now use \LdfInit to perform initial checks	233
magyar-1.4a	
General: order inverting in headings/titles/captions; definite article handling; ac- tive char for special hyphenation	232
\captionsmagyar: the initial letter of fejezet, táblázat, rész, lásd changed to lowercase	233
\datemagyar: Use \number\day instead of \ifcase construct	234
magyar-1.4b	
\captionsmagyar: Added \glossaryname	234
magyar-1.4c	
General: Make sure that the grave accent has catcode 12 <i>before</i> it is made \active	246
magyar-1.4d	
General: Corrected checksum	232
The \else clause got outside of the \if statement, breaking the Hungarian support	233
magyar-1.4f	
\hun@tempadef: Added \def\safe@activesfalse as a fix for PR3426	241
magyar-1.4g	
General: Further change to make it work when neither \l@magyar nor \l@hugarian are defined	233
magyar-1.4h	
\captionsmagyar: Inserted translation for Glossary	234
magyar-1.4i	
\fnun@table: Use \nobreakspace instead of tilde	235
magyar-1.4j	
General: Added missing comment characters in the redefinitions of \ps@headings to prevent spurious spaces	232
ngermab-v2.6n	
\captionснаustrian: Corrected typo \captionnsgerman	86
ngermanb-2.6f	
General: Renamed from germanb.ldf; language names changed from german and austrian to ngerman and наustrian.	86
ngermanb-2.6j	
\nоextrаснаustrian: Deactivate shorthands outside of German	87
ngermanb-2.6k	
\captionснаustrian: Added \glossaryname	86

\noextrasnaustrian: Now use \providehyphenmins to provide a default value	87
ngermanb-2.6m	
\noextrasnaustrian: Turn frenchspacing on, as in <code>german.sty</code>	87
norks-2.0h	
\captionesnynorsk: Changed \ccname and \alsoname	220
norsk-1.0a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	219
norsk-1.0c	
General: Removed use of \@ifundefined	219
norsk-1.1a	
General: Added a warning when no hyphenation patterns were loaded.	219
Brought up-to-date with babel 3.2a	219
\captionesnorsk: Added \seename, \alsoname and \prefacename	220
\captionesnynorsk: Added \seename, \alsoname and \prefacename	220
norsk-1.1b	
\captionesnorsk: \headpagename should be \pagename	220
\captionesnynorsk: \headpagename should be \pagename	220
norsk-1.1c	
General: Added a couple of translations (from Per Norman Oma, TeX@itk.unit.no)	219
norsk-1.2a	
General: Update for L ^A T _E X 2 _ε	219
norsk-1.2d	
General: Now use \@nopatterns to produce the warning	219
Removed the use of \filedate and moved identification after the loading of	
<code>babel.def</code>	219
norsk-1.2f	
\captionesnorsk: Added \proofname for AMS-L ^A T _E X	220
\norskhyphenmins: Added setting of hyphenmin parameters	219
norsk-1.2g	
\captionesnorsk: Replaced ‘Proof’ by its translation	220
\captionesnynorsk: Replaced ‘Proof’ by its translation	220
norsk-1.2h	
General: Moved the definition of \atcatcode right to the beginning.	219
Now use \ldf@finish to wrap up	222
Now use \LdfInit to perform initial checks	219
Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X	219
norsk-1.2i	
\datenorsk: use \def instead of \edef	220
Use \edef to define \today to save memory	220
norsk-2.0a	
General: Describe the use of double quote as active character	219
Made double quote character active	221
\norskhyphenmins: Changed setting of hyphenmin parameters to 2 2	219
norsk-2.0b	
General: added the french double quotes	222
Copied the coding for “f from <code>germanb.dtx</code> version 2.6g	222
norsk-2.0c	
General: Deactivate shorthands outside of Norsk	221
norsk-2.0d	
General: Shorthands are the same for both spelling variants, no need to use	
\CurrentOption	221
Use \bbl@allowhyphens in “-	222
norsk-2.0e	
\captionesnorsk: Added \glossaryname	220
\captionesnynorsk: Added \glossaryname	220
\norskhyphenmins: Now use \providehyphenmins to provide a default value	219
norsk-2.0g	
\captionesnorsk: Replaced ‘Glossary’ by its translation	220
\captionesnynorsk: Replaced ‘Glossary’ by its translation	220

polish-1.0a	
\textpl: Initially execute 'textpl	270
polish-1.1c	
General: Now use \@nopatterns to produce the warning	268
Removed the use of \filedate and moved identification after the loading of	
babel.def	268
polish-1.1d	
General: The dqmacro for C used \'c	271
polish-1.2a	
General: Don't modify \rm and friends for L ^A T _E X 2 _ε , take \selectfont instead	271
polish-1.2b	
\captionspolish: Added \proofname for AMS-L ^A T _E X	268
polish-1.2d	
General: Now use \ldf@finish to wrap up	272
Now use \LdfInit to perform initial checks	268
Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X, moved the definition of \atcatcode right to the beginning.	268
\Eob: Use the constructed version of the characters only in OT1; use proper characters in T1.	270
\skb: \skb is meant to be used in horizontal mode; so leave vertical mode if necessary	270
\sob: This macro is meant to be used in horizontal mode; so leave vertical mode if necessary	269
\spb: \spb is meant to be used in horizontal mode; so leave vertical mode if necessary	270
polish-1.2e	
General: Removed empty groups after double quote and guillemot characters	272
polish-1.2f	
\captionspolish: Added translation for Proof and changed translation of Contents	268
\datepolish: use \def instead of \edef	269
Use \edef to define \today to save memory	269
\mdqoff: Now use \shorthandon and \shorthandoff	272
polish-1.2h	
\noextrapolish: Deactivate shorthands outside of Polish	269
polish-1.2i	
\captionspolish: \bibname and \refname were swapped	268
Added \glossaryname	268
\datepolish: A missing comment character caused an unwanted space character in the output	269
polish-1.2j	
\polishzx: Added support for two notationstyles for kropka and accented z	271
polish-1.2k	
\polishzx: Fixed a typo	272
polish-1.2l	
General: Changed closing quote	272
portuges-1.0a	
General: Renamed babel.sty in babel.com	153
portuges-1.0b	
General: Removed use of cs@ifundefined	153
portuges-1.1	
General: Added a warning when no hyphenation patterns were loaded.	153
Brought up-to-date with babel 3.2a	153
\captionsportuges: \headpagename should be \pagename	154
Added \seename, \alsoname and \prefacename	154
portuges-1.2	
General: Update for L ^A T _E X 2 _ε	153
portuges-1.2d	
General: Now use \@nopatterns to produce the warning	153
Removed the use of \filedate and moved identification after the loading of	
babel.def	153

portuges-1.2e	
\captionsportuges: Added a few missing translations	154
portuges-1.2g	
General: Enhanced support for brasilian	153
\captionsbrazil: The captions for brasilian and portuguese are different now	155
\extrasportuges: Added the definition of some " shorthands	155
\ord: Added macro	156
\orda: Added macro	156
\portugeshyphenmins: Added setting of hyphenmin values	155
\ra: Added macro	156
\ro: Added macro	156
portuges-1.2h	
\captionsportuges: Added \proofname for AMS-L ^A T _E X	154
portuges-1.2i	
\captionsbrazil: Added \proofname for AMS-L ^A T _E X	155
\captionsportuges: Substituted ‘Prova’ for ‘Proof’	154
portuges-1.2j	
General: Moved the definition of \atcatcode right to the beginning.	153
Now use \LdfInit to perform initial checks	153
Now use \ldf@finish to wrap up	156
Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X	153
portuges-1.2k	
\datebrazil: use \def instead of \edef	155
Use \edef to define \today to save memory	155
\dateportuges: use \def instead of \edef	154
Use \edef to define \today to save memory	154
\noextrasportuges: Removed empty groups after guillemot characters	156
portuges-1.2m	
\captionsbrazil: Added \glossaryname	155
\captionsportuges: Added \glossaryname	154
\noextrasportuges: Deactivate shorthands outside of Basque	155
\portugeshyphenmins: Now use \providehyphenmins to provide a default value	155
portuges-1.2n	
\datebrazil: Removed spurious space after dezembro	155
\dateportuges: Removed spurious space after Dezembro	154
portuges-1.2o	
\portugeshyphenmins: Set \righthyphenmin to 3 if not provided by the pattern file.	155
portuges-1.2p	
\captionsportuges: Substituted ‘Glossário’ for ‘Glossary’	154
portuges-1.2q	
\captionsbrazil: Substituted ‘Glossário’ for ‘Glossary’	155
romanian-1.0a	
General: Renamed babel.sty in babel.com	206
romanian-1.0b	
General: Removed use of \@ifundefined	206
romanian-1.1	
General: Added a warning when no hyphenation patterns were loaded.	206
Brought up-to-date with babel 3.2a	206
\captionsromanian: \headpagename should be \pagename	206
Added \seename, \alsoname and \prefacename	206
Translation errors found by Robert Juhasz fixed	206
\dateromanian: Translation errors found by Robert Juhasz fixed	206
romanian-1.2	
General: Update for LaTeX2e	206
romanian-1.2d	
General: Now use \@nopatterns to produce the warning	206
Removed the use of \filedate and moved identification after the loading of babel.def	206
romanian-1.2e	
General: Updated for babel release 3.5	206

romanian-1.2f	
\captionsromanian: Added \proofname for AMS-L ^A T _E X	206
romanian-1.2g	
\captionsromanian: Added translation of ‘Proof’	206
romanian-1.2h	
General: Now use \ldf@finish to wrap up	207
Now use \LdfInit to perform initial checks	206
Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X, moved the definition of \atcatcode right to the beginning.	206
romanian-1.2i	
\dateromanian: use \def instead of \edef	206
Use \edef to define \today to save memory	206
romanian-1.2k	
\captionsromanian: Added \glossaryname	206
romanian-1.2l	
\captionsromanian: Added translation for Glossary	206
russianb-1.1a	
\extrarussian: Use \ddot instead of \@MATHUMLAUT	299
use \russianhyphenmins to store the correct values	301
Use the new mechanism for dealing with active characters	297
\russian@sh@?@: Use new \DefineActiveNoArg	298
Use the more general mechanism of \declare@shorthand	298
\system@sh@;@: Added system level shorthands	299
russianb-1.1b	
\extrarussian: Added switch to LWN encoding	296
\russian@sh@?@: Updated to reflect the latest french definitions	298
\verbatim@font: Added changing of \verbatim@font	296
russianb-1.1c	
General: Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X	291
russianb-1.1d	
General: Moved the definition of \atcatcode right to the beginning.	291
Now use \ldf@finish to wrap up	302
Now use \LdfInit to perform initial checks	291
russianb-1.1e	
General: Added closing brace to second argument of \LdfInit	291
russianb-1.1f	
General: Added definitions of Cyrillic emdash stuff and thinspace	291
\extrarussian: Add commands for switch on/off doublequote activeness. Borrowed from german.	300
Add macro for thinspace between initials	300
Added a hook to insert space or not before ‘double punctuation’ (from frenchb).	297, 298
Rearranging of cyrillic emdash stuff	300
\FDPoff: One more chance to avoid spaces before double punctuation	299
\russian@sh@?@: changed to kern.1em (space bit thinner)	298
Added a hook to insert space or not before ‘double punctuation’ (from frenchb).	298
russianb-1.1k	
General: replaced all \penalty\@M with \nobreak	291
russianb-1.1l	
General: Made not using inputenc a warning instead of an error	294
russianb-1.1m	
\captionsrussian: Added \glossaryname	296
\extrarussian: Now use \providehyphenmins to provide a default value	301
russianb-1.1n	
General: As this code generates a textfont 7 error it is commented out for now.	294
russianb-1.1o	
\latintext: \latintext is already defined by the core of babel	294
\textlatin: \textlatin already defined by the core of babel	295
russianb-1.1q	
General: Change definition of \th only for this language	302

russianb-1.1r	
<code>\extrarussian</code> : Removed the commanet character before the next code line, see	
R3669	300
samin-1.0b	
<code>\captionssamin</code> : Added <code>\glossaryname</code>	227
<code>\saminhyphenmins</code> : use <code>\providehyphenmins</code>	227
samin-1.0c	
<code>\captionssamin</code> : Provided the translation for glossary	227
scottish-1.0b	
General: Corrected typos (PR1652)	96
scottish-1.0c	
<code>\captionsscottish</code> : Added <code>\proofname</code> for AMS- \LaTeX	96
scottish-1.0d	
General: Now use <code>\ldf@finish</code> to wrap up	97
Now use <code>\LdfInit</code> to perform initial checks	96
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX , moved the definition of <code>\atcatcode</code> right to the beginning.	96
scottish-1.0e	
<code>\datescottish</code> : use <code>\def</code> instead of <code>\edef</code>	96
Use <code>\edef</code> to define <code>\today</code> to save memory	96
scottish-1.0g	
<code>\captionsscottish</code> : Added <code>\glossaryname</code>	96
serbian-1.0b	
General: Added suggestions for shorthands and so on from Jankovic Slobodan	273
<code>\noextrasserbian</code> : Introduced the active "	274
serbian-1.0c	
<code>\noextrasserbian</code> : Deactivate shorthands ouside of Serbian	274
serbian-1.0d	
<code>\captionsserbian</code> : Added <code>\glossaryname</code>	273
<code>\noextrasserbian</code> : Changed definition of "-" to be the same as for other languages	274
slovak-1.0	
General: First version	276
slovak-1.2	
General: Update for \LaTeX 2 ϵ	276
slovak-1.2b	
General: Added setting of left- and righthyphenmin	279
slovak-1.2d	
General: Now use <code>\@nopatterns</code> to produce the warning	277
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code>	276
slovak-1.2e	
General: Now use <code>\slovakhyphenmins</code>	279
<code>\noextrasslovak</code> : Use \LaTeX 's <code>\v</code> accent command	279
slovak-1.2g	
<code>\captionsslovak</code> : Added <code>\proofname</code> for AMS- \LaTeX	278
slovak-1.2i	
General: Now use <code>\ldf@finish</code> to wrap up	288
Now use <code>\LdfInit</code> to perform initial checks	277
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX , moved the definition of <code>\atcatcode</code> right to the beginning.	276
slovak-1.2j	
<code>\dateslovak</code> : use <code>\def</code> instead of <code>\edef</code>	279
Use <code>\edef</code> to define <code>\today</code> to save memory	279
slovak-1.2k	
<code>\captionsslovak</code> : Repaired a few spelling mistakes	278
<code>\dateslovak</code> : Repaired a spelling mistake	279
slovak-1.2l	
General: Now use <code>\providehyphenmins</code> to provide a default value	279
<code>\captionsslovak</code> : Added <code>\glossaryname</code>	278
slovak-1.3a	
General: Added contributed shorthand definitions	276

\noextrasslovak: Make three characters available for shorthands	279
slovak-3.0	
General: Changed default \righthyphenmin to 3 characters.	279
Implemented the functionality of \LaTeX 's slovak.sty. The version number was bumped to 3.0 to minimize confusion by being higher than the last version of \LaTeX .	276
\captionsslovak: Updated some translations. Former translations were: 'Úvod' for \prefacename, 'Referencie' for \refname, 'Index' for \indexname, 'Obrázok' for \figurename, 'Prílohy' for \enclname, 'CC' for \ccname, 'Komu' for \headtoname, 'Strana' for \pagename	278
\noextrasslovak: now use \bbl@frenchspacing and \bbl@nonfrenchspacing	279
slovak-3.1	
\cs@emdash: ensure correct catcode for the saved hyphen	283
\cs@splitattr: attribute added	286
\noextrasslovak: move \languageshorthands here, so that the language group is always initialized correctly	279
\splithyphens: activate with split hyphens and deactivate with standard hyphens, not vice versa	286
slovene-1.0a	
General: Renamed babel.sty in babel.com	289
slovene-1.0b	
General: Removed use of \@ifundefined	289
slovene-1.1	
General: Added a warning when no hyphenation patterns were loaded.	289
Brought up-to-date with babel 3.2a	289
\captionsslovene: \headpagename should be \pagename	289
Added \seename, \alsoname and \prefacename	289
slovene-1.2	
General: Update for \LaTeX 2 ϵ	289
slovene-1.2b	
\captionsslovene: Added extra translations from Josef Leydold, leydold@statix2.wu-wien.ac.at	289
slovene-1.2d	
General: Now use \@nopatterns to produce the warning	289
Removed the use of \filedate and moved identification after the loading of babel.def	289
slovene-1.2f	
\noextrasslovene: Introduced the active "	290
slovene-1.2g	
\captionsslovene: Added \proofname for AMS- \LaTeX	289
slovene-1.2h	
\captionsslovene: Added translation of 'Proof'	289
slovene-1.2i	
General: Now use \ldf@finish to wrap up	290
Now use \LdfInit to perform initial checks	289
Replaced \undefined with \@undefined and \empty with \@empty for consistency with \LaTeX , moved the definition of \atcatcode right to the beginning.	289
Replaced 'Slovanian' with correct 'Slovenian'	289
\noextrasslovene: removed shorthand for "L as it is not needed for slovenian	290
slovene-1.2j	
\dateslovene: use \def instead of \edef	290
Use \edef to define \today to save memory	290
\noextrasslovene: Removed empty groups after double quote and guillemot characters	290
slovene-1.2l	
\noextrasslovene: Deactivate shorthands outside of Slovene	290
slovene-1.2m	
\captionsslovene: Added \glossaryname	289
spanish 5.0a	
General: Reimplemented in full, which some parts rewritten from scratch. Added the es- mechanism and the mexico option. Many bug fixes.	161

spanish 5.0d	
General: Fixed two bugs: misplaced subscripts with <code>\lim</code> and the like; problem with <code>\roman</code> and <code>hyperref</code> .	161
spanish 5.0e	
General: Two acutes in a row should be turned into a double right quote	171
spanish 5.0g	
General: Fixed bad kerning before two acutes	171
spanish 5.0h	
General: Removed unnecessary <code>\strings</code> with two acutes. Added <code>es-noenumerate</code> , <code>es-noitemize</code> .	161
swedish-1.0a	
General: Renamed <code>babel.sty</code> in <code>babel.com</code>	223
swedish-1.0b	
<code>\captionsswedish</code> : added definition for <code>\enclname</code>	223
made definition of <code>\refname</code> pluralis	223
removed type in definition of <code>\contentsname</code>	223
swedish-1.0c	
General: Removed use of <code>\@ifundefined</code>	223
swedish-1.1	
General: Added a warning when no hyphenation patterns were loaded.	223
Brought up-to-date with <code>babel 3.2a</code>	223
<code>\captionsswedish</code> : <code>\headpagename</code> should be <code>\pagename</code>	223
Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	223
swedish-1.1b	
<code>\captionsswedish</code> : Added translations	223
swedish-1.2	
General: Update for LaTeX2e	223
swedish-1.3d	
General: Now use <code>\@nopatterns</code> to produce the warning	223
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code>	223
<code>\captionsswedish</code> : Changed <code>\aa</code> to <code>\csname aa\endcsname</code> , to make <code>\uppercase</code> do the right thing	223
swedish-1.3e	
General: Update for release 3.5	223
<code>\captionsswedish</code> : Changed <code>\alsoname</code> from ‘se också’	224
<code>\extrasswedish</code> : Added <code>\bbl@frenchspacing</code>	224
<code>\noextrasswedish</code> : Added <code>\bbl@nonfrenchspacing</code>	224
<code>\swedishhyphenmins</code> : use <code>\swedishhyphenmins</code> to store the correct values	224
swedish-1.3f	
<code>\captionsswedish</code> : Added <code>\proofname</code> for AMS- \LaTeX	223
swedish-1.3g	
<code>\captionsswedish</code> : Replaced ‘Proof’ by its translation	224
swedish-2.0	
General: Introduced active double quote	223
<code>\noextrasswedish</code> : Added active double quote character	225
swedish-2.1	
General: Now use <code>\ldf@finish</code> to wrap up	226
Now use <code>\LdfInit</code> to perform initial checks	223
Replaced <code>\undefined</code> with <code>\@undefined</code> and <code>\empty</code> with <code>\@empty</code> for consistency with \LaTeX , moved the definition of <code>\atcatcode</code> right to the beginning.	223
swedish-2.2	
<code>\dateswedish</code> : use <code>\def</code> instead of <code>\edef</code>	224
Use <code>\edef</code> to define <code>\today</code> to save memory	224
swedish-2.2b	
<code>\noextrasswedish</code> : Deactivate shorthands outside of Swedish	225
swedish-2.3a	
General: added <code>\allowhyphens</code>	225
changed definition of “=, \- and ”	225
<code>\captionsswedish</code> : Added full stop after “Bil”	224
<code>\datesdmy</code> : Command added	224
<code>\datesymd</code> : Command added	224

swedish-2.3b	
\captionsswedish: Added \glossaryname	224
swedish-2.3c	
\captionsswedish: Provided translation for Glossary	224
swedish-2.3d	
General: Fixed a \changes entry	223
turkish-1.1	
\captionsturkish: \headpagename should be \pagename	331
turkish-1.2	
General: Update for L ^A T _E X 2 _ε	331
turkish-1.2b	
\captionsturkish: Added braces behind \i in strings	331
\dateturkish: Added braces behind \i in strings	331
turkish-1.2c	
General: Now use \@nopatterns to produce the warning	331
Removed the use of \filedate and moved identification after the loading of	
babel.def	331
turkish-1.2d	
\dateturkish: removed extra closing brace, \mont should be \month	331
\turkish@sh@: Added missing \def	332
turkish-1.2e	
\extrasturkish: Completely rewrote macro	332
\noextrasturkish: now use \bbl@frenchspacing and \bbl@nonfrenchspacing	332
\turkish@sh@: Use the new mechanism of \declare@shorthand	332
turkish-1.2f	
\captionsturkish: Added \proofname for AMS-L ^A T _E X	331
turkish-1.2h	
General: Replaced \undefined with \@undefined and \empty with \@empty for	
consistency with L ^A T _E X	331
turkish-1.2i	
General: Moved the definition of \atcatcode right to the beginning.	331
Now use \ldf@finish to wrap up	333
ow use \LdfInit to perform initial checks	331
turkish-1.2j	
\captionsturkish: Added and modified translations	331
\dateturkish: use \def instead of \edef	331
Use \edef to define \today to save memory	331
turkish-1.2k	
\captionsturkish: Incorporated some more corrections	331
turkish-1.2l	
\dateturkish: removed dot from the date format	331
turkish-1.2m	
\captionsturkish: Added \glossaryname	331
ukraine-1.1d	
\captionssukrainian: replace \CYRUKRI with \CYRII in \authornam	318
ukraine-1.1e	
General: replaced all \penalty\@M with \nobreak	314
ukraine-1.1f	
General: Made not using inputenc a warning instead of an error	317
ukraine-1.1g	
\captionssukrainian: Added \glossaryname	318
\extrasukrainian: Now use \providehyphenmins to provide a default value	323
ukraine-1.1h	
\captionssukrainian: Added translation for ‘Glossary’	318
ukraine-1.1i	
General: As this code generates a textfont 7 error it is commented out for now.	317
ukraine-1.1j	
\latintext: \latintext is already defined by the core of babel	317
\textlatin: \latintext is already defined by the core of babel	318
ukraine-1.1k	
General: Change definition of \th only for this language	325

usorbian-0.1	
General: First version	328
usorbian-0.1b	
General: Made it possible to run through L ^A T _E X; added \MF and removed extra \endmacro	328
usorbian-0.1c	
\captionusorbian: Removed two typos (Kapitel and Dodatki)	328
usorbian-1.0a	
General: Removed stuff that has been moved to babel.def	329
usorbian-1.0b	
\captionusorbian: Added \proofname for AMS-L ^A T _E X	328
usorbian-1.0c	
General: Now use \bbl@disc	330
usorbian-1.0d	
General: Replaced \undefined with \@undefined and \empty with \@empty for consistency with L ^A T _E X	328
usorbian-1.0e	
General: Moved the definition of \atcatcode right to the beginning.	328
Now use \ldf@finish to wrap up	330
Now use \LdfInit to perform initial checks	328
usorbian-1.0f	
General: Removed empty groups after double quote and guillemot characters	330
usorbian-1.0g	
\ck: Now use \shorthandon and \shorthandoff	330
\newdateusorbian: use \def instead of \edef	328
Use \edef to define \today to save memory	328
\olddateusorbian: use \def instead of \edef	329
Use \edef to define \today to save memory	329
usorbian-1.0h	
\extrasusorbian: Deactivate shorthands outside of Upper Sorbian	329
usorbian-1.0i	
\captionusorbian: Added \glossaryname	328
\extrasusorbian: Now use \providehyphenmins to provide a default value	329
usorbian-1.0j	
General: Check for the option lowersorbian	328
Make this work for more than one option name	329, 330
This file can be loaded under more than one name.	328
\captionusorbian: Make this work for more than one option name	328
\extrasusorbian: Make this work for more than one option name	329
\newdateusorbian: Make this work for more than one option name	328
\olddateusorbian: Make this work for more than one option name	329
usorbian-1.0k	
\extrasusorbian: Make sure the caret has the right \catcdoe	329
v2.0	
General: \parindentFFN not changed if already defined (required by JA for cah- gut.cls).	126
Added warning for OT1 encoding.	135
AtBeginDocument, save again the definitions of the ‘list’ and ‘itemize’ envi- ronments and the values of labelitems. As of frenchb v.1.6, ‘ORI’ values were set when reading frenchb.ldf, later changes were ignored.	135
Footnotes are now printed by default ‘à la française’ for the whole document.	125
New command \frenchbsetup added for global customisation.	128
\bsc: \hbox dropped, replaced by \kernOpt.	119
\captionsfrench: ‘Fig.’ changed to ‘Figure’ and ‘Tab.’ to ‘Table’.	121
Set \CaptionSeparator in \extrasfrench now instead of \captionsfrench because it has to be reset when leaving French.	121
\datefrench: 2 ‘\relax’ added in \today’s definition.	117
\FBtextellipsis: Added special case for LY1 encoding, see bug report from Bruno Voisin (2004/05/18).	127
\nombre: \nombre requires now numprint.sty.	121

v2.0b	General: Footnotes: Just do nothing (except warning) when the bigfoot package is loaded.	125
v2.0c	General: <code>\ThinSpaceInFrenchNumbers</code> added for compatibility with frenchb-1.x.	121
	Option <code>ThinSpaceInFrenchNumbers</code> added.	128
	There is no need to define here numprint's command <code>\npstylefrench</code> , it will be redefined 'AtBeginDocument' by <code>\FBprocess@options</code>	121
v2.0d	General: Options <code>og</code> and <code>fg</code> changed: limit the definition to French so that quote characters can be used in German.	128
v2.0e	General: New option: <code>StandardLists</code>	128
v2.0f	General: <code>StandardLayout</code> option had no effect on lists. Test moved to <code>\FBprocess@options</code>	128
	Two typos corrected in option <code>StandardLists</code> : <code>[false]</code> → <code>[true]</code> and <code>StandardLayout</code> → <code>StandardLists</code>	128
v2.0g	General: Revert previous change to <code>StandardLayout</code> . This option must set the three flags <code>\FBReduceListSpacingfalse</code> , <code>\FBCompactItemizefalse</code> , and <code>\FBStandardItemLabeltrue</code> instead of <code>\FBStandardListstrue</code> , so that later options can still change their value before executing <code>\FBprocess@options</code> . Same thing for option <code>StandardLists</code>	128
	<code>\StandardLayout</code> : Flag <code>\iffBStandardLayout</code> not checked by <code>\FBprocess@options</code> , low-level flags have to be set one by one.	127
v2.1a	General: Command <code>\fup</code> added to produce better superscripts than <code>\textsuperscript</code>	117
	New option: <code>FrenchSuperscripts</code> to define <code>\up</code> as <code>\fup</code> or as <code>\textsuperscript</code>	128
	New option: <code>LowercaseSuperscripts</code>	128
	<code>\datefrench</code> : <code>\today</code> changed (correction in 2.0 was wrong: <code>\today</code> was printed without spaces in toc).	117
v2.1b	General: Disable some commands in bookmarks.	135
	<code>\fup</code> : Command <code>\fup</code> changed to use real superscripts from fourier v. 1.6.	117
v2.1c	General: Added commands <code>\Nos</code> and <code>\nos</code>	119
	<code>\degres</code> : Provide a temporary definition (hyperref safe) of <code>\degres</code> in case it has to be expanded in the preamble (by beamer's <code>\title</code> command for instance).	120
	<code>\up</code> : Provide a temporary definition (hyperref safe) of <code>\up</code> in case it has to be expanded in the preamble (by beamer's <code>\title</code> command for instance).	117
v2.1d	General: Argument of <code>\ProvidesLanguage</code> changed above from 'french' to 'frenchb' (otherwise <code>\listfiles</code> prints no date/version information). The real name of current language (french) as to be corrected before calling <code>\LdfInit</code>	112
	Avoid warning " <code>\end</code> occurred when <code>\ifx ... incomplete</code> " with LaTeX-2.09.	112
v2.2a	General: The global layout of the document is no longer changed when frenchb is not the last option of babel (<code>\bbl@main@language</code>). Suggested by Ulrike Fischer.	128
	Values of flags <code>\iffBReduceListSpacing</code> , <code>\iffBCompactItemize</code> , <code>\iffBStandardItemLabels</code> , <code>\iffBIndentFirst</code> , <code>\iffBFrenchFootnotes</code> , <code>\iffBAutoSpaceFootnotes</code> changed: default now means 'StandardLayout', it will be changed to 'FrenchLayout' AtEndOfPackage only if french is <code>\bbl@main@language</code>	128
	When frenchb is babel's last option, French becomes the document's main language, so <code>GlobalLayoutFrench</code> applies.	128
	<code>\fup</code> : <code>\newif</code> and <code>\newdimen</code> moved before <code>\ifLaTeXe</code> to avoid an error with plainTeX.	117

v2.3a	
General: <code>\NoAutoSpaceBeforeFDP</code> and <code>\AutoSpaceBeforeFDP</code> now set the flag <code>\iffBAutoSpacePunctuation</code> accordingly (LaTeX only).	114
In LaTeX, frenchb no longer adds spaces before ‘double punctuation’ characters in computer code. Suggested by Yannis Haralambous.	115
New option: <code>OriginalTypewriter</code> . Now frenchb switches to <code>\noautospace@beforeFDP</code> when a tt-font is in use. When <code>OriginalTypewriter</code> is set to true, frenchb behaves as in pre-2.3 versions.	128
<code>\fup</code> : <code>\lowercase</code> changed to <code>\MakeLowercase</code> as the former doesn’t work for non ASCII characters in encodings like applemac, utf-8,...	117
v2.3b	
General: New commands <code>\dotFFN</code> and <code>\kernFFN</code> for more flexibility (suggested by JA).	126
v2.3c	
General: Commands <code>\ttfamily</code> , <code>\rmfamily</code> and <code>\sffamily</code> have to be robust. Bug introduced in 2.3a, pointed out by Manuel Pégourié-Gonnard.	115
v2.3d	
<code>\bbl@nonfrenchindent</code> : Bug correction: previous versions of frenchb set the flag <code>\if@afterindent</code> to false outside French which is correct for English but wrong for some languages like Spanish. Pointed out by Juan José Torrens.	125
<code>\frenchbsetup</code> : Warning added to <code>\GlobalLayoutFrench</code> when French is not the main language.	129
v3.8a	
<code>\lower@umlaut</code> : Use <code>\leavevmode\bgroup</code> to prevent problems when this command occurs in vertical mode.	50
<code>\umlauthigh</code> : Use <code>\leavevmode\bgroup</code> to prevent problems when this command occurs in vertical mode.	49
v3.8d	
<code>\@notshorthand</code> : Error message added	40
welsh-1.0b	
<code>\datewelsh</code> : use <code>\def</code> instead of <code>\edef</code>	92
Use <code>\edef</code> to define <code>\today</code> to save memory	92
welsh-1.0c	
<code>\captionswelsh</code> : Added <code>\glossaryname</code>	92
<code>\welshhyphenmins</code> : Now use <code>\providehyphenmins</code> to provide a default value	92
welsh-1.0d	
<code>\captionswelsh</code> : Provided the translation for Glossary	92
<code>\datewelsh</code> : removed ‘a viz’ from the definition of <code>\today</code>	92