# Babel, a multilingual package for use with $\mathrm{LT}_{\mathrm{E}} \mathrm{X}^{\prime}$ 's standard document classes* 

Johannes Braams<br>Kersengaarde 33<br>2723 BP Zoetermeer<br>The Netherlands<br>babel@braams.cistron.nl

Printed May 28, 2005


#### Abstract

The standard distribution of $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ version 3 to provide an environment in which documents can be typeset in a non-American language, or in more than one language.


## Contents

1 The user interface 5
1.1 Languages supported by
Babel . . . . . . . . . . . . 6
1.2 Workarounds . . . . . . . 7

2 Changes for $\mathrm{HT}_{\mathrm{E}} \mathbf{2} 2 \boldsymbol{\varepsilon} \quad 7$
3 Changes in Babel version 3.78
4 Changes in Babel version 3.69
5 Changes in Babel version $3.5 \quad 10 \quad 7 \quad$ Compatibility with german. sty 14
6 The interface between the
core of babel and the language definition files11
6.1 Support for active char- acters ..... 13
6.2 Support for saving macro definitions ..... 13
6.3 Support for extending macros ..... 14
6.4 Macros common to anumber of languages . . . 14

[^0]9 Compatibility with the
french package ..... 15
10 Identification ..... 15
11 The Package File ..... 16
11.1 Language options ..... 16
12 The Kernel of Babel ..... 20
12.1 Encoding issues (part 1) ..... 21
12.2 Multiple languages ..... 21
12.3 Support for active char- acters ..... 37
12.4 Shorthands ..... 38
12.5 Language attributes ..... 49
12.6 Support for saving macro definitions ..... 52
12.7 Support for extending macros ..... 53
12.8 Macros common to a number of languages ..... 54
12.9 Making glyphs available ..... 55
12.10Quotation marks ..... 55
12.11Letters ..... 56
12.12Shorthands for quotation marks ..... 57
12.13Umlauts and trema's ..... 59
12.14The redefinition of the style commands ..... 60
12.14.1 Redefinition of macros ..... 61
12.15Cross referencing macros . ..... 65
12.16marks ..... 70
12.17Encoding issues (part 2) ..... 71
12.18Preventing clashes withother packages71
12.18.1 ifthen ..... 71
12.18.2 varioref ..... 72
12.18 .3 hhline ..... 73
12.18.4 General ..... 73
13 Local Language Configura- tion ..... 74
14 Driver files for the docu- mented source code ..... 76
15 Conclusion
15 Conclusion ..... 80 ..... 80
16 Acknowledgements
16 Acknowledgements ..... 80 ..... 80
17 The Esperanto language ..... 81
18 The Interlingua language ..... 84
19 The Dutch language ..... 86
20 The English language ..... 91
21 The German language ..... 96
22 The German language - new orthography ..... 101
23 The Breton language ..... 105
24 The Welsh language ..... 109
25 The Irish language ..... 111
26 The Scottish language ..... 113
27 The Greek language ..... 115
27.1 Typing conventions ..... 115
27.2 Greek numbering ..... 115
28 The French language ..... 125
29 TEXnical details ..... 135
29.1 Initial setup ..... 135
29.2 Caption names ..... 138
29.3 Punctuation ..... 140
29.4 French quotation marks ..... 142
29.5 French lists ..... 144
29.6 French indentation of sections ..... 147
29.7 Formatting footnotes ..... 147
29.8 Formatting numbers ..... 148
29.9 Dots. ..... 152
29.10Global layout ..... 152
29.11Extra utilities ..... 153
29.12Date and clean up ..... 157
30 The Italian language ..... 158
30.1 Support for etymological hyphenation ..... 161
30.2 Facilities required by the ISO 31/XI regulations . ..... 163
30.3 Accents ..... 164
30.4 Caporali or French dou- ble quotes ..... 165
30.5 Finishing commands ..... 168
31 The Latin language ..... 169
32 Latin shortcuts ..... 173
33 Etymological hyphenation ..... 175
34 The Portuguese language ..... 178
35 The Spanish language ..... 183
35.1 The Code ..... 185
36 The Catalan language ..... 206
37 The Galician language ..... 214
38 The Basque language ..... 219
39 The Romanian language ..... 223
40 The Danish language ..... 225
41 The Icelandic language ..... 228
41.1 Overview ..... 228
41.2 $\mathrm{T}_{\mathrm{E} X n i c a l ~ d e t a i l s ~}$ ..... 229
41.3 Captionnames and date ..... 231
41.4 Icelandic quotation marks ..... 231
41.5 Old Icelandic ..... 232
41.6 Formatting numbers ..... 233
41.7 Extra utilities ..... 235
42 The Norwegian language ..... 238
43 The Swedish language ..... 243
44 The North Sami language ..... 248
44.1 The code of samin. dtx ..... 248
45 The Finnish language ..... 251
47 The Estonian language ..... 274
47.1 Implementation ..... 275
48 The Croatian language ..... 279
49 The Czech language ..... 281
50 The Polish language ..... 284
51 The Serbocroatian language 290
52 The Slovak language ..... 294
53 The Slovenian language ..... 299
54 The Russian language ..... 302
55 The Bulgarian language ..... 317
56 The Ukrainian language ..... 330
57 The Lower Sorbian lan- guage ..... 345
58 The Upper Sorbian lan- guage ..... 347
59 The Turkish language ..... 351
60 The Hebrew language ..... 354
60.1 Acknowledgement ..... 354
60.2 The Docstrip modules ..... 355
60.3 Hebrew language defini- tions ..... 355
60.3.1 Hebrew numerals ..... 359
60.4 Right to left support ..... 367
60.4.1 Switching fromLR to RL modeand back . . . . . 367
60.4.2 Counters ..... 371
60.4.3 Preserving logos ..... 372
60.4.4 List environments ..... 372
60.4.5 Tables of moving stuff ..... 373
60.4.6 Two-column mode ..... 378
60.4.7 Footnotes ..... 379
60.4.8 Headings and two-side support . 379
60.4.9 Postscript Porblems383
60.4.10 Miscellaneous in-ternal $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ macros384
60.4.11 Bibliography and citations ..... 386
60.4.12 Additional bidi- rectional commands388
60.5 Hebrew calendar ..... 389
60.5.1 Introduction ..... 390
60.5.2 Registers, Com- mands, Format- ting Macros ..... 391
60.5.3 Auxiliary Macros ..... 394
60.5.4 Gregorian Part ..... 394
60.5.5 Hebrew Part ..... 396
61 Hebrew input encodings ..... 400
61.1 Default definitions for characters ..... 402
61.2 The SI-960 encoding ..... 403
61.3 The ISO 8859-8 encod-ing and the MS Windowscp1255 encoding . . . . . 403
61.4 The IBM code page 862 ..... 406
62 Hebrew font encodings ..... 408
62.1 THIS SECTION IS OUTOF DATE. UPDATEDOCS TO MATCH HE8ENCODING408
62.2 The DOCSTRIP modules ..... 409
62.3 The LHEencoding defini- tion file ..... 411
62.4 The font definition files (in LHE encoding) ..... 412
62.4.1 Hebrew default font 412
62.4.2 Hebrew sans-seriffont . . . . . . . . 413
62.4.3 Hebrew type- writer font . . . . 413
62.4.5 Hebrew shalom fonts ..... 415
62.4.6 Hebrew frank- ruehl font ..... 415
62.4.7 Hebrew carmel font 416
62.4.8 Hebrew redis font . 417
62.5 The HE8encoding defini- tion file ..... 418
62.5.1 CHECK HEREFOR HE8 UP-DATES . . . . . . 418
62.6 The font definition files (in HE8 encoding) ..... 420
62.6.1 Hebrew default font 420
62.6.2 Hebrew sans-seriffont420
62.6.3 Hebrew type- writer font . . . . 421
62.6.4 8Bit OmegaHe- brew font ..... 421
62.6.5 8Bit Aharoni font ..... 422
62.6.6 8Bit David font ..... 422
62.6.7 8Bit Drugulin font ..... 422
62.6.8 8Bit Ellinia font ..... 423
62.6.9 8Bit FrankRuehl font ..... 423
62.6.10 8Bit KtavYad font 42 ..... 423
62.6.11 8Bit Miri-amMono font . . . 424
62.6.12 8Bit Nachlieli font ..... 424
62.6.13 Hebrew fontswitching com-mands . . . . . . . 424
63 Hebrew in LATEX 2.09 com- patibility mode ..... 427
63.1 The Docstrip modules ..... 428
63.2 Obsolete style files ..... 428
64 The Bahasa language ..... 429
65 Not renaming hyphen.tex ..... 431
66 Support for formats based on PlainTEX ..... 432

## 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.
\selectlanguage
otherlanguage
\foreignlanguage
otherlanguage*
hyphenrules
\languagename
\iflanguage
shorthands
\defineshorthand
\aliasshorthand

When a user wants to switch from one language to another he can do so using the macro \selectlanguage. 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.

The environment otherlanguage does basically the same as \selectlanguage, 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 \begin\{otherlanguage\}. }

The command \foreignlanguage 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, not the names and dates.

In the environment otherlanguage* 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.

The environment hyphenrules can be used to select only the hyphenation rules to be used. This can for instance be used to select 'nohyphenation', provided that in language. dat the 'language' nohyphenation is defined by loading serohyph.tex.

The control sequence \languagename contains the name of the current language.

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 \iflanguage. 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 true or false respectively.

The command \useshorthands initiates the definition of user-defined shorthand sequences. It has one argument, the character that starts these personal shorthands.

The command \defineshorthand takes two arguments: the first is a oneor two-character shorthand sequence, and the second is the code the shorthand should expand to.

The command \aliasshorthand 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 \aliasshorthand\{"\}\{/\}. Please note that the substitute shorthand character must have been declared in the preamble of your document, using a command such as \useshorthands $\{/\}$ in this example.

The command anguageshorthandscanbeusedtoswitchtheshorthandsonthelanguagelevel.Ittakesoneargument,thenameofalanguage.Notethatforthistoworkthelanguageshouldhavebeenspecifiedasanoptionwhenloadingthebabelpackage.\shorthandonItissometimesnecessarytoswitchashorthandcharacterofftemporarily,becauseitmustbeusedinanentirelydifferentway.Forthispurpose,theusercommands\shorthandoffand\shorthandonareprovided.Theyeachtakealistofcharactersastheirarguments.Thecommand\shorthandoffsetsthe\catcodeforeachofthecharactersinitsargumenttoother(12);thecommand\shorthandonsetsthe\catcodetoactive(13).Bothcommandsonlyworkon'known'shorthandcharacters.Ifacharacterisnotknowntobeashorthandcharacteritscategorycodewillbeleftunchanged.\languageattributeThisisauser-levelcommand,tobeusedinthepreambleofadocument(after\usepackage[...]\{babel\}),thatdeclareswhichattributesaretobeusedforagivenlanguage.Ittakestwoarguments:thefirstisthenameofthelanguage;thesecond,a(listof)attribute(s)toused.Thecommandcheckswhetherthelanguageisknowninthisdocumentandwhethertheattribute(s)areknownforthislanguage.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

### 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 |
| 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 |


| Language | Option(s) |
| :--- | :--- |
| 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 |
| 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 typestting 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, $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ 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 $\{\backslash$ ref $\{\mathrm{foo}\}\}$ inside the argument of $\backslash$ chapter.

## 2 Changes for $\mathrm{EAT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$

With the advent of $\mathrm{LA}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ the interface to babel in the preamble of the document has changed. With $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2.09$ one used to call up the babel system with a line such as:
\documentstyle[dutch, english]\{article\}
which would tell $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$ that the document would be written in two languages, Dutch and English, and that English would be the first language in use.

The (\operatorname{La}_{\mathrm{E}}\mathrm{X}2\varepsilon\)wayofprovidingthesameinformationis:\documentclass\{article\}\usepackage[dutch,english]\{babel\}or,makingdutchandenglishglobaloptionsinordertoletotherpackagesdetectandusethem:undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
\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 $\pi о \lambda v \tau o \nu \kappa o ́$ ("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 3.6 k 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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$.


## 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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ 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 $\backslash$ language 0 .

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 $\mathrm{T}_{\mathrm{E}} \mathrm{u}$ users, so the files have to be coded so that they can be read by both $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ and plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$. 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 $\backslash\langle l a n g\rangle$ hyphenmins,
\captions $\langle l a n g\rangle, \backslash$ date $\langle l a n g\rangle, \backslash e x t r a s\langle l a n g\rangle$ and $\backslash$ noextras $\langle l a n g\rangle ;$ where $\langle l a n g\rangle$ is either the name of the language definition file or the name of the ${ }^{\mathrm{EAT}} \mathrm{E}_{\mathrm{E}} \mathrm{X}$ option that is to be used．These macros and their functions are dis－ cussed below．
－When a language definition file is loaded，it can define \l＠$\langle l a n g\rangle$ to be a dialect of \language0 when \1＠〈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．
\providehyphenmins


## \langhyphenmins

\captionslang
\datelang
\extraslang
\noextraslang
\bbl＠declare＠ttribute
main＠language
$\backslash$ ProvidesLanguage
\LdfInit

Idf＠quit
\ldf＠finish
\loadlocalcfg

The macro \providehyphenmins should be used in the language definition files to set the \lefthyphenmin and \righthyphenmin．This macro will check whether these parameters were provided by the hyphenation file before it takes any action．

The macro $\backslash\langle\operatorname{lang}\rangle$ hyphenmins is used to store the values of the \lefthyphenmin and \righthyphenmin．

The macro \captions $\langle l a n g\rangle$ defines the macros that hold the texts to replace the original hard－wired texts．

The macro \date〈lang〉defines \today and
The macro \extras $\langle l a n g\rangle$ contains all the extra definitions needed for a specific language．

Because we want to let the user switch between languages，but we do not know what state $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ might be in after the execution of $\backslash$ extras $\langle l a n g\rangle$ ，a macro that brings $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ into a predefined state is needed．It will be no surprise that the name of this macro is \noextras $\langle l a n g\rangle$ ．

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．

To postpone the activation of the definitions needed for a language until the beginning of a document，all language definition files should use \main＠language instead of \selectlanguage．This will just store the name of the language，and the proper language will be activated at the start of the document．

The macro \ProvidesLanguage should be used to identify the language definition files．Its syntax is similar to the syntax of the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ command \ProvidesPackage．

The macro \LdfInit 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＠－sign，preventing the ．ldf file from being processed twice，etc．

The macro \ldf＠quit does work needed if a ．ldf file was processed earlier． This includes resetting the category code of the＠－sign，preparing the language to be activated at \begin\｛document\} time, and ending the input stream. $｛document\} time, and ending the input stream. }$

The macro \ldf＠finish does work needed at the end of each ．ldf file．This includes resetting the category code of the＠－sign，loading a local configuration file，and preparing the language to be activated at \begin\｛document\} time. $｛document\} time. }$

After processing a language definition file， $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ can be instructed to load
\substitutefontfamily
@activate
\bbl@deactivate
\declare@shorthand
\bbl@add@special
\bbl@remove@special
babel@save
To save the current meaning of any control sequence, the macro \babel@save is provided. It takes one argument, $\langle\operatorname{csname}\rangle$, the control sequence for which the meaning has to be saved.
\babel@savevariable
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 $\langle$ variable $\rangle$.

[^1]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 $\backslash$ addto $\{\langle$ control sequence $\rangle\}\left\{\left\langle T_{E} X\right.\right.$ code $\left.\rangle\right\}$ 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
In a couple of European languages compound words are used. This means that when $T_{E} X$ 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.
\set@low@box 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 $\backslash$ 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\}.

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 $\mathrm{LAT}_{\mathrm{E}}$, some of the features implemented in the language definition files might be needed by plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ users. Care has been taken that all files in the system can be processed by plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$.

[^2]
## 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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$, therefor we make sure that the format file used is the right one.
$\backslash$ ProvidesLanguage The identification code for each file is something that was introduced in $\mathrm{LA}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$. When the command \ProvidesFile does not exist, a dummy definition is provided temporarily. For use in the language definition file the command $\backslash$ ProvidesLanguage is defined by babel.

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

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

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

[^3]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>
0.16 \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．

```
0.22\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.ldf\endcsname{#2}%
10.31 \endgroup}
10.32 </kernel\
10.33 \fi
10.34 </!package\rangle
```

Identify each file that is produced from this source file．

```
10.35 <+package\rangle\ProvidesPackage{babel}
10.36 <+core〉\ProvidesFile{babel.def}
10.37 <+kernel & patterns\rangle\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 [2005/05/21 v3.8g %
10.42 <+package\rangle The Babel package]
10.43 <+core\rangle Babel common definitions]
10.44 <+kernel\rangle Babel language switching mechanism]
10.45 <+driver\]
```


## 11 The Package File

In order to make use of the features of {AT}_{\mathrm{E}}\mathrm{X}2_{\varepsilon}\)，thebabelsystemcontainsapackagefile，babel．sty．Thisfileisloadedbythe\usepackagecommandanddefinesallthelanguageoptionsknowninthebabelsystem．Italsotakescareofanumberofcompatibilityissueswithotherpackages．undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

## 11．1 Language options

```
11.1〈*package\rangle
```


## 11.2 \ifx\LdfInit\@undefined\input babel.def $\backslash r e l a x \backslash f i$

For all the languages supported we need to declare an option.
$11.3 \backslash$ DeclareOption\{acadian\}\{\input\{frenchb.ldf\}\}
11.4 \DeclareOption\{afrikaans\}\{\input\{dutch.ldf\}\}
11.5 \DeclareOption\{american\}\{\input\{english.ldf\}\}
$11.6 \backslash$ DeclareOption\{australian\}\{\input\{english.ldf\}\}
Austrian is really a dialect of German.
11.7 \DeclareOption\{austrian\}\{\input\{germanb.ldf\}\}
$11.8 \backslash$ DeclareOption\{bahasa\}\{\input\{bahasa.ldf\}\}
11.9 \DeclareOption\{basque\}\{\input\{basque.ldf\}\}
$11.10 \backslash$ DeclareOption\{brazil\}\{\input\{portuges.ldf\}\}
11.11 \DeclareOption\{brazilian\}\{\input\{portuges.ldf\}\}
11.12 \DeclareOption\{breton\}\{\input\{breton.ldf\}\}
$11.13 \backslash$ DeclareOption\{british\}\{\input\{english.ldf\}\}
$11.14 \backslash$ DeclareOption\{bulgarian\}\{\input\{bulgarian.ldf\}\}
$11.15 \backslash$ DeclareOption\{canadian\}\{\input\{english.ldf\}\}
$11.16 \backslash$ DeclareOption\{canadien\}\{\input\{frenchb.ldf\}\}
11.17 \DeclareOption\{catalan\}\{\input\{catalan.ldf\}\}
11.18 \DeclareOption\{croatian\}\{\input\{croatian.ldf\}\}
11.19 \DeclareOption\{czech\}\{\input\{czech.ldf\}\}
$11.20 \backslash$ DeclareOption\{danish\}\{\input\{danish.ldf\}\}
11.21 \DeclareOption\{dutch\}\{\input\{dutch.ldf\}\}
11.22 \DeclareOption\{english\}\{\input\{english.ldf\}\}
$11.23 \backslash$ DeclareOption\{esperanto\}\{\input\{esperanto.ldf\}\}
11.24 \DeclareOption\{estonian\}\{\input\{estonian.ldf\}\}
$11.25 \backslash$ DeclareOption\{finnish\}\{\input\{finnish.ldf\}\}
The babel support or French used to be stored in francais.ldf; therefor the
${ }^{\mathrm{LA}} \mathrm{T}_{\mathrm{E}} \mathrm{X} 2.09$ option used to be francais. The hyphenation patterns may be loaded as either 'french' or as 'francais'.
$11.26 \backslash$ DeclareOption\{francais\}\{\input\{frenchb.ldf\}\}
11.27 \DeclareOption\{frenchb\}\{\input\{frenchb.ldf\}\}

With $\mathrm{HA}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ we can now also use the option french and still call the file frenchb.ldf.
11.28 \DeclareOption\{french\}\{\input\{frenchb.ldf\}\}\%
11.29 \DeclareOption\{galician\}\{\input\{galician.ldf\}\}
$11.30 \backslash$ DeclareOption\{german\}\{\input\{germanb.ldf\}\}
11.31 \DeclareOption\{germanb\}\{\input\{germanb.ldf\}\}
11.32 \DeclareOption\{greek\}\{\input\{greek.ldf\}\}
11.33 \DeclareOption\{polutonikogreek\}\{\%
11.34 \input\{greek.ldf\}\%
11.35 \languageattribute\{greek\}\{polutoniko\}\}
11.36 \DeclareOption\{hebrew\} \{\%
11.37 \input\{rlbabel.def\}\%
11.38 \input\{hebrew.ldf\}\}
hungarian is just a synonym for magyar
11.39 \DeclareOption\{hungarian\}\{\input\{magyar.ldf\}\}
$11.40 \backslash$ DeclareOption\{icelandic\}\{\input\{icelandic.ldf\}\}
11.41 \DeclareOption\{interlingua\}\{\input\{interlingua.ldf\}\}
11.42 \DeclareOption\{irish\}\{\input\{irish.ldf\}\}
11.43 \DeclareOption\{italian\}\{\input\{italian.ldf\}\}
11.44 \DeclareOption\{latin\}\{\input\{latin.ldf\}\}
11.45 \DeclareOption\{lowersorbian\}\{\input\{lsorbian.ldf\}\}
$11.46 \%{ }^{\sim}$ A $\backslash$ DeclareOption\{kannada\}\{\input\{kannada.ldf\}\}
11.47 \DeclareOption\{magyar\}\{\input\{magyar.ldf\}\}
$11.48 \% \sim \sim$ A $\backslash$ DeclareOption\{nagari\}\{\input\{nagari.ldf\}\}
'New' German orthography, including Austrian variant:
$11.49 \backslash$ DeclareOption\{naustrian\}\{\input\{ngermanb.ldf\}\}
$11.50 \backslash$ DeclareOption\{newzealand\}\{\input\{english.ldf\}\}
11.51 \DeclareOption\{ngerman\}\{\input\{ngermanb.ldf\}\}
11.52 \DeclareOption\{norsk\}\{\input\{norsk.ldf\}\}
11.53 \DeclareOption\{samin\}\{\input\{samin.ldf\}\}

For Norwegian two spelling variants are provided.
11.54 \DeclareOption\{nynorsk\}\{\input\{norsk.ldf\}\}
11.55 \DeclareOption\{polish\}\{\input\{polish.ldf\}\}
$11.56 \backslash$ DeclareOption\{portuges\}\{\input\{portuges.ldf\}\}
$11.57 \backslash$ DeclareOption\{portuguese\}\{\input\{portuges.ldf\}\}
11.58 \DeclareOption\{romanian\}\{\input\{romanian.ldf\}\}
11.59 \DeclareOption\{russian\}\{\input\{russianb.ldf\}\}
$11.60 \% \sim \sim A \backslash D e c l a r e O p t i o n\{s a n s k r i t\}\{\backslash i n p u t\{s a n s k r i t . l d f\}\}$
11.61 \DeclareOption\{scottish\}\{\input\{scottish.ldf\}\}
11.62 \DeclareOption\{serbian\}\{\input\{serbian.ldf\}\}
$11.63 \backslash$ DeclareOption\{slovak\}\{\input\{slovak.ldf\}\}
11.64 \DeclareOption\{slovene\}\{\input\{slovene.ldf\}\}
11.65 \DeclareOption\{spanish\}\{\input\{spanish.ldf\}\}
$11.66 \backslash$ DeclareOption\{swedish\}\{\input\{swedish.ldf\}\}
$11.67 \%$ ~~A
11.68 \DeclareOption\{turkish\}\{\input\{turkish.ldf\}\}
11.69 \DeclareOption\{ukrainian\}\{\input\{ukraineb.ldf\}\}
$11.70 \backslash$ DeclareOption\{uppersorbian\}\{\input\{usorbian.ldf\}\}
11.71 \DeclareOption\{welsh\}\{\input\{welsh.ldf\}\}
11.72 \DeclareOption\{UKenglish\}\{\input\{english.ldf\}\}
11.73 \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.
11.74 \DeclareOption*\{\%
11.75 \InputIfFileExists\{\CurrentOption.ldf\}\{\}\{\%
$11.76 \backslash$ PackageError\{babel\}\{\%
11.77 Language definition file \CurrentOption.ldf not found\}\{\%
11.78 Maybe you misspelled the language option?\}\}\%
$11.79\}$
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.

```
11.80 \InputIfFileExists{bblopts.cfg}{%
11.81 \typeout{*************************************^^J%
11.82 * Local config file bblopts.cfg used^^J%
11.83 *}%
11.84 }{}
```

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.
11.85 \DeclareOption\{activeacute\}\{\}
11.86 \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.
11.87 \DeclareOption\{KeepShorthandsActive\} \{\}

The options have to be processed in the order in which the user specified them: 11.88 \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.

```
11.89 \ifx\bbl@main@language\@undefined
11.90\PackageError{babel}{%
11.91 You haven't specified a language option}{%
11.92 You need to specify a language, either as a global
11.93 option\MessageBreak
11.94 or as an optional argument to the \string\usepackage\space
11.95 command; \MessageBreak
11.96 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.

```
11.97 \input{babel.def}
11.98\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.

```
11.99 \def\substitutefontfamily#1#2#3{%
11.100 \lowercase{\immediate\openout15=#1#2.fd\relax}%
11.101 \immediate\write15{%
11.102 \string\ProvidesFile{#1#2.fd}%
11.103 [\the\year/\two@digits{\the\month}/\two@digits{\the\day}
11.104 \space generated font description file]~ J
11.105 \string\DeclareFontFamily{#1}{#2}{}`~J
11.106 \string\DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}~- J
11.107 \string\DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}~~J
```

```
11.108 \string\DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{}~~ J
11.109 \string\DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}~~ J
11.110 \string\DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/bx/n}{}^^J
11.111 \string\DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/bx/it}{}^~J
11.112 \string\DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/bx/sl}{}^^J
11.113 \string\DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/bx/sc}{}^~ J
11.114 }%
11.115 \closeout15
11.116 }
```

This command should only be used in the preamble of a document.
11.117 \@onlypreamble\substitutefontfamily
11.118 〈/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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $T_{E} X$ users might want to use some of the features of the babel system too, care has to be taken that plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ and $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$, some of it is for the $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ 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 \/kernel core\rangle
```

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

```
12.7 <*core\rangle
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\rangle
12.13\AtEndOfPackage{\edef\latinencoding{\cf@encoding}}
```

But this might be overruled with a later loading of the package fontenc. Therefor we check at the execution of $\backslash$ 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{%
    \gdef\latinencoding{0T1}%
    \ifx\cf@encoding\bbl@t@one
        \xdef\latinencoding{\bbl@t@one}%
    \else
        \@ifl@aded{def}{t1enc}{\xdef\latinencoding{\bbl@t@one}}{}%
    \fi
    }
```

\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 $T_{E} X$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ environment has to record for which languages he has hyphenation patterns and in which files these are stored ${ }^{4}$. 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 IATEX 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\rangle
12.32〈*!patterns\rangle
12.33 \expandafter\ifx\csname iflanguage\endcsname\relax \else
12.34 \expandafter\endinput
12.35\fi
12.36 </!patterns\rangle
```

\language Plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 x language \(\backslash\) @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,

[^4]```
12.40\ifx\newlanguage\@undefined
12.41 \csname newcount\endcsname\last@language
    plain TEX version 3.0 uses \count }19\mathrm{ for this purpose.
```

```
12.42 \else
```

12.42 \else
12.43 \countdef\last@language=19
12.43 \countdef\last@language=19
12.44 \fi

```
12.44 \fi
```

\addlanguage To add languages to $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 's memory plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ version 3.0 supplies $\backslash$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ version 3.0.

```
ifx\newlanguage\@undefined
\def\addlanguage#1{%
\global\advance\last@language \@ne
\ifnum\last@language<\@cclvi
\else
                    \errmessage{No room for a new \string\language!}%
        \i
    \global\chardef#1\last@language
    \wlog{\string#1 = \string\language\the\last@language}}
```

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

```
12.54\else
12.55\\def\addlanguage{\alloc@9\language\chardef\@cclvi}
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.

```
2.57 \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
    \expandafter\@secondoftwo
    \fi}%
\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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ '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 $\backslash$ selectlanguage $\sqcup_{\sqcup}$. 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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X} 2.09$ writes to files expanded whereas $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ 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 $B u t$ when the language change happens inside a group the end of the group doesn't write anything to the auxiliary files. Therefor we need $T_{E}$ '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 $\backslash$ 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 $\backslash$ 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 $\backslash$ bbl@pop@language function can be simple:

```
12.81 \def\bbl@push@language{%
12.82 \xdef\bbl@language@stack{\languagename+\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 \languagename. For this we first define a helper function.
\bbl@pop@lang This macro stores its first element (which is delimited by the ' + '-sign) in \languagename 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\languagename{#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 $\backslash$ bbl@pop@lang is executed $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ first expands the stack, stored in \bbl@language@stack. The result of that is that the argument string of $\backslash \mathrm{bbl@pop@l}$ ang 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\bbl@set@language\expandafter{\languagename}%
12.91 }
```

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

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

\bbl@set@language The macro \bbl@set@language takes care of switching the language environment and of writing entries on the auxiliary files

```
12.96 \def \(\backslash \mathrm{bbl@set@language} \mathrm{\# 1} \mathrm{\{ } \mathrm{\%}\)
12.97 \edef \(\backslash\) languagename \(\{\%\)
2.98 \ifnum\escapechar=\expandafter'\string\#1\@empty
12.99 \else \string\#1\@empty\fi\}\%
12.100 \select@language \(\{\backslash\) languagename \(\} \%\)
```

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{\languagename}}%
12.103 \addtocontents{toc}{\xstring\select@language{\languagename}}%
12.104 \addtocontents{lof}{\xstring\select@language{\languagename}}%
12.105 \addtocontents{lot}{\xstring\select@language{\languagename}}%
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 $\backslash i f x \backslash c s n a m e ~ l @ \# 1 \backslash e n d c s n a m e \backslash r e l a x$
\@nolanerr\{\#1\}\%
12.110 \else
12.111 \expandafter\ifx\csname date\#1 \endcsname\relax
\@noopterr\{\#1\}\%
\else
\language=\csname l@\#1\endcsname\relax
\originalTeX

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

```
\edef\tmp{\string english}
\ifx\languagename\tmp
    ...
\else
    ...
\i
```

The construction with \string is necessary because \languagename 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 $\langle$ lang $\rangle$ 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 $\backslash\langle l a n g\rangle$ hyphenmins is defined. If it is not, we set default values (2 and 3), otherwise the values in \〈lang〉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 $\backslash$ languagename; it then calls \selectlanguage ${ }_{\sqcup}$ 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\languagename\endcsname
12.147 }
```

$\backslash f o r e i g n l a n g u a g e ~ T h e ~ \ f o r e i g n l a n g u a g e ~ c o m m a n d ~ i s ~ a n o t h e r ~ s u b s t i t u t e ~ f o r ~ t h e ~ \ s e l e c t l a n g u a g e ~$ 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 $\backslash$ 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 $\backslash$ languagename\{\#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 \language=\csname l@#1\endcsname\relax
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 }
```

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

```
12.173\def\hyphenrules#1{%
12.174 \expandafter\ifx\csname l@#1\endcsname\@undefined
12.175 \@nolanerr{#1}%
12.176 \else
12.177 \language=\csname 1@#1\endcsname\relax
12.178 \languageshorthands{none}%
12.179 \fi
12.180 }
12.181 \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 $\backslash r i g h t h y p h e n m i n$. If the macro $\backslash\langle l a n g\rangle$ hyphenmins is already defined this command has no effect

```
12.182 \def \providehyphenmins#1#2{%
12.183 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
12.184 \@namedef{#1hyphenmins}{#2}%
12.185 \fi}
```

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

```
12.186\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.

```
12.187 \def \LdfInit{%
12.188 \chardef\atcatcode=\catcode`\@
12.189 \catcode`\@=11\relax
12.190 \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

```
12.191 \catcode`\@=\atcatcode \let\atcatcode\relax
12.192 \LdfInit}
12.193 </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

```
12.194 <*core〉
12.195\def\LdfInit#1#2{%
12.196 \chardef\atcatcode=\catcode'\@
12.197 \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. Therefor we store its current catcode and restore it later on.

```
12.198 \chardef\eqcatcode=\catcode'\=
12.199 \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.

```
12.200 \let\bbl@tempa\relax
12.201 \expandafter\if\expandafter\@backslashchar
    \expandafter\@car\string#2\@nil
    \ifx#2\@undefined
    \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.

```
12.205
12.206 \fi
12.207 \else
```

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

```
12.208 \expandafter\ifx\csname#2\endcsname\relax
12.209 \else
12.210 \def\bbl@tempa{\ldf@quit{#1}}
12.211 \fi
12.212 \fi
12.213 \bbl@tempa
```

Finally we check \originalTeX.

```
12.214 \ifx\originalTeX\@undefined
12.215 \let\originalTeX\@empty
12.216 \else
12.217 \originalTeX
12.218 \fi}
```

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

```
12.219 \def\ldf@quit#1{%
12.220 \expandafter\main@language\expandafter{#1}%
12.221 \catcode`\@=\atcatcode \let\atcatcode\relax
12.222 \catcode`\==\eqcatcode \let\eqcatcode\relax
12.223 \endinput
12.224 }
```

\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.

```
12.225 \def\ldf@finish#1{%
12.226 \loadlocalcfg{#1}%
12.227 \expandafter\main@language\expandafter{#1}%
12.228 \catcode'\@=\atcatcode \let\atcatcode\relax
12.229 \catcode`\==\eqcatcode \let\eqcatcode\relax
12.230 }
```

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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$

```
12.231 \@onlypreamble\LdfInit
12.232 \@onlypreamble\ldf@quit
12.233 \@onlypreamble\ldf@finish
```

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

```
12.234 \def\main@language#1{%
12.235 \def\bbl@main@language{#1}%
12.236 \let\languagename\bbl@main@language
12.237 \language=\csname l@\languagename\endcsname\relax
12.238 }
```

The default is to use English as the main language.

```
12.239 \ifx\l@english\@undefined
12.240 \let\l@english\z@
12.241 \fi
12.242 \main@language{english}
```

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

```
12.243\AtBeginDocument{%
12.244 \expandafter\selectlanguage\expandafter{\bbl@main@language}}
12.245 </core)
```

\originalTeX The macro\originalTeX should be known to $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ at this moment. As it has to be expandable we \let it to \@empty instead of \relax.

```
12.246 <*kernel>
12.247\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.
$12.248 \backslash i f x \backslash$ babel@beginsave $\backslash$ @undefined $\backslash$ let $\backslash$ babel@beginsave $\backslash r e l a x \backslash f i$
\@nolanerr The babel package will signal an error when a documents tries to select a language \@nopatterns 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 $\backslash$ PackageError it must be $\mathrm{LATEX}_{2} 2_{\varepsilon}$, so we can safely use its error handling interface. Otherwise we'll have to 'keep it simple'.

```
12.249\ifx\PackageError\@undefined
12.250 \def\@nolanerr#1{%
12.251 \errhelp{Your command will be ignored, type <return> to proceed}%
12.252 \errmessage{You haven't defined the language #1\space yet}}
12.253\\def\@nopatterns#1{%
12.254 \message{No hyphenation patterns were loaded for}%
12.255 \message{the language '#1'}%
12.256 \message{I will use the patterns loaded for \string\language=0
12.257 instead}}
12.258\\def\@noopterr#1{%
12.259 \errmessage{The option #1 was not specified in \string\usepackage}
12.260 \errhelp{You may continue, but expect unexpected results}}
12.261 \def\@activated#1{%
12.262 \wlog{Package babel Info: Making #1 an active character}}
12.263\else
12.264 \newcommand*{\@nolanerr}[1]{%
12.265 \PackageError{babel}%
12.266 {You haven't defined the language #1\space yet}%
12.267 {Your command will be ignored, type <return> to proceed}}
12.268 \newcommand*{\@nopatterns}[1]{%
12.269 \PackageWarningNoLine{babel}%
12.270 {No hyphenation patterns were loaded for\MessageBreak
12.271 the language '#1'\MessageBreak
```

```
12.272 I will use the patterns loaded for \string\language=0
12.273 instead}}
12.274 \newcommand*{\@noopterr}[1]{%
12.275 \PackageError{babel}%
12.276 {You haven't loaded the option #1\space yet}%
12.277 {You may proceed, but expect unexpected results}}
12.278 \newcommand*{\@activated}[1]{%
12.279 \PackageInfo{babel}{%
12.280 Making #1 an active character}}
12.281 \fi
```

The following code is meant to be read by iniTEX because it should instruct $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ to read hyphenation patterns. To this end the docstrip option patterns can be used to include this code in the file hyphen.cfg.

```
12.282〈*patterns\rangle
```

$\backslash$ process@line Each line in the file language. dat is processed by $\backslash$ 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.

```
12.283\def\process@line#1#2 #3/{%
12.284 \ifx=#1
12.285 \process@synonym#2 /
12.286 \else
12.287 \process@language#1#2 #3/%
12.288 \fi
12.289 }
```

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

```
12.290 \toks@{}
12.291 \def\process@synonym#1 /{%
12.292 \ifnum\last@language=\m@ne
```

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

```
12.293 \expandafter\chardef\csname l@#1\endcsnameO\relax
12.294 \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.

```
\(12.295 \backslash\) toks@\expandafter\{\the \(\backslash\) toks@
12.296 \expandafter\let\csname \#1hyphenmins \(\backslash e x p a n d a f t e r \backslash e n d c s n a m e\)
12.297 \csname\languagename hyphenmins \endcsname\}\%
12.298 \else
```

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

```
12.299 \expandafter\chardef\csname l@#1\endcsname\last@language
12.300 \wlog{\string\l@#1=\string\language\the\last@language}
```

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

```
12.301 \expandafter\let\csname #1hyphenmins\expandafter\endcsname
12.302 \csname\languagename hyphenmins\endcsname
12.303 \fi
12.304 }
```

\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'

```
12.305 \def\process@language#1 #2 #3/{%
12.306 \expandafter\addlanguage\csname l@#1\endcsname
12.307 \expandafter\language\csname l@#1\endcsname
12.308\def\languagename{#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.

```
12.309 \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 ' $: T 1$ ' 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.

```
12.310 \begingroup
12.311 \bbl@get@enc#1:\@@@
12.312 \ifx\bbl@hyph@enc\@empty
12.313 \else
12.314 \fontencoding{\bbl@hyph@enc}\selectfont
12.315 \fi
```

Some pattern files contain assignments to \lefthyphenmin and $\backslash r i g h t h y p h e n m i n$. $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ does not keep track of these assignments. Therefor we try to detect such assignments and store them in the $\backslash\langle$ lang $\rangle$ hyphenmins macro. When no assignments were made we provide a default setting.

## $12.316 \quad \backslash l e f t h y p h e n m i n \backslash m @ n e$

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

### 12.317 \input \#2\relax

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

```
12.318 \ifnum\lefthyphenmin=\m@ne
12.319 \else
```

```
12.320 \expandafter\xdef\csname #1hyphenmins\endcsname{%
12.321 \the\lefthyphenmin\the\righthyphenmin}%
12.322 \fi
12.323 \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 .

```
12.324 \ifnum\the\language=\z@
12.325 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
    \set@hyphenmins\tw@\thr@@\relax
    \else
            \expandafter\expandafter\expandafter\set@hyphenmins
                \csname #1hyphenmins\endcsname
    \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.

```
12.331 \the\toks@
12.332 \fi
```

Empty the token register after use.

```
12.333 \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.

```
12.334 \def\bbl@tempa{#3}%
12.335 \ifx\bbl@tempa\@empty
12.336 \else
12.337 \ifx\bbl@tempa\space
12.338 \else
12.339 \input #3\relax
12.340 \fi
12.341 \fi
12.342 }
```

\bbl@get@enc The macro \bbl@get@enc extracts the font encoding from the language name and $\backslash$ bbl@hyph@enc stores it in \bbl@hyph@enc. It uses delimited arguments to achieve this
12.343 \def $\backslash \mathrm{bbl@get@enc} \mathrm{\# 1:} \mathrm{\# 2} \mathrm{\backslash @@@} \mathrm{\{ } \mathrm{\%}$

First store both arguments in temporary macros,
12.344 \def $\backslash$ bbl@tempa\{\#1\}\%
12.345 \def $\backslash$ bbl@tempb $\{\# 2\} \%$
then, if the second argument was empty, no font encoding was specified and we're done.

```
12.346 \ifx\bbl@tempb\@empty
12.347 \let\bbl@hyph@enc\@empty
12.348 \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.

```
12.349 \bbl@get@enc#2\@@@
12.350 \edef\bbl@hyph@enc{\bbl@tempa}%
12.351 \fi}
```

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

```
12.352 \openin1 = language.dat
```

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

```
12.353 \ifeof1
12.354 \message{I couldn't find the file language.dat,\space
12.355 I will try the file hyphen.tex}
12.356 \input hyphen.tex\relax
12.357 \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 .

```
12.358 \last@language\m@ne
```

We now read lines from the file until the end is found
12.359 \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.

```
12.360 \endlinechar\m@ne
12.361 \read1 to \bbl@line
12.362 \endlinechar'\~`M
```

Empty lines are skipped.
\ifx\bbl@line\@empty
12.364 lelse

Now we add a space and a / character to the end of $\backslash \mathrm{bbl@l}$ ine. This is needed to be able to recognize the third, optional, argument of \process@language later on.

```
12.365 \edef\bbl@line{\bbl@line\space/}%
12.366 \expandafter\process@line\bbl@line
12.367 \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 $T_{E} X$ has to read \if/\fi pairs. So we have to insert a 'dummy' \iftrue.

```
12.368 \iftrue \csname fi\endcsname
12.369 \csname if\ifeof1 false\else true\fi\endcsname
12.370 \repeat
    Reactivate the default patterns,
12.371 \language=0
12.372 \fi
```

    and close the configuration file.
    12.373 \closein1
Also remove some macros from memory
12.374 \let \process@language\@undefined
12.375 \let $\backslash p r o c e s s @ s y n o n y m \ @ u n d e f i n e d$
12.376 \let $\backslash p r o c e s s @ l i n e \backslash @ u n d e f i n e d ~$
12.377 \let $\backslash b b 1 @ t e m p a \backslash$ @undefined
12.378 \let $\backslash b b l @ t e m p b \backslash @ u n d e f i n e d$
12.379 \let\bbl@eq@\@undefined
12.380 \let $\backslash b b 1 @ 1 i n e \backslash @ u n d e f i n e d$
12.381 \let $\backslash \mathrm{bbl@get@enc} \mathrm{\ @undefined}$

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

```
12.382 \ifx\addto@hook\@undefined
12.383 \else
12.384 \expandafter\addto@hook\expandafter\everyjob\expandafter{%
12.385 \expandafter\typeout\expandafter{\the\toks8 loaded.}}
12.386\fi
```

Here the code for $\mathrm{ini}_{\mathrm{E}} \mathrm{X}$ ends.

```
12.387 </patterns>
```

12.388 〈/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 $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ 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.

```
12.389 <*core | shorthands\rangle
```

12.390 \def \bbl@add@special\#1\{\begingroup
$12.391 \backslash$ def $\backslash$ do $\{\backslash$ noexpand $\backslash$ do $\backslash$ noexpand $\} \%$
$12.392 \backslash$ def $\backslash$ @makeother\{ \noexpand $\backslash$ @makeother $\backslash$ 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.

```
12.393 \edef\x{\endgroup
```

```
12.394 \def\noexpand\dospecials{\dospecials\do#1}%
12.395 \expandafter\ifx\csname @sanitize\endcsname\relax \else
12.396 \def\noexpand\@sanitize{\@sanitize\@makeother#1}%
12.397 \fi}%
```

The macro $\backslash \mathrm{x}$ contains at this moment the following:
\endgroup $\backslash$ def $\backslash$ dospecials\{old contents $\backslash$ do $\langle$ char $\rangle$ \}.
If \@sanitize is defined, it contains an additional definition of this macro. The last thing we have to do, is the expansion of $\backslash \mathrm{x}$. Then \endgroup is executed, which restores the old meaning of $\backslash \mathrm{x}$, \do and \@makeother. After the group is closed, the new definition of \dospecials (and \@sanitize) is assigned.

```
12.398 \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 $\backslash \mathrm{x}$, which expands to empty if the characters match, otherwise it expands to its nonexpandable input. Because $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ inserts a $\backslash r e l a x$, if the corresponding \else or \fi is scanned before the comparison is evaluated, we provide a 'stop sign' which should expand to nothing.

```
12.399 \def\bbl@remove@special#1{\begingroup
12.400\\def\x##1##2{\ifnum`#1=`##2\noexpand\@empty
12.401 \else\noexpand##1\noexpand##2\fi}%
```

With the help of this macro we define \do and $\backslash$ make@other

```
12.402 \def \do{\x\do}%
12.403 \def\@makeother{\x\@makeother}%
```

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

```
12.404 \edef\x{\endgroup
12.405 \def\noexpand\dospecials{\dospecials}%
12.406 \expandafter\ifx\csname @sanitize\endcsname\relax \else
12.407 \def\noexpand\@sanitize{\@sanitize}%
12.408 \fi}%
12.409 \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 $\langle$ char $\rangle$ by default ( $\langle c h a r\rangle$ being the character to be made active). Later its definition can be changed to expand to \active@char $\langle$ char $\rangle$ by calling \bbl@activate\{ $\langle$ char $\rangle\}$.

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

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

 \bbl@afterfi look ahead, we take extra care to 'throw' it over the \else and \fi parts of an $\backslash i f-s t a t e m e n t{ }^{5}$. These macros will break if another \if... $\backslash$ fi statement appears in one of the arguments.

```
12.410\long\def\bbl@afterelse#1\else#2\fi{\fi##1}
12.411 \long\def\bbl@afterfi#1\fi{\fi#1}
```

$\backslash$ 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.

```
12.412 %\def\peek@token#1#2{%
12.413% \expandafter\let\expandafter\bbl@nexta\csname #1\string#2\endcsname
12.414% \expandafter\let\expandafter\bbl@nextb
12.415 % \csname system@active\string#2\endcsname
12.416 % \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.

```
12.417%\def\bbl@test@token{%
12.418% \let\bbl@next\bbl@nexta
12.419% \ifcat\noexpand\bbl@token a%
12.420% \else
12.421% \ifcat\noexpand\bbl@token=%
12.422% \else
12.423% \ifcat\noexpand\bbl@token\noexpand\bbl@next
12.424% \else
12.425% \let\bbl@next\bbl@nextb
12.426% \fi
12.427% \fi
12.428% \fi
12.429% \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.

[^5]```
12.430\def\initiate@active@char#1{%
12.431 \expandafter\ifx\csname active@char\string##1\endcsname\relax
12.432 \bbl@afterfi{\@initiate@active@char{#1}}%
12.433 \fi}
```

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

```
12.434 \begingroup
12.435 \catcode'\~\active
12.436 \def\x{\endgroup
12.437 \def\@initiate@active@char##1{%
```

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

```
12.438 \ifcat\noexpand##1\noexpand~\relax
12.439 \@ifundefined{normal@char\string##1}{%
12.440 \expandafter\let\csname normal@char\string##1\endcsname##1%
12.441 \expandafter\gdef
12.442 \expandafter##1%
12.443 \expandafter{%
12.444 \expandafter\active@prefix\expandafter##1%
12.445 \csname normal@char\string##1\endcsname}}{}%
12.446 \else
```

Otherwise we write a message in the transcript file,
$12.447 \backslash$ @activated\{\#\#1\}\%
and define \normal@char〈char〉 to expand to the character in its default state.
12.448 \@namedef\{normal@char\string\#\#1\}\{\#\#1\}\%

If we are making the right quote active we need to change $\backslash p r @ m @ s$ as well.

```
12.449 \ifx##1'%
12.450 \let\prim@s\bbl@prim@s
```

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

```
12.451 \@namedef{normal@char\string##1}{%
12.452 \textormath{##1}{^\bgroup\prim@s}}%
12.453 \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.

```
12.454 \ifx##1~%
12.455 \gdef\bbl@act@caret{%
12.456 \ifmmode
12.457 \csname normal@char\string^\endcsname
12.458 \else
12.459 \bbl@afterfi
12.460 {\if@safe@actives
12.461 \bbl@afterelse\csname normal@char\string##1\endcsname
12.462 \else
```

```
12.463
12.464
12.465
12.466
                                    \bbl@afterfi\csname user@active\string##1\endcsname
                                    \fi}%
    \i}
\i
```

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

```
12.467 \@ifpackagewith{babel}{KeepShorthandsActive}{}{%
12.468 \edef\bbl@tempa{\catcode`\noexpand##1\the\catcode`##1}%
12.469 \expandafter\AtEndOfPackage\expandafter{\bbl@tempa}}%
```

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

| 12.470 | \@tempenta=\lccode' ) $^{\text {/ }}$ |
| :---: | :---: |
| 12.471 | \lccode ${ }^{\text {~ }}$ = ${ }^{\text {\#\# }} 1 \%$ |
| 12.472 | \lowercase\{\% |

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

```
12.473 \catcode* \\active
12.474 \expandafter\bbl@add@special
12.475 \csname \string##1\endcsname
```

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

| 12.476 | $\backslash$ AtBeginDocument $\{\%$ |
| :--- | ---: |
| 12.477 | \catcode‘\#\#1 $\backslash$ 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 $\backslash$ bibitem for example.

```
12.478
12.479 \immediate\write\@mainaux{%
12.480 \string\catcode`##1\string\active}%
12.481 \fi}%
```

Define the character to expand to

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

(where \active@char〈char〉 is one control sequence!).

```
12.482 \expandafter\gdef
12.483 \expandafter ~
12.484 \expandafter{%
12.485 \expandafter\active@prefix\expandafter##1%
12.486 \csname normal@char\string##1\endcsname}}%
12.487 \lccode`\~\@tempcnta
12.488 \fi
```

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

```
12.489
12.490
\ifx##1~%
    \@namedef{active@char\string##1}{\bbl@act@caret}%
12.491
\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 $\langle c h a r\rangle$.

```
12.492 \@namedef{active@char\string##1}{%
12.493 \if@safe@actives
        \bbl@afterelse\csname normal@char\string##1\endcsname
    \else
        \bbl@afterfi\csname user@active\string##1\endcsname
    \fi}%
\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.

```
12.499 \@namedef{user@active\string##1}{%
12.500 \expandafter\ifx
12.501 \csname \user@group @sh@\string##1@\endcsname
12.502 \relax
12.503 \bbl@afterelse\bbl@sh@select\user@group##1%
12.504 {user@active@arg\string##1}{language@active\string##1}%
12.505 \else
    \bbl@afterfi\csname \user@group @sh@\string##1@\endcsname
    \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. Therefor \peek@token is called to decide that.

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

| 12.522 | \expandafter\ifx |
| :---: | :---: |
| 12.523 | \csname \language@group @sh@\string\#\#1@\endcsname |
| 12.524 | $\backslash \mathrm{relax}$ |
| 12.525 | \bbl@afterelse\bbl@sh@select\language@group\#\#1\% |
| 12.526 | \{language@active@arg\string\#\#1\}\{system@active\string\#\#1\}\% |
| 12.527 | $\backslash \mathrm{else}$ |
| 12.528 | \bbl@afterfi |
| 12.529 | \csname \language@group @sh@\string\#\#1@\endcsname |
| 12.530 | \fi\}\% |
| 12.531 | $\backslash$ long\@namedef\{language@active@arg\string\#\#1\}\#\#\#\#1\{\% |
| 12.532 | \expandafter\ifx |
| 12.533 | \csname \language@group @sh@\string\#\#1@\string\#\#\#\#1@\endcsname |
| 12.534 | $\backslash \mathrm{relax}$ |
| 12.535 | \bbl@afterelse |
| 12.536 | \csname system@active\string\#\#1\endcsname\#\#\#\#1\% |
| 12.537 | $\backslash \mathrm{else}$ |
| 12.538 | \bbl@afterfi |
| 12.539 | \csname \language@group @sh@\string\#\#1@\string\#\#\#\#1@\% |
| 12.540 | \endcsname |
| 12.541 | \fi\}\% |

And the same goes for the system level.

```
12.542 \@namedef{system@active\string##1}{%
12.543 \expandafter\ifx
12.544 \csname \system@group @sh@\string##1@\endcsname
12.545 \relax
12.546 \bbl@afterelse\bbl@sh@select\system@group##1%
12.547 {system@active@arg\string##1}{normal@char\string##1}%
12.548 \else
    \bbl@afterfi\csname \system@group @sh@\string##1@\endcsname
    \fi}%
```

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

```
12.551 \long\@namedef{system@active@arg\string##1}####1{%
12.552 \expandafter\ifx
12.553 \csname \system@group @sh@\string##1@\string####1@\endcsname
12.554 \relax
12.555 \bbl@afterelse\csname normal@char\string##1\endcsname####1%
12.556 \else
12.557 \bbl@afterfi
12.558 \csname \system@group @sh@\string##1@\string####1@\endcsname
12.559 \fi}%
```

When a shorthand combination such as ', ends up in a heading $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ would see $\backslash$ protect' $\backslash$ protect'. To prevent this from happening a shorthand needs to be defined at user level.

```
12.560
12.561
12.562
```

```
\@namedef{user@sh@\string##1@\string\protect@}{%
```

\@namedef{user@sh@\string\#\#1@\string\protect@}{%
\csname user@active\string\#\#1\endcsname}%
\csname user@active\string\#\#1\endcsname}%
}%

```
}%
```

$\backslash$ 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 $\backslash$ bbl@firstcs or $\backslash$ bbl@scndcs. Hence two more arguments need to follow it.

```
12.564 \def\bbl@sh@select#1#2{%
12.565 \expandafter\ifx\csname#1@sh@\string#2@sel\endcsname\relax
12.566 \bbl@afterelse\bbl@scndcs
12.567 \else
12.568 \bbl@afterfi\csname#1@sh@\string#2@sel\endcsname
12.569 \fi
12.570 }
```

\active@prefix The command \active@prefix which is used in the expansion of active characters has a function similar to \0T1-cmd in that it \protects the active character whenever \protect is not \@typeset@protect.

```
12.571 \def\active@prefix#1{%
12.572 \ifx\protect\@typeset@protect
12.573 \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).

```
12.574 \ifx\protect\@unexpandable@protect
12.575 \bbl@afterelse\bbl@afterfi\noexpand#1\@gobble
12.576 \else
12.577 \bbl@afterfi\bbl@afterfi\protect#1\@gobble
12.578 \fi
12.579 \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 $\langle c h a r\rangle$.
12.580 \newif $\backslash i f @ s a f e @ a c t i v e s$
12.581 \@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.
\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 $\langle c h a r\rangle$ instead of \normal@char $\langle c h a r\rangle$.

```
12.583\def\bbl@activate#1{%
12.584 \expandafter\def
12.585 \expandafter#1\expandafter{%
12.586 \expandafter\active@prefix
12.587 \expandafter#1\csname active@char\string#1\endcsname}%
12.588 }
```

\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 $\langle c h a r\rangle$.
12.589 \def $\backslash$ bbl@deactivate\#1\{\%
12.590 \expandafter $\backslash$ def
12.591 \expandafter\#1 \expandafter\{\%
12.592 \expandafter\active@prefix
12.593 \expandafter\#1\csname normal@char\string\#1\endcsname\}\%
$12.594\}$
\bbl@firstcs These macros have two arguments. They use one of their arguments to build a $\backslash \mathrm{bbl@scndcs}$ control sequence from.

```
12.595 \def\bbl@firstcs#1#2{\csname#1\endcsname}
12.596\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. $\sim$ or "a;
3. the code to be executed when the shorthand is encountered.
```
12.597\def\declare@shorthand#1#2{\@decl@short{#1}#2\@nil}
12.598 \def\@decl@short#1#2#3\@nil#4{%
12.599 \def\bbl@tempa{#3}%
12.600 \ifx\bbl@tempa\@empty
12.601 \expandafter\let\csname #1@sh@\string#2@sel\endcsname\bbl@scndcs
12.602 \@namedef{#1@sh@\string#2@}{#4}%
12.603 \else
12.604 \expandafter\let\csname #1@sh@\string#2@sel\endcsname\bbl@firstcs
12.605 \@namedef{#1@sh@\string#2@\string#3@}{#4}%
12.606 \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.

```
12.608 \ifmmode
12.609 \bbl@afterelse#2%
12.610 \else
12.611 \bbl@afterfi#1%
12.612 \fi}
```

\user@group The current concept of 'shorthands' supports three levels or groups of shorthands. $\backslash$ 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'.
12.613 \def \user@group\{user\}
12.614 \def $\backslash$ language@group\{english\}
12.615 \def $\backslash$ system@group\{system $\}$
\useshorthands This is the user level command to tell $\mathrm{HAT}_{\mathrm{E}} \mathrm{X}$ that user level shorthands will be used in the document. It takes one argument, the character that starts a shorthand.
12.616 \def \useshorthands\#1 $\{\%$

First note that this is user level.

### 12.617 \def \user@group\{user\}\%

Then initialize the character for use as a shorthand character.
12.618 \initiate@active@char\{\#1\}\%

Now that $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ has seen the character its category code is fixed, but for the actions of $\backslash$ bbl@activate to succeed we need it to be active. Hence the trick with the \lccode to circumvent this.
12.619 \@tempenta\lccode‘ ${ }^{\text {~ }}$
12.620 \lccode ${ }^{\text {‘~ }}=$ ‘\#1\%
12.621 \lowercase\{\catcode‘~ ${ }^{\text {(active\} \backslash \text { bbl@activate\{~\}\} } }$
12.622 \lccode‘\~\@tempenta\}
\defineshorthand Currently we only support one group of user level shorthands, called 'user'.
$12.623 \backslash \operatorname{def} \backslash$ defineshorthand\{\declare@shorthand\{user\}\}
\languageshorthands A user level command to change the language from which shorthands are used.
$12.624 \backslash$ def $\backslash$ languageshorthands\#1\{\def $\backslash$ language@group\{\#1\}\}
\aliasshorthand
12.625 \def $\backslash$ aliasshorthand\#1\#2\{\%

First the new shorthand needs to be initialized,
12.626 \expandafter\ifx\csname active@char\string\#2\endcsname\relax
12.627 \ifx\document \@notprerr
12.628 \@notshorthand\{\#2\}
12.629 \else
12.630 \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.

```
12.631 \@tempcnta\lccode'\~
12.632 \lccode` ~}=`#2
12.633 \lowercase{\let~#1}%
12.634 \lccode`\~\@tempcnta
12.635 \fi
12.636 \fi
12.637 }
```

\@notshorthand

```
12.638\def\@notshorthand#1{%
12.639 \PackageError{babel}{%
12.640 The character '\string #1t' should be made
12.641 a shorthand character;\MessageBreak
12.642 add the command \string\useshorthands\string{#1\string} to
12.643 the preamble.\MessageBreak
12.644 I will ignore your instruction}{}%
12.645 }
```

\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.

```
12.646 \newcommand*\shorthandon[1]{\bbl@switch@sh{on}#1\@nil}
12.647 \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.

```
12.648 \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.
12.649 \@ifundefined\{active@char\string\#2\}\{\%
12.650 \PackageError\{babel\}\{\%
12.651 The character '\string \#2' is not a shorthand character
12.652 in \languagename\}\{\%
12.653 Maybe you made a typing mistake? $\backslash$ MessageBreak
12.654 I will ignore your instruction\}\}\{\%
12.655 \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.

```
12.656 \ifx#3\@empty\else
12.657 \bbl@afterfi\bbl@switch@sh{#1}#3\@nil
12.658 \fi}
```

$\backslash b b l @ s w i t c h @ s h @ o f f$ 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).

```
12.659 \def\bbl@switch@sh@off#1{\catcode`#112\relax}
```

$\backslash$ 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 $\backslash \mathrm{x}$ gets executed.

```
12.660 \begingroup
12.661 \catcode`\~\active
12.662 \def\x{\endgroup
12.663 \def\bbl@switch@sh@on##1{%
12.664 \lccode` ~`"#1%
12.665 \lowercase{%
12.666 \catcode` ` \active
12.667 }%
12.668 }%
12.669 }
```

The next operation makes the above definition effective.

```
12.670 \x
```

12.671 \%

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

```
12.672 \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.)

```
12.673\declare@shorthand{system}{'}{%
12.674 \textormath{\csname normal@char\string'\endcsname}%
12.675 {\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.

```
12.676 \declare@shorthand{system}{`}{\csname normal@char\string`\endcsname}
```

$\backslash b b l @ p r i m @ s$ One of the internal macros that are involved in substituting \prime for each right $\backslash b b l @ p r @ m$ s quote in mathmode is $\backslash$ 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.

```
12.677 \def\bbl@prim@s{%
12.678 \prime\futurelet\@let@token\bbl@pr@m@s}
```

```
12.679 \begingroup
12.680 \catcode`\`\active\let'\relax
12.681 \def\x{\endgroup
12.682 \def\bbl@pr@m@s{%
12.683 \ifx'\@let@token
12.684 \expandafter\pr@@@s
12.685 \else
12.686 \ifx^\@let@token
12.687 \expandafter\expandafter\expandafter\pr@@@t
12.688 \else
12.689 \egroup
12.690 \fi
12.691 \fi}%
12.692 }
12.693\x
12.694 \/core | shorthands〉
```

Normally the $\sim$ is active and expands to \penalty $\backslash @ M \backslash_{\sqcup}$. 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.

```
12.695 <*Core)
12.696 \initiate@active@char{~}
12.697\declare@shorthand{system}{~ }{\leavevmode\nobreak\}
12.698 \bbl@activate{~
```

$\backslash 0 T 1$ dqpos The position of the double quote character is different for the OT1 and T1 encod$\backslash T 1 d q p o s$ ings. It will later be selected using the $\backslash$ f@encoding macro. Therefor we define two macros here to store the position of the character in these encodings.

```
12.699 \expandafter\def\csname OT1dqpos\endcsname{127}
12.700 \expandafter\def\csname T1dqpos\endcsname{4}
```

When the macro \f@encoding is undefined (as it is in plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ ) we define it here to expand to OT1
12.701 \ifx $\backslash$ f@encoding $\backslash$ @undefined
$12.702 \backslash$ def $\backslash$ f@encoding\{OT1\}
12.703 \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.
12.704 \newcommand \languageattribute [2] \{\%

First check whether the language is known.

```
12.705\expandafter\ifx\csname l@#1\endcsname\relax
```

```
12.706\@nolanerr{#1}%
12.707 \else
```

Than process each attribute in the list.

```
12.708 \@for\bbl@attr:=#2\do{%
```

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

```
\(12.709 \backslash i f x \backslash b b l @ k n o w n @ a t t r i b s \backslash @ u n d e f i n e d\)
12.710 \in@false
12.711 \else
```

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

```
12.712 \edef\bbl@tempa{\noexpand\in@{,#1-\bbl@attr,}%
12.713 {,\bbl@known@attribs,}}%
12.714 \bbl@tempa
12.715 \fi
```

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

```
12.716 \ifin@
12.717 \PackageWarning{Babel}{%
12.718 You have more than once selected the attribute
12.719 '\bbl@attr'\MessageBreak for language #1}%
12.720 \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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$-code.

```
12.721 \edef\bbl@tempa{%
12.722 \noexpand\bbl@add@list\noexpand\bbl@known@attribs{#1-\bbl@attr}}%
12.723 \bbl@tempa
12.724 \edef\bbl@tempa{#1-\bbl@attr}%
12.725 \expandafter\bbl@ifknown@ttrib\expandafter{\bbl@tempa}\bbl@attributes%
12.726 {\csname#1@attr@\bbl@attr\endcsname}%
12.727 {\@attrerr{#1}{\bbl@attr}}%
12.728 \fi
12.729 }
12.730 \fi}
```

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

```
12.731 \@onlypreamble\languageattribute
```

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

```
12.732 \newcommand*{\@attrerr}[2]{%
12.733 \PackageError{babel}%
12.734 {The attribute #2 is unknown for language #1.}%
```

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

```
12.736 \def\bbl@declare@ttribute#1#2#3{%
12.737 \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\}. }

```
12.738\expandafter\def\csname#1@attr@#2\endcsname{#3}%
12.739 }
```

\bbl@ifattributeset This internal macro has 4 arguments. It can be used to interpret $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ code based on whether a certain attribute was set. This command should appear inside the argument to $\backslash$ 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

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

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

```
12.741 \ifx\bbl@known@attribs\@undefined
12.742 \in@false
12.743 \else
```

The we need to check the list of known attributes.

```
12.744 \edef\bbl@tempa{\noexpand\in@{,#1-#2,}%
```

12.745 \{,\bbl@known@attribs,\}\}\%
12.746 \bbl@tempa
12.747 \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'.

```
12.748 \ifin@
12.749 \bbl@afterelse#3%
12.750 \else
12.751 \bbl@afterfi#4%
12.752 \fi
12.753 }
```

\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

```
12.754 \def\bbl@add@list#1#2{%
12.755 \ifx#1\@undefined
12.756 \def#1{#2}%
12.757 \else
12.758 \ifx#1\@empty
12.759 \def#1{#2}%
12.760 \else
\edef#1{#1,#2}%
    \fi
    \i
    }
```

\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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$-code to be executed when the attribute is known and the $\mathrm{T}_{\mathrm{E}} \mathrm{X}$-code to be executed otherwise.

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

We first assume the attribute is unknown.

```
12.766\let\bbl@tempa\@secondoftwo
```

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

```
12.767 \@for\bbl@tempb:=#2\do{%
12.768 \expandafter\in@\expandafter{\expandafter,\bbl@tempb,}{,#1,}%
12.769 \ifin@
```

When a match is found the definition of $\backslash \mathrm{bbl@tempa}$ is changed.

```
12.770 \let\bbl@tempa\@firstoftwo
12.771 \else
12.772 \fi}%
```

Finally we execute \bbl@tempa.

```
12.773 \bbl@tempa
```

12.774 \}
\bbl@clear@ttribs This macro removes all the attribute code from EAEX's memory at $\backslash$ begin\{document\} time (if any is present).

```
12.775\def\bbl@clear@ttribs{%
12.776 \ifx\bbl@attributes\@undefined\else
12.777 \@for\bbl@tempa:=\bbl@attributes\do{%
12.778 \expandafter\bbl@clear@ttrib\bbl@tempa.
12.779 }%
12.780 \let\bbl@attributes\@undefined
12.781 \fi
12.782 }
12.783\def\bbl@clear@ttrib#1-#2.{%
12.784 \expandafter\let\csname#1@attr@#2\endcsname\@undefined}
12.785\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.
$\backslash$ babel@beginsave ${ }_{2} .786 \backslash$ def $\backslash$ babel@beginsave $\{\backslash$ babel@savecnt $\backslash z @\}$

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

```
12.787 \newcount\babel@savecnt
12.788 \babel@beginsave
```

\babel＠save The macro \babel＠save〈csname〉 saves the current meaning of the control se－ quence $\langle$ csname $\rangle$ 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．

```
12.789 \def\babel@save#1{%
12.790 \expandafter\let\csname babel@\number\babel@savecnt\endcsname #1\relax
12.791 \begingroup
12.792 \toks@\expandafter{\originalTeX \let#1=}%
12.793 \edef\x{\endgroup
12.794 \def\noexpand\originalTeX{\the\toks@ \expandafter\noexpand
12.795 \csname babel@\number\babel@savecnt\endcsname\relax}}%
12.796 \x
12.797\\advance\babel@savecnt\@ne}
```

\babel＠savevariable The macro \babel＠savevariable〈variable〉 saves the value of the variable． $\langle$ variable〉 can be anything allowed after the \the primitive．

```
12.798 \def \babel@savevariable#1{\begingroup
12.799 \toks@\expandafter{\originalTeX #1=}%
12.800 \edef\x{\endgroup
12.801 \def\noexpand\originalTeX{\the\toks@ \the#1\relax}}%
12.802 \x}
```

$\backslash b b l @ f r e n c h s p a c i n g ~ S o m e ~ l a n g u a g e s ~ n e e d ~ t o ~ h a v e ~ \ f r e n c h s p a c i n g ~ i n ~ e f f e c t . ~ O t h e r s ~ d o n ' t ~ w a n t ~ t h a t . ~$ \bbl＠nonfrenchspacing The command \bbl＠frenchspacing switches it on when it isn＇t already in effect and \bbl＠nonfrenchspacing switches it off if necessary．

```
12.803\def\bbl@frenchspacing{%
12.804 \ifnum\the\sfcode`\.=\@m
12.805 \let\bbl@nonfrenchspacing\relax
12.806 \else
12.807 \frenchspacing
12.808 \let\bbl@nonfrenchspacing\nonfrenchspacing
12.809 \fi}
12.810\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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$－code to be added to the $\langle$ control sequence $\rangle$ ．

If the 〈control sequence〉 has not been defined before it is defined now． 12.811 \def $\backslash$ addto\＃1\＃2\｛\％

[^6]```
12.812 \ifx#1\@undefined
12.813 \def#1{#2}%
12.814 \else
```

The control sequence could also expand to $\backslash r e l a x$ ，in which case a circular defi－ nition results．The net result is a stack overflow．

```
12.815 \ifx#1\relax
12.816 \def#1{#2}%
12.817 \else
```

Otherwise the replacement text for the 〈control sequence〉 is expanded and stored in a token register，together with the $\mathrm{T}_{\mathrm{E}} \mathrm{X}$－code to be added．Finally the 〈control sequence〉 is redefined，using the contents of the token register．

```
12.818 {\toks@\expandafter{#1#2}%
12.819 \xdef#1{\the\toks@}}%
12.820 \fi
12.821 \fi
12.822}
```


## 12．8 Macros common to a number of languages

\allowhyphens This macro makes hyphenation possible．Basically its definition is nothing more than \nobreak \hskip Opt plus Opt ${ }^{7}$ ．

```
12.823\def\bbl@t@one{T1}
12.824 \def\allowhyphens{%
12.825 \ifx\cf@encoding\bbl@t@one\else\bbl@allowhyphens\fi}
12.826\def\bbl@allowhyphens{\nobreak\hskip\z@skip}
```

\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.

```
12.827\def\set@low@box#1{\setbox\tw@\hbox{,}\setbox\z@\hbox{#1}%
12.828 \dimen\z@\ht\z@ \advance\dimen\z@ -\ht\tw@%
12.829 \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．

```
12.830 \def\save@sf@q #1{\leavevmode
12.831 \begingroup
12.832 \edef\@SF{\spacefactor \the\spacefactor}#1\@SF
12.833 \endgroup
12.834}
```

$\backslash$ bbl＠disc For some languages the macro \bbl＠disc is used to ease the insertion of discre－ tionaries for letters that behave＇abnormally＇at a breakpoint．
12.835 \def $\backslash$ bbl＠disc\＃1\＃2\｛\％
12.836 \nobreak\discretionary\｛\＃2－\}\{\}\{\#1\}\allowhyphens\}

[^7]
### 12.9 Making glyphs available

The file babel. $\mathrm{dtx}^{8}$ 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, therefor we make it available by lowering the normal open quote character to the baseline.

```
12.837 \ProvideTextCommand{\quotedblbase}{0T1}{%
12.838 \save@sf@q{\set@low@box{\textquotedblright\/}%
12.839 \box\z@\kern-.04em\allowhyphens}}
```

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

```
12.840 \ProvideTextCommandDefault{\quotedblbase}{%
12.841 \UseTextSymbol{0T1}{\quotedblbase}}
```

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

```
12.842 \ProvideTextCommand{\quotesinglbase}{OT1}{%
12.843 \save@sf@q{\set@low@box{\textquoteright\/}%
12.844 \box\z@\kern-.04em\allowhyphens}}
```

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

```
12.845 \ProvideTextCommandDefault{\quotesinglbase}{%
```

12.846 \UseTextSymbol\{OT1\}\{\quotesinglbase\}\}
\guillemotleft The guillemet characters are not available in OT1 encoding. They are faked.
\guillemotrigh $\ddagger_{2.847}$ \ProvideTextCommand\{\guillemotleft\}\{OT1\}\{\%
12.848 \ifmmode
12.849 \ll
12.850 \else
12.851 \save@sf@q\{\nobreak
$12.852 \backslash$ raise.2ex $\backslash$ hbox $\{\$ \backslash$ scriptscriptstyle $\backslash l l \$\} \backslash a l l o w h y p h e n s\} \%$
12.853 \fi\}
12.854 \ProvideTextCommand\{\guillemotright\}\{0T1\} \{\%
12.855 \ifmmode
$12.856 \quad$ ggg
12.857 \else
12.858 \save@sf@q\{\nobreak
$12.859 \backslash$ raise.2ex $\backslash$ hbox $\{\$ \backslash$ scriptscriptstyle $\backslash g g \$\} \backslash a l l o w h y p h e n s\} \%$
12.860 \fi\}

[^8]Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.
12.861 ProvideTextCommandDefault\{\guillemotleft\}\{\%
12.862 \UseTextSymbol\{OT1\}\{\guillemotleft\}\}
12.863 \ProvideTextCommandDefault\{ \guillemotright\}\{\%
12.864 \UseTextSymbol\{OT1\}\{\guillemotright\}\}
\guilsinglleft The single guillemets are not available in OT1 encoding. They are faked.
\guilsinglrigh $\ddagger_{2.865} \backslash$ ProvideTextCommand $\{\backslash$ guilsinglleft\}\{0T1\}\{\%

```
12.866 \ifmmode
    12.867 <%
    12.868 \else
    12.869 \save@sf@q{\nobreak
    12.870 \raise.2ex\hbox{$\scriptscriptstyle<$}\allowhyphens}%
    12.871 \fi}
    12.872 \ProvideTextCommand{\guilsinglright}{0T1}{%
    12.873 \ifmmode
    12.874 >%
    12.875 \else
    12.876 \save@sf@q{\nobreak
    12.877 \raise.2ex\hbox{$\scriptscriptstyle>$}\allowhyphens}%
    12.878 \fi}
```

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

```
12.879 \ProvideTextCommandDefault{\guilsinglleft}{%
12.880 \UseTextSymbol{OT1}{\guilsinglleft}}
12.881 \ProvideTextCommandDefault{\guilsinglright}{%
12.882 \UseTextSymbol{OT1}{\guilsinglright}}
```


### 12.11 Letters

\ij The dutch language uses the letter ' ij '. It is available in T 1 encoded fonts, but not \IJ in the OT1 encoded fonts. Therefor we fake it for the OT1 encoding.

```
12.883\DeclareTextCommand{\ij}{OT1}{%
12.884 \allowhyphens i\kern-0.02em j\allowhyphens}
12.885\DeclareTextCommand{\IJ}{0T1}{%
12.886 \allowhyphens I\kern-0.02em J\allowhyphens}
12.887 \DeclareTextCommand{\ij}{T1}{\char188}
12.888 \DeclareTextCommand{\IJ}{T1}{\char156}
```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.
12.889 \ProvideTextCommandDefault $\{\backslash i j\}\{\%$
12.890 \UseTextSymbol\{0T1\}\{\ij\}\}
12.891 \ProvideTextCommandDefault $\{\backslash I J\}$ \%
12.892 \UseTextSymbol\{OT1\}\{\IJ\}\}
\dj The croatian language needs the letters \dj and $\backslash \mathrm{DJ}$; they are available in the T 1 \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).
```

```
12.893 \def\crrtic@{\hrule height0.1ex width0.3em}
```

12.893 \def\crrtic@{\hrule height0.1ex width0.3em}
12.894\def\crttic@{\hrule height0.1ex width0.33em}
12.894\def\crttic@{\hrule height0.1ex width0.33em}
12.895 %
12.895 %
12.896 \def\ddj@{%
12.896 \def\ddj@{%
12.897 \setbox0\hbox{d}\dimen@=\ht0
12.897 \setbox0\hbox{d}\dimen@=\ht0
12.898 \advance\dimen@1ex
12.898 \advance\dimen@1ex
12.899 \dimen@.45\dimen@
12.899 \dimen@.45\dimen@
12.900 \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
12.900 \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
12.901 \advance\dimen@ii.5ex
12.901 \advance\dimen@ii.5ex
12.902 \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crrtic@}}}}
12.902 \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crrtic@}}}}
12.903 \def\DDJ@{%
12.903 \def\DDJ@{%
12.904 \setbox0\hbox{D}\dimen@=.55\ht0
12.904 \setbox0\hbox{D}\dimen@=.55\ht0
12.905<br>dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
12.905<br>dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
12.906 \advance\dimen@ii.15ex % correction for the dash position
12.906 \advance\dimen@ii.15ex % correction for the dash position
12.907 \advance\dimen@ii-.15\fontdimen7\font % correction for cmtt font
12.907 \advance\dimen@ii-.15\fontdimen7\font % correction for cmtt font
12.908<br>dimen\thr@@\expandafter\rem@pt\the\fontdimen7\font\dimen@
12.908<br>dimen\thr@@\expandafter\rem@pt\the\fontdimen7\font\dimen@
12.909 \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crttic@}}}}
12.909 \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crttic@}}}}
12.910 %
12.910 %
12.911 \DeclareTextCommand{\dj}{0T1}{\ddj@ d}
12.911 \DeclareTextCommand{\dj}{0T1}{\ddj@ d}
12.912 \DeclareTextCommand{\DJ}{0T1}{\DDJ@ D}

```
12.912 \DeclareTextCommand{\DJ}{0T1}{\DDJ@ D}
```

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

```
12.913 \ProvideTextCommandDefault{\dj}{%
12.914 \UseTextSymbol{OT1}{\dj}}
12.915 \ProvideTextCommandDefault{\DJ}{%
12.916 \UseTextSymbol{OT1}{\DJ}}
```

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

```
12.917\DeclareTextCommand{\SS}{OT1}{SS}
12.918 \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.
\grq2.919 \ProvideTextCommand\{\glq\}\{0T1\}\{\%
12.920 \textormath\{\quotesinglbase\}\{\mbox\{\quotesinglbase\}\}\}
12.921 \ProvideTextCommand\{\glq\}\{T1\}\{\%
12.922 \textormath\{\quotesinglbase\}\{\mbox\{\quotesinglbase\}\}\}
$12.923 \backslash$ ProvideTextCommandDefault\{\glq\}\{\UseTextSymbol\{OT1\}\glq\}
The definition of \grq depends on the fontencoding. With T1 encoding no extra kerning is needed.

```
12.924\ProvideTextCommand{\grq}{T1}{%
12.925 \textormath{\textquoteleft}{\mbox{\textquoteleft}}}
12.926 \ProvideTextCommand{\grq}{0T1}{%
12.927 \save@sf@q{\kern-.0125em%
12.928 \textormath{\textquoteleft}{\mbox{\textquoteleft}}%
12.929 \kern.07em\relax}}
12.930\ProvideTextCommandDefault{\grq}{\UseTextSymbol{0T1}\grq}
```

\glqq The 'german' double quotes
\grqq2. 931 \ProvideTextCommand\{\glqq\}\{0T1\}\{\%
12.932 \textormath\{\quotedblbase\}\{\mbox\{\quotedblbase\}\}\}
$12.933 \backslash$ ProvideTextCommand\{\glqq\}\{T1\}\{\%
12.934 \textormath\{\quotedblbase\}\{\mbox\{\quotedblbase\}\}\}
$12.935 \backslash$ ProvideTextCommandDefault\{\glqq\}\{\UseTextSymbol\{0T1\}\glqq\}

The definition of $\backslash \mathrm{grqq}$ depends on the fontencoding. With T1 encoding no extra kerning is needed
12.936 \ProvideTextCommand\{\grqq\}\{T1\}\{\%
12.937 \textormath\{“\}\{\mbox\{“\}\}\}
12.938 \ProvideTextCommand\{\grqq\}\{0T1\}\{\%
12.939 \save@sf@q\{\kern-.07em\%
12.940 \textormath\{“\}\{\mbox\{“\}\}\%
12.941 \kern.07em\relax\}\}
12.942 \ProvideTextCommandDefault\{\grqq\}\{\UseTextSymbol\{0T1\}\grqq\}
\flq The 'french' single guillemets.
\frq2.943 \ProvideTextCommand\{\flq\}\{0T1\}\{\%
$12.944 \backslash$ textormath\{\guilsinglleft\}\{\mbox\{\guilsinglleft\}\}\}
12.945 \ProvideTextCommand\{\flq\}\{T1\}\{\%
12.946 \textormath\{\guilsinglleft\}\{\mbox\{\guilsinglleft\}\}\}
$12.947 \backslash$ ProvideTextCommandDefault $\{\backslash f l q\}\{\backslash$ UseTextSymbol $\{0 T 1\} \backslash f l q\}$
12.948 \ProvideTextCommand\{\frq\}\{0T1\}\{\%
12.949 \textormath\{\guilsinglright\}\{\mbox\{\guilsinglright\}\}\}
$12.950 \backslash$ ProvideTextCommand\{\frq\}\{T1\}\{\%
12.951 \textormath\{\guilsinglright\}\{\mbox\{\guilsinglright\}\}\}
12.952 \ProvideTextCommandDefault\{\frq\}\{\UseTextSymbol\{OT1\}\frq\}
\flqq The 'french' double guillemets.
\frqqi2. 953 \ProvideTextCommand\{\flqq\}\{0T1\}\{\%
12.954 \textormath\{\guillemotleft\}\{\mbox\{\guillemotleft\}\}\}
$12.955 \backslash$ ProvideTextCommand\{\flqq\}\{T1\}\{\%
12.956 \textormath\{\guillemotleft\}\{\mbox\{\guillemotleft\}\}\}
12.957 \ProvideTextCommandDefault\{\flqq\}\{\UseTextSymbol\{0T1\}\flqq\}
12.958 \ProvideTextCommand\{\frqq\}\{0T1\}\{\%
12.959 \textormath\{\guillemotright\}\{\mbox\{\guillemotright\}\}\}
$12.960 \backslash$ ProvideTextCommand\{\frqq\}\{T1\}\{\%
12.961 \textormath\{\guillemotright\}\{\mbox\{\guillemotright\}\}\}
12.962 \ProvideTextCommandDefault\{\frqq\}\{\UseTextSymbol\{0T1\}\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 \umlautlow the positioning, the default will be \umlauthigh (the normal positioning).

```
12.963 \def\umlauthigh{%
12.964\def\bbl@umlauta##1{\leavevmode\bgroup%
12.965 \expandafter\accent\csname\f@encoding dqpos\endcsname
12.966 ##1\allowhyphens\egroup}%
12.967 \let\bbl@umlaute\bbl@umlauta}
12.968 \def\umlautlow{%
12.969 \def\bbl@umlauta{\protect\lower@umlaut}}
12.970 \def\umlautelow{%
12.971 \def\bbl@umlaute{\protect\lower@umlaut}}
12.972 \umlauthigh
```

\lower@umlaut The command \lower@umlaut is used to position the \" closer the the letter.
We want the umlaut character lowered, nearer to the letter. To do this we need an extra $\langle$ dimen $\rangle$ register.
12.973 \expandafter\ifx\csname U@D\endcsname\relax
12.974 \csname newdimen\endcsname\U@D
12.975 \fi

The following code fools $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 's make_accent procedure about the current x-height of the font to force another placement of the umlaut character.

```
12.976 \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.

```
12.977 \leavevmode\bgroup
12.978 \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 . 45 ex depends on the METAFONT parameters with which the fonts were built. (Just try out, which value will look best.)
12.979 \{ \setbox $\backslash \mathrm{z@} \backslash$ hbox\{\%
$12.980 \backslash$ expandafter $\backslash c h a r \backslash c s n a m e \backslash f @ e n c o d i n g ~ d q p o s \backslash e n d c s n a m e\} \%$
$12.981 \backslash$ dimen@ -.45ex\advance\dimen@ $\backslash h t \backslash z @$
If the new $x$-height is too low, it is not changed.
$12.982 \backslash$ 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.

```
12.983 \expandafter\accent\csname\f@encoding dqpos\endcsname
12.984 \fontdimen5\font\U@D #1%
12.985
\egroup}
```

For all vowels we declare $\backslash "$ 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. Therefor these declarations are postponed until the beginning of the document.

```
12.986 \AtBeginDocument{%
12.987 \DeclareTextCompositeCommand{\"}{OT1}{a}{\bbl@umlauta{a}}%
12.988 \DeclareTextCompositeCommand{\"}{0T1}{e}{\bbl@umlaute{e}}%
12.989 \DeclareTextCompositeCommand{\"}{0T1}{i}{\bbl@umlaute{\i}}%
12.990 \DeclareTextCompositeCommand{\"}{0T1}{\i}{\bbl@umlaute{\i}}%
12.991 \DeclareTextCompositeCommand{\"}{OT1}{o}{\bbl@umlauta{o}}%
12.992 \DeclareTextCompositeCommand{\"}{0T1}{u}{\bbl@umlauta{u}}%
12.993 \DeclareTextCompositeCommand{\"}{OT1}{A}{\bbl@umlauta{A}}%
12.994 \DeclareTextCompositeCommand{\"}{OT1}{E}{\bbl@umlaute{E}}%
12.995 \DeclareTextCompositeCommand{\"}{0T1}{I}{\bbl@umlaute{I}}%
12.996 \DeclareTextCompositeCommand{\"}{0T1}{0}{\bbl@umlauta{0}}%
12.997 \DeclareTextCompositeCommand{\"}{OT1}{U}{\bbl@umlauta{U}}%
12.998}
```


### 12.14 The redefinition of the style commands

The rest of the code in this file can only be processed by $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$, so we check the current format. If it is plain $T_{E} X$, 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 TEX 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.

```
12.999 \{\def \(\backslash\) format \(\{l\) lplain\}
12.1000 \ifx \(\backslash\) fmtname \(\backslash\) format
12.1001 \else
12.1002 \def \(\backslash\) format \(\{\) LaTeX2e\}
12.1003 \ifx\fmtname\format
12.1004 \else
2.1005 \aftergroup \(\backslash e n d i n p u t\)
12.1006 \fi
12.1007 \fi\}
```

Now that we're sure that the code is seen by $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{HA}_{\mathrm{E}} \mathrm{X}$ dated before December 1991. Therefor this part of the code can optionally be included in babel. def by specifying the docstrip option names.

```
12.1008〈*names\rangle
```

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 \chapter and \opening macros are unde- <br> fined |
| :--- | :--- | :--- |
| report and book $:$ | the \chapter macro is defined and the \opening is |
| letter | undefined |
|  | the \chapter macro is undefined and the \opening <br> 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.
$\backslash$ 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 '.

```
12.1009 \ifx\@undefined\doc@style
12.1010 \def\doc@style{0}%
```

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

```
12.1011 \ifx\@undefined\opening
12.1012 \ifx\@undefined\chapter
12.1013 \def\doc@style{1}%
12.1014 \else
12.1015 \def\doc@style{2}%
12.1016 \fi
12.1017 \else
12.1018 \def\doc@style{3}%
12.1019 \fi%
12.1020 \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:
12.1021 \@ifundefined\{figurename\}\{\def \fnum@figure\{\figurename\{\} \thefigure\}\}\{\}
12.1022 \@ifundefined\{tablename\}\{\def \fnum@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.

```
12.1023 \ifcase \doc@style\relax
12.1024 \or
```

This means that babel.def is read after the article style, where no \chapter and \opening commands are defined ${ }^{9}$.

First we have the \tableofcontents, \listoffigures and \listoftables:
12.1025 \@ifundefined\{contentsname\}\%
12.1026 \{\def \tableofcontents\{\section*\{\contentsname \@mkboth
12.1027 \{\uppercase\{\contentsname\}\}\{\uppercase\{\contentsname\}\}\}\%
12.1028 \@starttoc\{toc\}\}\}\{\}
12.1029
12.1030 \@ifundefined\{listfigurename\}\%
$12.1031 \quad\{\backslash d e f \backslash l i s t o f f i g u r e s\{\backslash$ section*\{\listfigurename $@$ @mkboth
12.1032 \{\uppercase\{\listfigurename\}\}\{\uppercase\{\listfigurename\}\}\}\%
12.1033 \@starttoc\{lof\}\}\}\{\}
12.1034
12.1035 \@ifundefined\{listtablename\}\%
$12.1036\{\backslash \operatorname{def} \backslash$ listoftables $\{\backslash$ section*\{\listtablename $@$ @mboth \{\uppercase\{\listtablename\}\}\{\uppercase\{\listtablename\}\}\}\% \@starttoc\{lot\}\}\}\{\}

Then the \thebibliography and \theindex environments.

```
12.1039 \@ifundefined{refname}%
12.1041 \@mkboth{\uppercase{\refname}}{\uppercase{\refname}}}%
12.1042 \list{[\arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
12.1043 \leftmargin\labelwidth
12.1044 \advance\leftmargin\labelsep
12.1045 \usecounter{enumi}}%
12.1046 \def\newblock{\hskip.11em plus.33em minus.07em}%
12.1047 \sloppy\clubpenalty4000\widowpenalty\clubpenalty
12.1048 \sfcode'\.=1000\relax}}{}
12.1049
2.1050 \@ifundefined{indexname}%
12.1051 {\def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
2.1052 \columnseprule \z@
2.1053 \columnsep 35pt\twocolumn[\section*{\indexname}]%
12.1054 \@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
12.1055 \thispagestyle{plain}%
12.1056 \parskip\z@ plus.3pt\parindent\z@\let\item\@idxitem}}{}
```

The abstract environment:
12.1057 \@ifundefined\{abstractname\}\%
12.1058 \{\def \abstract\{\if@twocolumn
12.1059 \section*\{\abstractname\}\%

[^9]```
12.1060 \else \small
12.1061 \begin{center}%
2.1062 {\bf \abstractname\vspace{-.5em}\vspace{\z@}}%
12.1063 \end{center}%
12.1064 \quotation
12.1065 \fi}}{}
```

And last but not least, the macro \part:
12.1066 \@ifundefined\{partname\}\%
$12.1067\{\backslash \operatorname{def} \backslash @ p a r t[\# 1] \# 2\{\backslash$ ifnum $\backslash c @ s e c n u m d e p t h>\backslash m @ n e$
12.1068 \refstepcounter\{part\}\%
12.1069 \addcontentsline\{toc\}\{part\}\{\thepart
$12.1070 \backslash$ hspace $\{1 \mathrm{em}\} \# 1\} \backslash e l$ se
12.1071 \addcontentsline\{toc\}\{part\}\{\#1\}\fi
$12.1072 \quad$ \{ $\backslash$ parindent $\backslash$ ©@ \raggedright
12.1073 \ifnum \c@secnumdepth $>\backslash m @ n e$
$12.1074 \backslash$ Large \bf \partname\{\} \thepart
12.1075 \par \nobreak
12.1076 \fi
12.1077 \huge $\backslash b f$
12.1078 \#2 \markboth\{\}\{\} \par\}\%
12.1079 \nobreak
12.1080 \vskip 3ex\@afterheading\}\%
$12.1081\}\}$
This is all that needs to be done for the article style.
12.1082 \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:

```
12.1083 \@ifundefined{contentsname}%
12.1084 {\def\tableofcontents{\@restonecolfalse
12.1085 \if@twocolumn\@restonecoltrue\onecolumn
12.1086 \fi\chapter*{\contentsname\@mkboth
12.1087 {\uppercase{\contentsname}}{\uppercase{\contentsname}}}%
12.1088 \@starttoc{toc}%
12.1089 \csname if@restonecol\endcsname\twocolumn
12.1090 \csname fi\endcsname}}{}
12.1091
12.1092 \@ifundefined{listfigurename}%
12.1093 {\def\listoffigures{\@restonecolfalse
12.1094 \if@twocolumn\@restonecoltrue\onecolumn
12.1095 \fi\chapter*{\listfigurename\@mkboth
12.1096 {\uppercase{\listfigurename}}{\uppercase{\listfigurename}}}%
12.1097 \@starttoc{lof}%
12.1098 \csname if@restonecol\endcsname\twocolumn
12.1099 \csname fi\endcsname}}{}
12.1100
```

```
12.1101 \@ifundefined{listtablename}%
12.1102 {\def\listoftables{\@restonecolfalse
12.1103 \if@twocolumn\@restonecoltrue\onecolumn
12.1104 \fi\chapter*{\listtablename\@mkboth
12.1105 {\uppercase{\listtablename}}{\uppercase{\listtablename}}}%
12.1106 \@starttoc{lot}%
12.1107 \csname if@restonecol\endcsname\twocolumn
12.1108 \csname fi\endcsname}}{}
```

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.

```
12.1109 \@ifundefined{bibname}%
12.1110 {\def\thebibliography#1{\chapter*{\bibname
12.1111 \@mkboth{\uppercase{\bibname}}{\uppercase{\bibname}}}%
12.1112 \list{[\arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
12.1113 \leftmargin\labelwidth \advance\leftmargin\labelsep
12.1114 \usecounter{enumi}}%
12.1115 \def\newblock{\hskip.11em plus.33em minus.07em}%
12.1116 \sloppy\clubpenalty4000\widowpenalty\clubpenalty
12.1117 \sfcode'\.=1000\relax}}{}
12.1118
12.1119 \@ifundefined{indexname}%
12.1120 {\def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
12.1121 \columnseprule \z@
12.1122 \columnsep 35pt\twocolumn[\@makeschapterhead{\indexname}]%
2.1123 \@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
12.1124 \thispagestyle{plain}%
12.1125 \parskip\z@ plus.3pt\parindent\z@ \let\item\@idxitem}}{}
```

Here is the abstract environment:

```
12.1126 \@ifundefined{abstractname}%
12.1127 {\def\abstract{\titlepage
2.1128 \null\vfil
12.1129 \begin{center}%
12.1130 {\bf \abstractname}%
12.1131 \end{center}}}{}
```

And last but not least the \chapter, \appendix and \part macros.
12.1132 \@ifundefined\{chaptername\}\{\def \@chapapp\{\chaptername\}\}\{\}
12.1133 \%
12.1134 \@ifundefined\{appendixname\}\%
$12.1135 \quad\{\backslash d e f \backslash$ appendix\{\par
12.1136 \setcounter\{chapter\}\{0\}\%
12.1137 \setcounter\{section\}\{0\}\%
12.1138 \def \@chapapp\{\appendixname\}\%
$12.1139 \backslash$ def $\backslash$ thechapter $\{\backslash$ Alph\{chapter\}\}\}\}\{\}
12.1140 \%
12.1141 \@ifundefined\{partname\}\%

```
12.1142 {\def\@part[#1]#2{\ifnum \c@secnumdepth >-2\relax
12.1143 \refstepcounter{part}%
12.1144 \addcontentsline{toc}{part}{\thepart
12.1145 \hspace{1em}#1}\else
12.1146 \addcontentsline{toc}{part}{#1}\fi
12.1147 \markboth{}{}%
12.1148 {\centering
12.1149 \ifnum \c@secnumdepth >-2\relax
12.1150 \huge\bf \partname{} \thepart
12.1151 \par
12.1152 \vskip 20pt \fi
12.1153 \Huge \bf
12.1154 #1\par}\@endpart}}{}%
12.1155 \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．

```
12.1156 \@ifundefined{ccname}%
12.1157 {\def\cc#1{\par\noindent
12.1158 \parbox[t]{\textwidth}%
12.1159 {\@hangfrom{\rm \ccname : }\ignorespaces #1\strut}\par}}{}
12.1160
12.1161 \@ifundefined{enclname}%
12.1162 {\def\encl#1{\par\noindent
12.1163 \parbox[t]{\textwidth}%
12.1164 {\@hangfrom{\rm \enclname : }\ignorespaces #1\strut}\par}}{}
```

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

```
12.1165 \@ifundefined{headtoname}%
12.1166 {\def\ps@headings{%
12.1167 \def\@oddhead{\sl \headtoname{} \ignorespaces\toname \hfil
12.1168 \@date \hfil \pagename{} \thepage}%
12.1169 \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．

Here ends the code that can be optionally included when a version of $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ is in use that is dated before December 1991.
12.1171 〈／names〉
12.1172 〈／core〉

## 12．15 Cross referencing macros

The ${ }^{A} \mathrm{~T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{Xbook}$ [1] (Appendix D, page 382). The primitive \meaning applied to a token expands to the current meaning of this token. For example, ' $\backslash$ meaning $\backslash \mathrm{A}$ ' with $\backslash \mathrm{A}$ defined as ' $\backslash \mathrm{def} \backslash \mathrm{A} \# 1\{\backslash \mathrm{~B}\}$ ' expands to the characters 'macro: \#1-> ${ }^{\text {B' }}$ with all category codes set to 'other' or 'space'.
$\backslash$ 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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ 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 $\backslash$ bbl@redefine which takes care of this. It creates a new control sequence, lorg@...
12.1173 〈*core | shorthands〉
$12.1174 \backslash$ def $\backslash$ bbl@redefine\#1\{\%
12.1175 \edef \bbl@tempa\{\expandafter\@gobble\string\#1\}\%
12.1176 \expandafter\let\csname org@\bbl@tempa\endcsname\#1
12.1177 \expandafter\def \csname\bbl@tempa\endcsname\}

This command should only be used in the preamble of the document.
12.1178 \@onlypreamble\bbl@redefine
$\backslash$ bbl@redefine@long This version of \babel@redefine can be used to redefine \long commands such as \ifthenelse.
12.1179 \def $\backslash$ bbl@redefine@long\#1\{\%
$12.1180 \backslash$ \edef \bbl@tempa\{\expandafter\@gobble\string\#1\}\%
12.1181 \expandafter\let\csname org@\bbl@tempa\endcsname\#1
12.1182 \expandafter $\backslash$ long $\backslash$ expandafter $\backslash d e f \backslash c s n a m e \backslash b b l @ t e m p a \backslash e n d c s n a m e\}$
12.1183 \@onlypreamble\bbl@redefine@long
$\backslash$ bbl@redefinerobust 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 $\backslash$ protect $\backslash$ foo $_{\sqcup}$. So it is necessary to check whether $\backslash f o_{\sqcup}$ exists.

```
12.1184 \def\bbl@redefinerobust#1{%
12.1185 \edef\bbl@tempa{\expandafter\@gobble\string#1}%
12.1186 \expandafter\ifx\csname \bbl@tempa\space\endcsname\relax
12.1187 \expandafter\let\csname org@\bbl@tempa\endcsname#1
12.1188 \expandafter\edef\csname\bbl@tempa\endcsname{\noexpand\protect
12.1189 \expandafter\noexpand\csname\bbl@tempa\space\endcsname}%
```

```
12.1190 \else
12.1191 \expandafter\let\csname org@\bbl@tempa\expandafter\endcsname
12.1192 \csname\bbl@tempa\space\endcsname
12.1193 \fi
```

The result of the code above is that the command that is being redefined is always robust afterwards. Therefor all we need to do now is define $\backslash f \mathrm{fo}_{\sqcup}$.

### 12.1194 \expandafter \def \csname\bbl@tempa\space\endcsname\}

This command should only be used in the preamble of the document.
12.1195 \@onlypreamble\bbl@redefinerobust
\newlabel The macro \label writes a line with a \newlabel command into the .aux file to define labels.

```
12.1196 %\bbl@redefine\newlabel#1#2{%
12.1197% \@safe@activestrue\org@newlabel{#1}{#2}\@safe@activesfalse}
```

\@newl@bel We need to change the definition of the $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$-internal macro \@newl@bel. This is needed because we need to make sure that shorthand characters expand to their non-active version.

```
12.1198 \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.

```
12.1199 {%
12.1200 \@safe@activestrue
12.1201 \@ifundefined{#1@#2}%
12.1202 \relax
12.1203 {%
12.1204 \gdef \@multiplelabels {%
12.1205 \@latex@warning@no@line{There were multiply-defined labels}}%
12.1206 \@latex@warning@no@line{Label '#2' multiply defined}%
12.1207 }%
12.1208\\global\@namedef{#1@#2}{#3}%
12.1209 }%
12.1210 }
```

\@testdef An internal IATEX 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 $\backslash \# 1 @ \# 2$ contains the same characters as the \#3; but the character codes differ. Therefor $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ keeps reporting that the labels may have changed.

```
12.1211 \CheckCommand*\@testdef[3]{%
12.1212\def\reserved@a{#3}%
12.1213 \expandafter \ifx \csname #1@#2\endcsname \reserved@a
12.1214 \else
12.1215 \@tempswatrue
12.1216 \fi}
```

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

```
12.1217 \def\@testdef #1#2#3{%
12.1218 \@safe@activestrue
```

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

```
12.1219 \expandafter\let\expandafter\bbl@tempa\csname #1@#2\endcsname
```

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

```
12.1220\\def\bbl@tempb{#3}%
12.1221 \@safe@activesfalse
```

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

```
12.1222 \ifx\bbl@tempa\relax
12.1223 \else
12.1224 \edef\bbl@tempa{\expandafter\strip@prefix\meaning\bbl@tempa}%
12.1225 \fi
```

We do the same for $\backslash \mathrm{bbl@tempb}$.
$12.1226 \backslash e d e f \backslash b b l @ t e m p b\{\backslash e x p a n d a f t e r \backslash s t r i p @ p r e f i x \backslash m e a n i n g \backslash b b l @ t e m p b\} \%$

If the label didn't change, $\backslash$ bbl@tempa and $\backslash \mathrm{bbl@t}$ empb should be identical macros.
12.1227 \ifx \bbl@tempa \bbl@tempb
12.1228 \else
12.1229 \@tempswatrue
12.1230 \fi\}
$\backslash$ ref The same holds for the macro \ref that references a label and \pageref to refer\pageref ence 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.

```
12.1231 \bbl@redefinerobust\ref#1{%
12.1232 \@safe@activestrue\org@ref{#1}\@safe@activesfalse}
12.1233\bbl@redefinerobust\pageref#1{%
12.1234 \@safe@activestrue\org@pageref{#1}\@safe@activesfalse}
```

\@citex The macro used to cite from a bibliography, \cite, uses an internal macro, $\backslash @ c i t e x$. It is this internal macro that picks up the $\operatorname{argument(s),~so~we~rede-~}$ fine 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.

```
12.1235\bbl@redefine\@citex[#1]#2{%
12.1236 \@safe@activestrue\edef\@tempa{#2}\@safe@activesfalse
12.1237 \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.
$12.1238 \backslash$ AtBeginDocument $\{\%$
12.1239 \@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).

```
12.1240 \def\@citex[#1] [#2] #3{%
12.1241 \@safe@activestrue\edef\@tempa{#3}\@safe@activesfalse
12.1242 \org@@citex[#1][#2]{\@tempa}}%
12.1243 }{}}
```

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

```
12.1244 \AtBeginDocument{%
12.1245 \@ifpackageloaded{cite}{%
12.1246 \def\@citex[#1]#2{%
12.1247 \@safe@activestrue\org@@citex[#1]{#2}\@safe@activesfalse}%
12.1248 }{}}
```

$\backslash$ nocite The macro \nocite which is used to instruct $\mathrm{BiBT}_{\mathrm{E}} \mathrm{X}$ to extract uncited references from the database.

```
12.1249 \bbl@redefine\nocite#1{%
12.1250 \@safe@activestrue\org@nocite{#1}\@safe@activesfalse}
```

\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.

```
12.1251 \bbl@redefine\bibcite{%
```

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

```
12.1252 \bbl@cite@choice
```

12.1253 \bibcite\}
\bbl@bibcite The macro \bbl@bibcite holds the definition of \bibcite needed when neither natbib nor cite is loaded
12.1254 \def \bbl@bibcite\#1\#2\{\%
12.1255 \org@bibcite\{\#1\}\{\@safe@activesfalse\#2\}\}
$\backslash b b l @ c i t e @ c h o i c e ~ T h e ~ m a c r o ~ \ b b l @ c i t e @ c h o i c e ~ d e t e r m i n e s ~ w h i c h ~ d e f i n i t i o n ~ o f ~ \ b i b c i t e ~ i s ~ n e e d e d . ~$ 12.1256 \def \bbl@cite@choice\{\%

First we give \bibcite its default definition. 12.1257 \global\let\bibcite\bbl@bibcite

Then, when natbib is loaded we restore the original definition of $\backslash$ bibcite . 12.1258 \@ifpackageloaded\{natbib\}\{\global\let\bibcite\org@bibcite\}\{\}\%

For cite we do the same.

```
12.1259 \@ifpackageloaded{cite}{\global\let\bibcite\org@bibcite}{}%%
```

Make sure this only happens once.

```
12.1260 \global\let\bbl@cite@choice\relax
12.1261 }
```

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

```
12.1262\AtBeginDocument{\bbl@cite@choice}
```

\@bibitem One of the two internal $\mathrm{LA}_{\mathrm{E}} \mathrm{X}$ macros called by \bibitem that write the citation label on the .aux file.

```
12.1263\bbl@redefine\@bibitem#1{%
12.1264 \@safe@activestrue\org@@bibitem{#1}\@safe@activesfalse}
```


### 12.16 marks

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

```
12.1265 \bbl@redefine\markright#1{%
```

First of all we temporarily store the language switching command, using an expanded definition in order to get the current value of \languagename.

```
12.1266 \edef\bbl@tempb{\noexpand\protect
12.1267 \noexpand\foreignlanguage{\languagename}}%
```

Then, we check whether the argument is empty; if it is, we just make sure the scratch token register is empty.

```
12.1268 \def\bbl@arg{#1}%
12.1269 \ifx\bbl@arg\@empty
12.1270 \toks@{}%
12.1271 \else
```

Next, we store the argument to \markright in the scratch token register, together with the expansion of $\backslash \mathrm{bbl@t}$ empb (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@activestrue is in effect.

```
12.1272 \expandafter\toks@\expandafter{%
12.1273 \bbl@tempb{\protect\bbl@restore@actives#1}}%
12.1274 \fi
```

Then we define a temporary control sequence using \edef.

```
12.1275 \edef\bbl@tempa{%
```

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

```
12.1276 \noexpand\org@markright{\the\toks@}}%
12.1277 \bbl@tempa
12.1278 }
```

The definition of \markboth is equivalent to that of \markright, except that we need two token registers.

```
12.1279\bbl@redefine\markboth#1#2{%
12.1280 \edef\bbl@tempb{\noexpand\protect
12.1281 \noexpand\foreignlanguage{\languagename}}%
12.1282 \def\bbl@arg{#1}%
12.1283 \ifx\bbl@arg\@empty
12.1284 \toks@{}%
12.1285 \else
12.1286 \expandafter\toks@\expandafter{%
12.1287 \bbl@tempb{\protect\bbl@restore@actives#1}}%
12.1288 \fi
12.1289 \def\bbl@arg{#2}%
12.1290 \ifx\bbl@arg\@empty
12.1291 \toks8{}%
12.1292 \else
12.1293 \expandafter\toks8\expandafter{%
12.1294 \bbl@tempb{\protect\bbl@restore@actives#2}}%
12.1295 \fi
12.1296 \edef\bbl@tempa{%
12.1297 \noexpand\org@markboth{\the\toks@}{\the\toks8}}%
12.1298 \bbl@tempa
12.1299}
12.1300 </core | shorthands>
```


### 12.17 Encoding issues (part 2)

$\backslash$ TeX Because documents may use font encodings other than one of the latin encodings, \LaTeX we make sure that the logos of $\mathrm{TEX}_{\mathrm{E}}$ and $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ always come out in the right encoding.

```
12.1301 <*core\rangle
12.1302 \bbl@redefine\TeX{\textlatin{\org@TeX}}
12.1303\bbl@redefine\LaTeX{\textlatin{\org@LaTeX}}
12.1304 </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. }

```
12.1305 <*package>
12.1306 \AtBeginDocument{%
12.1307 \@ifpackageloaded{ifthen}{%
```

    Then we can redefine \ifthenelse:
    12.1308 \bbl@redefine@long\ifthenelse\#1\#2\#3\{\%

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

```
12.1309 \let\bbl@tempa\pageref
12.1310 \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.

| 12.1311 | \@safe@activestrue |
| :--- | :--- |
| 12.1312 | \org@ifthenelse\{\#1\}\{\% |
| 12.1313 | \let $\backslash$ pageref $\backslash$ bbl@tempa |
| 12.1314 | \@safe@activesfalse |
| 12.1315 | \#2\}\{\% |
| 12.1316 | \let $\backslash$ pageref $\backslash$ bbl@tempa |
| 12.1317 | \@safe@activesfalse |
| 12.1318 | \#3\}\% |
| 12.1319 | $\} \%$ |

When the package wasn't loaded we do nothing.

```
12.1320 }{}%
12.1321 }
```

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

```
12.1322 \AtBeginDocument{%
12.1323 \@ifpackageloaded{varioref}{%
12.1324 \bbl@redefine\@@vpageref#1[#2]#3{%
12.1325 \@safe@activestrue
```

```
12.1326 \org@@@vpageref{#1}[#2]{#3}%
12.1327 \@safe@activesfalse}%
```

The same needs to happen for \vrefpagenum.

```
12.1328 \bbl@redefine\vrefpagenum#1#2{%
12.1329 \@safe@activestrue
12.1330 \org@vrefpagenum{#1}{#2}%
12.1331 \@safe@activesfalse}%
```

The package varioref defines \Ref to be a robust command wich uppercases the first character of the reference text. In order to be able to do that it needs to access the exandable form of \ref. So we employ a little trick here. We redefine the (internal) command $\backslash \operatorname{Ref}_{\sqcup}$ to call \org@ref instead of $\backslash r e f$. The disadvantgage of this solution is that whenever the derfinition of $\backslash$ Ref changes, this definition needs to be updated as well.

```
12.1332 \expandafter\def\csname Ref \endcsname#1{%
12.1333 \protected@edef\@tempa{\org@ref{#1}}\expandafter\MakeUppercase\@tempa}
12.1334 }{}%
12.1335 }
```


### 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. Therefor we need to reload the package when the ' $:$ ' is an active character.

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

```
12.1336 \AtBeginDocument{%
12.1337 \@ifpackageloaded{hhline}%
```

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

```
12.1338 {\expandafter\ifx\csname normal@char\string:\endcsname\relax
12.1339 \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.

| 12.1340 | $\backslash$ makeatletter |
| :--- | :---: |
| 12.1341 | $\backslash$ def $\backslash$ @currname\{hhline $\} \backslash$ input\{hhline.sty $\} \backslash$ makeatother |
| 12.1342 | $\backslash f i\} \%$ |
| 12.1343 | $\}\}$ |

### 12.18.4 General

$\backslash$ FOREIGNLANGUAGE The package fancyhdr treats the running head and fout 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 $\backslash$ FOREIGNLANGUAGE here.

```
12.1344 \DeclareRobustCommand{\FOREIGNLANGUAGE}[1] {%
12.1345 \lowercase{\foreignlanguage{#1}}}
12.1346 </package\rangle
```

 middle of processing the document. In order to guard against any characters having the wrong \catcodes it always calls $\backslash n f s s @ c a t c o d e s$ before loading a file. Unfortunately, the characters " and' are not dealt with. Therefor we have to add them until ${ }^{\mathrm{A}} \mathrm{T}_{\mathrm{E}} \mathrm{X}$ does that herself.

```
12.1347 <*core | shorthands\rangle
12.1348\ifx\nfss@catcodes\@undefined
12.1349 \else
12.1350 \addto\nfss@catcodes{%
12.1351 \@makeother\'%
12.1352 \@makeother\"%
12.1353 }
12.1354 \fi
12.1355 </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.

```
\ifx\loadlocalcfg\@undefined
\def\loadlocalcfg#1{%
\InputIfFileExists{#1.cfg}
{\typeout{******************************************-J%
                                    * Local config file #1.cfg used~~ J%
                                    *}%
                                    }
{}}
13.9 <
```

13.10 \fi

Just to be compatible with $\mathrm{EAT}_{\mathrm{E}} 2.09$ 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
```

$13.18 \backslash$ edef $\backslash$ reserved@a\{\write\#1\{\#3\}\}\%
13.19
13.20
$\backslash$ reserved@a
\endgroup
$\backslash i f @$ nobreak $\backslash i f v m o d e \backslash n o b r e a k \backslash f i \backslash f i$
13.21
13.22
$13.23 \backslash f i$
13.24 〈/core〉

## 14 Driver files for the documented source code

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 $\mathrm{LA}_{\mathrm{E}} \mathrm{X}$ to get a user guide.

```
14.1〈*driver〉
14.2 \documentclass{ltxdoc}
14.3 \usepackage{url,t1enc,supertabular}
    \usepackage[icelandic,english] {babel}
    \DoNotIndex{\!,\',\, ,\.,\-,\:,\;,\?,\/,\`,\`,\@M}
    \ \DoNotIndex{\@,\@ne,\@m, \@afterheading,\@date,\@endpart}
    \DoNotIndex{\@hangfrom,\@idxitem, \@makeschapterhead,\@mkboth}
    \DoNotIndex{\@oddfoot,\@oddhead,\@restonecolfalse,\@restonecoltrue}
    \DoNotIndex{\@starttoc,\@unused}
    \DoNotIndex{\accent,\active}
    \DoNotIndex{\addcontentsline,\advance,\Alph,\arabic}
    \DoNotIndex{\baselineskip,\begin,\begingroup,\bf,\box,\c@secnumdepth}
    \DoNotIndex{\catcode,\centering,\char,\chardef,\clubpenalty}
    \DoNotIndex{\columnsep,\columnseprule,\crcr,\csname}
    \DoNotIndex{\day,\def,\dimen,\discretionary,\divide,\dp,\do}
    \DoNotIndex{\edef,\else,\@empty,\end,\endgroup,\endcsname,\endinput}
    \DoNotIndex{\errhelp,\errmessage,\expandafter,\fi,\filedate}
    \DoNotIndex{\fileversion,\fmtname,\fnum@figure,\fnum@table,\fontdimen}
    \DoNotIndex{\gdef,\global}
    \DoNotIndex{\hbox,\hidewidth,\hfil,\hskip,\hspace,\ht,\Huge,\huge}
    \DoNotIndex{\ialign,\if@twocolumn,\ifcase,\ifcat,\ifhmode,\ifmmode}
    \DoNotIndex{\ifnum,\ifx,\immediate,\ignorespaces,\input,\item}
    \DoNotIndex{\kern}
    \DoNotIndex{\labelsep,\Large,\large,\labelwidth,\lccode,\leftmargin}
    \DoNotIndex{\lineskip,\leavevmode,\let,\list,\ll, \long,\lower}
    \DoNotIndex{\m@ne,\mathchar,\mathaccent,\markboth,\month,\multiply}
    \DoNotIndex{\newblock,\newbox,\newcount,\newdimen,\newif,\newwrite}
    \DoNotIndex{\nobreak, \noexpand, \noindent, \null, \number}
    \DoNotIndex{\onecolumn,\or}
    \DoNotIndex{\p@,par, \parbox,\parindent,\parskip,\penalty}
    \DoNotIndex{\protect,\ps@headings}
4.32\DoNotIndex{\quotation}
14.33\DoNotIndex{\raggedright,\raise,\refstepcounter,\relax,\rm,\setbox}
14.34\DoNotIndex{\section,\setcounter,\settowidth,\scriptscriptstyle}
14.35\DoNotIndex{\sfcode,\sl,\sloppy,\small,\space,\spacefactor,\strut}
14.36 \DoNotIndex{\string}
14.37\DoNotIndex{\textwidth,\the,\thechapter,\thefigure,\thepage,\thepart}
14.38\DoNotIndex{0ble,\thispagestyle,\titlepage,\tracingmacros}
14.39 \DoNotIndex{\tw@,\twocolumn,\typeout,\uppercase,\usecounter}
14.40\DoNotIndex{\vbox,\vfil,\vskip,\vspace,\vss}
14.41 \DoNotIndex{\widowpenalty,\write,\xdef,\year,\z@,\z@skip}
```

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

```
14.42 \makeatletter
14.43\gdef\dlqq{{\setbox\tw@=\hbox{,}\setbox\z@=\hbox{',}%
14.44 \dimen\z@=\ht\z@ \advance\dimen\z@-\ht\tw@
14.45\setbox\z@=\hbox{\lower\dimen\z@\box\z@}\ht\z@=\ht\tw@
14.46\\dp\z@=\dp\tw@ \box\z@\kern-.04em}}
```

The code lines are numbered within sections,

```
14.47 <*!user\rangle
14.48 \@addtoreset{CodelineNo}{section}
14.49 \renewcommand\theCodelineNo{%
14.50\reset@font\scriptsize\thesection.\arabic{CodelineNo}}
```

which should also be visible in the index; hence this redefinition of a macro from doc.sty.
14.51 }

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 $* \backslash M F\{\{\backslash$ manual META $\} \backslash-\{\backslash$ manual FONT $\}\}$
14.63 \newcommand $* \backslash$ TeXhax $\{\backslash \mathrm{TeX}$ hax $\}$
14.64 \newcommand $* \backslash$ babel $\{\backslash$ textsf $\{$ babel $\}\}$
14.65 \newcommand* $\backslash$ Babel $\{\backslash$ textsf $\{$ Babel $\}\}$
$14.66 \backslash$ newcommand $* \backslash m[1]\{\backslash \operatorname{mbox}\{\$ \backslash$ langle\$ 1 it\#1 $\backslash / \$ \backslash$ 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 $\backslash$ bslash\}
14.71 \newcommand*\Lopt[1]\{\textsf\{\#1\}\}
$14.72 \backslash$ newcommand $* \backslash \operatorname{Lenv}[1]\{\backslash$ textsf $\{\# 1\}\}$
14.73 \newcommand*\file[1]\{\texttt\{\#1\}\}
14.74 \newcommand* $\backslash$ cls [1]\{\texttt $\{\# 1\}\}$
14.75 \newcommand* $\backslash \operatorname{pkg}[1]\{\backslash$ texttt $\{\# 1\}\}$
$14.76 \backslash$ newcommand*\langdeffile[1] \{\%
14.77 〈-user〉 

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

Inlude 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{croatian.dtx}
14.124 \langdeffile{czech.dtx}
14.125 \langdeffile{polish.dtx}
14.126 \langdeffile{serbian.dtx}
14.127 \langdeffile{slovak.dtx}
14.128 \langdeffile{slovene.dtx}
14.129 \langdeffile{russianb.dtx}
14.130 \langdeffile{bulgarian.dtx}
14.131 \langdeffile{ukraineb.dtx}
14.132 %
14.133 \langdeffile{lsorbian.dtx}
14.134\langdeffile{usorbian.dtx}
14.135 \langdeffile{turkish.dtx}
14.136 %
14.137\langdeffile{hebrew.dtx}
14.138 \DocInput{hebinp.dtx}
14.139 \DocInput{hebrew.fdd}
14.140\DocInput{heb209.dtx}
14.141 \langdeffile{bahasa.dtx}
14.142 %\langdeffile{sanskrit.dtx}
14.143 %\langdeffile{kannada.dtx}
14.144 %\langdeffile{nagari.dtx}
14.145 %\langdeffile{tamil.dtx}
14.146 \clearpage
14.147 \DocInput{bbplain.dtx}
```

Finally print the index and change log（not for the user guide）．
14.148 〈＊！user〉
14.149 

$14.150 \backslash d e f \backslash f i l e n a m e\{i n d e x\}$
14．151 \PrintIndex
14.152 

$14.153 \backslash d e f \backslash f i l e n a m e\{c h a n g e s\}$
14．154 \PrintChanges
14.155 〈／！user〉
14.156 \end\｛document\}
14.157 〈／driver〉

## 15 Conclusion

A system of document options has been presented that enable the user of $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ to adapt the standard document classes of $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ users as well as to $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$ 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 $\beta$-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.

## References

[1] Donald E. Knuth, The $T_{E} X b o o k$, Addison-Wesley, 1986.
[2] Leslie Lamport, $L^{A} T_{E} X, A$ 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 $T_{E} X$, TUGboat 9 (1988) \#1, p. 70-72.
[5] Leslie Lamport, in: $T_{E}$ Xhax Digest, Volume 89, \#13, 17 February 1989.
[6] Johannes Braams, Victor Eijkhout and Nico Poppelier, The development of national $L^{A} T_{E} X$ styles, TUGboat 10 (1989) \#3, p. 401-406.
[7] Joachim Schrod, International $L^{A} T_{E} X$ is ready to use, TUGboat 11 (1990) \#1, p. 87-90.

## 17 The Esperanto language

The file esperanto.dtx ${ }^{10}$ defines all the language-specific macros for the Esperanto language.

For this language the character ${ }^{\text {- }}$ is made active. In table 2 an overview is given of its purpose.

- c gives $\hat{c}$ with hyphenation in the rest of the word allowed, this works for c, C, g, G, H, J, s, S, z, Z
${ }^{\circ} \mathrm{h}$ prevents hî from becoming too tall
*j gives $\hat{\jmath}$
${ }^{\circ} \mathrm{u}$ gives $\breve{\mathrm{u}}$, with hyphenation in the rest of the word allowed
${ }^{\wedge} \mathrm{U}$ gives $\breve{\mathrm{U}}$, with hyphenation in the rest of the word allowed
- 1 inserts a \discretionary\{-\}\{\}\{\}

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.
17.1 〈*code〉
17.2 \LdfInit\{esperanto\} \captionsesperanto

When this file is read as an option, i.e. by the command,esperantowillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@esperantotoseewhetherwehavetodosomethinghere.17.3\ifx\1@esperanto\@undefined17.4\@nopatterns\{Esperanto\}17.5\adddialect\1@esperanto0\fiundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

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.
\captionsesperanto The macro \captionsesperanto defines all strings used in the four standard documentclasses provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
17.6 \addto\captionsesperanto{%
17.7 \def\prefacename{Anta\u{u}parolo}%
17.8\def\refname{Cita\^\j{}oj}%
17.9\def\abstractname{Resumo}%
17.10 \def\bibname{Bibliografio}%
17.11 \def\chaptername{{\`C}apitro}%
17.12\def\appendixname{Apendico}%
```

[^10]```
17.13\\def\contentsname{Enhavo}%
17.14 \def\listfigurename{Listo de figuroj}%
7.15\def\listtablename{Listo de tabeloj}%
17.16 \def\indexname{Indekso}%
17.17\def\figurename{Figuro}%
17.18\def\tablename{Tabelo}%
17.19\def\partname{Parto}%
17.20\\def\enclname{Aldono(j)}%
17.21 \def\ccname{Kopie al}%
17.22 \def\headtoname{Al}%
17.23\def\pagename{Pa\`go}%
17.24\def\subjectname{Temo}%
17.25\def\seename{vidu}% a^u: vd.
17.26\def\alsoname{vidu anka\u{u}}% a^u vd. anka\u{u}
17.27 \def\proofname{Pruvo}%
17.28\def\glossaryname{Glosaro}%
17.29 }
```

\dateesperanto The macro \dateesperanto redefines the command \today to produce Esperanto dates.

```
17.30\def\dateesperanto{%
17.31 \def\today{\number\day{--a}~ de~\ifcase\month\or
17.32 januaro\or februaro\or marto\or aprilo\or majo\or junio\or
17.33 julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
17.34 decembro\fi,\space \number\year}}
```

\extrasesperanto The macro \extrasesperanto performs all the extra definitions needed for the \noextrasesperanto 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.

```
17.35 \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 ${ }^{\wedge}$. It checks whether math mode is active; if so the shorthand mechanism is bypassed by a direct call of \normal@char^.

```
17.36 \addto\extrasesperanto{\languageshorthands{esperanto}}
17.37\addto\extrasesperanto{\bbl@activate{^}}
17.38 \addto\noextrasesperanto{\bbl@deactivate{^}}
```

In order to prevent problems with the active ^ we add a shorthand on system level which expands to a 'normal ^.

```
17.39 \declare@shorthand{system}{^}{\csname normal@char\string^\endcsname}
```

And here are the uses of the active ${ }^{\sim}$ :

```
17.40\declare@shorthand{esperanto}{^c}{\\{c}\allowhyphens}
17.41 \declare@shorthand{esperanto}{~C}{\^{C}\allowhyphens}
17.42 \declare@shorthand{esperanto}{^g}{\^{g}\allowhyphens}
```

```
17.43\declare@shorthand{esperanto}{^G}{\^{G}\allowhyphens}
17.44 \declare@shorthand{esperanto}{^h}{h\llap{\^{}}\allowhyphens}
17.45 \declare@shorthand{esperanto}{^H}{\^{H}\allowhyphens}
17.46 \declare@shorthand{esperanto}{~j}{\^{\j}\allowhyphens}
17.47\declare@shorthand{esperanto}{^J}{\^{J}\allowhyphens}
17.48 \declare@shorthand{esperanto}{^s}{\^{s}\allowhyphens}
17.49 \declare@shorthand{esperanto}{^S}{\^{S}\allowhyphens}
17.50 \declare@shorthand{esperanto}{^u}{\u u\allowhyphens}
17.51 \declare@shorthand{esperanto}{~U}{\u U\allowhyphens}
17.52\declare@shorthand{esperanto}{~|}{\discretionary{-}{}{}\allowhyphens}
```

\Esper In esperant.sty Jörg Knappen provides the macros \esper and \Esper that can
\esper be used instead of $\backslash \mathrm{alph}$ and $\backslash \mathrm{Alph}$. These macros are available in this file as well.

Their definition takes place in two steps. First the toplevel.

```
17.53\def\esper#1{\@esper{\@nameuse{c@#1}}}
17.54\def\Esper#1{\@Esper{\@nameuse{c@#1}}}
```

Then the second level

```
17.55\def\@esper#1{%
17.56 \ifcase#1\or a\or b\or c\or \^c\or d\or e\or f\or g\or \^g\or
17.57 h\or h\llap{\^{}}\or i\or j\or \^\j\or k\or l\or m\or n\or o\or
17.58 p\or s\or \^s\or t\or u\or \u{u}\or v\or z\else\@ctrerr\fi}
17.59\def\@Esper#1{%
17.60 \ifcase#1\or A\or B\or C\or \^C\or D\or E\or F\or G\or \~G\or
17.61 H\or \^H\or I\or J\or \^J\or K\or L\or M\or N\or O\or
17.62 P\or S\or \^S\or T\or U\or \u{U}\or V\or Z\else\@ctrerr\fi}
```

\hodiau In esperant.sty Jörg Knappen provides two alternative macros for \today, \hodiaun \hodiau and \hodiaun. The second macro produces an accusative version of the date in Esperanto.

```
17.63\addto\dateesperanto{\def\hodiau{la \today}}
17.64\def\hodiaun{la \number\day --an~de~\ifcase\month\or
17.65 januaro\or februaro\or marto\or aprilo\or majo\or junio\or
17.66 julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
17.67 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 $\backslash$ begin\{document\} and resetting the category code of © to its original value.

```
17.68\ldf@finish{esperanto}
17.69 </code\
```


## 18 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.

```
18.1 〈*code>
18.2\LdfInit{interlingua}{captionsinterlingua}
```

When this file is read as an option, i.e. by the command,interlinguacouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@interlinguatoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
18.3\ifx\undefined\l@interlingua
18.4 \@nopatterns{Interlingua}
18.5 \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 $\backslash l e f t h y p h e n m i n ~ a n d ~ \ r i g h t h y p h e n m i n . ~$
$18.6 \backslash$ providehyphenmins\{interlingua\} $\left\{\right.$ tww@ ${ }^{\text {tw@ }\}}$
\captionsinterlingua The macro \captionsinterlingua defines all strings used in the four standard documentclasses provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
18.7\def\captionsinterlingua{%
18.8\\def\prefacename{Prefacio}%
18.9 \def\refname{Referentias}%
18.10\\def\abstractname{Summario}%
18.11 \def\bibname{Bibliographia}%
18.12\def\chaptername{Capitulo}%
18.13\def\appendixname{Appendice}%
18.14\def\contentsname{Contento}%
18.15 \def\listfigurename{Lista de figuras}%
18.16 \def\listtablename{Lista de tabellas}%
18.17\\def\indexname{Indice}%
18.18\\def\figurename{Figura}%
18.19\def\tablename{Tabella}%
```

[^11]```
18.20\\def\partname{Parte}%
18.21 \def\enclname{Incluso}%
18.22 \def\ccname{Copia}%
18.23\def\headtoname{A}%
18.24\def\pagename{Pagina}%
18.25\def\seename{vide}%
18.26\def\alsoname{vide etiam}%
18.27 \def\proofname{Prova}%
18.28 \def\glossaryname{Glossario}%
18.29 }
```

\dateinterlingua The macro \dateinterlingua redefines the command \today to produce Interlingua dates.

```
18.30 \def \(\backslash\) dateinterlingua\{\%
```



```
18.32 januario\or februario\or martio\or april\or maio\or junio\or
18.33 julio\or augusto\or septembre\or octobre\or novembre\or
18.34 decembre\fi
18.35 \space \number\year\}\}
```

\extrasinterlingua The macro \extrasinterlingua will perform all the extra definitions needed for \noextrasinterlingua 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.

```
18.36 \addto\extrasinterlingua{}
18.37\addto\noextrasinterlingua{}
```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of @ to its original value.
$18.38 \backslash l d f @ f i n i s h\{i n t e r l i n g u a\}$
18.39 〈/code〉

## 19 The Dutch language

The file dutch.dtx ${ }^{12}$ defines all the language-specific macros for the Dutch language and the 'Afrikaans' version ${ }^{13}$ 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 hyphencharcter without the following \discretionary\{\}\{\}\{\}.
"" to produce an invisible 'breakpoint'.
"' lowered double left quotes (see example below).
", normal double right quotes.
\- like the old $\backslash$-, but allowing hyphenation in the rest of the word.

Table 3: The extra definitions made by dutch.ldf

```
19.1% \changes{dutch-3.8a}{1996/10/04}{made check dependant on
19.2% \cs{CurrentOption}}
19.3%
19.4% The macro |\LdfInit| takes care of preventing that this file is
19.5% loaded more than once, checking the category code of the
19.6% \texttt{@} sign, etc.
19.7% \changes{dutch-3.8a}{1996/10/30}{Now use \cs{LdfInit} to perform
19.8% initial checks}
19.9% \begin{macrocode}
19.10 <*code\rangle
19.11\LdfInit\CurrentOption{captions\CurrentOption}
```

[^12]When this file is read as an option, i.e. by the command,dutchcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@dutchorl@afrikaanstoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

First we try to establish with which option we are being processed.

```
19.12\def\bbl@tempa{dutch}
19.13\ifx\CurrentOption\bbl@tempa
```

If it is dutch then we first check if the Dutch hyphenation patterns wer loaded, $19.14 \backslash i f x \backslash l @ d u t c h \backslash u n d e f i n e d$
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.

```
19.15 \@nopatterns{Dutch}
19.16 \ifx\l@afrikaans\undefined
19.17 \adddialect\l@dutch0
19.18 \else
19.19 \adddialect\l@dutch\l@afrikaans
19.20 \fi
19.21 \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 $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

| 19.22 | $\backslash$ begingroup |
| :---: | :---: |
| 19.23 | \catcode'\"\active |
| 19.24 | $\backslash \mathrm{def} \backslash \mathrm{x}\{\backslash \mathrm{endgroup}$ |
| 19.25 | $\backslash d e f \backslash c a p t i o n s d u t c h\{\%$ |
| 19.26 | \def \prefacename\{Voorwoord\}\% |
| 19.27 | \def \refname\{Referenties\}\% |
| 19.28 | \def \abstractname\{Samenvatting\}\% |
| 19.29 | $\backslash d e f \backslash$ bibname\{Bibliografie\}\% |
| 19.30 | \def \chaptername\{Hoofdstuk\}\% |
| 19.31 | \def\appendixname\{B"ylage\}\% |
| 19.32 | $\backslash d e f \backslash c o n t e n t s n a m e\{$ Inhoudsopgave\}\% |
| 19.33 | \def \listfigurename\{L"yst van figuren\}\% |
| 19.34 | \def \listtablename\{L"yst van tabellen\}\% |
| 19.35 | $\backslash \mathrm{def}$ \indexname\{Index\}\% |
| 19.36 | $\backslash \mathrm{def}$ \figurename\{Figuur\}\% |
| 19.37 | \def \tablename\{Tabel\}\% |
| 19.38 | \def \partname\{Deel\}\% |
| 19.39 | \def \enclname\{B"ylage(n) \}\% |
| 19.40 | $\backslash \mathrm{def}$ \ccname\{cc\}\% |
| 19.41 | $\backslash \mathrm{def} \backslash$ headtoname\{Aan\}\% |
| 19.42 | \def \pagename\{Pagina\}\% |
| 19.43 | $\backslash d e f \backslash$ seename\{zie\}\% |
| 19.44 | \def \alsoname\{zie ook\}\% |
| 19.45 | \def\proofname\{Bew"ys\}\% |

```
19.46 \def\glossaryname{Verklarende Woordenl"yst}%
19.47 }
19.48 }\x
```

\datedutch The macro \datedutch redefines the command \today to produce Dutch dates.

```
19.49 \def\datedutch{%
9.50\\def\today{\number\day~\ifcase\month\or
19.51 januari\or februari\or maart\or april\or mei\or juni\or
19.52 juli\or augustus\or september\or oktober\or november\or
19.53 december\fi
19.54 \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 paterns.

```
9.55 \else
    \ifx\l@afrikaans\undefined
    \@nopatterns{Afrikaans}
    ifx\l@dutch\undefined
                \adddialect\l@afrikaans0
        \else
                            \adddialect\l@afrikaans\l@dutch
        \i
    \i
```

\captionsafrikaans Now is the time to define the words for 'Afrikaans'.

| 19.64 | \def \captionsafrikaans\{\% |
| :---: | :---: |
| 19.65 | \def \prefacename\{Voorwoord\}\% |
| 19.66 | $\backslash \mathrm{def} \backslash$ refname\{Verwysings\}\% |
| 19.67 | $\backslash d e f \backslash a b s t r a c t n a m e\{S a m e v a t t i n g\} \%$ |
| 19.68 | \def \bibname\{Bibliografie\}\% |
| 19.69 | \def \chaptername\{Hoofstuk\}\% |
| 19.70 | $\backslash \mathrm{def}$ \appendixname\{Bylae\}\% |
| 19.71 | \def \contentsname\{Inhoudsopgawe\}\% |
| 19.72 | \def\listfigurename\{Lys van figure\}\% |
| 19.73 | $\backslash d e f \backslash l i s t t a b l e n a m e\{L y s ~ v a n ~ t a b e l l e\} \% ~$ |
| 19.74 | \def \indexname\{Inhoud\}\% |
| 19.75 | \def \figurename\{Figuur\}\% |
| 19.76 | \def \tablename\{Tabel\}\% |
| 19.77 | $\backslash d e f \backslash p a r t n a m e\{D e e l\} \%$ |
| 19.78 | $\backslash \mathrm{def}$ \enclname\{Bylae(n) \}\% |
| 19.79 | $\backslash d e f \backslash c c n a m e\{a . a\} \$.  \hline 19.80 & $\backslash$ def \headtoname\{Aan\}\% |
| 19.81 | $\backslash d e f \backslash p a g e n a m e\{B l a d s y\} \%$ |
| 19.82 | \def \seename\{sien\}\% |
| 19.83 | $\backslash d e f \backslash a l s o n a m e\{s i e n ~ o o k\} \%$ |
| 19.84 | $\backslash \mathrm{def}$ \proofname\{Bewys\}\% |
| 19.85 | \} |

\dateafrikaans Here is the 'Afrikaans' version of the date macro.

| 19.86 | \def \dateafrikaans\{\% |
| :---: | :---: |
| 19.87 | $\backslash$ def \today $\backslash \backslash$ number $\backslash$ day $\sim$ \if case $\backslash$ month ${ }^{\text {a }}$ or |
| 19.88 | Januarie\or Februarie\or Maart\or April\or Mei\or Junie\or |
| 19.89 | Julie\or Augustus \or September\or Oktober\or November\or |
| 19.90 | Desember $\backslash$ fi |
| 19.91 | $\backslash$ space \number\year\}\} |

\extrasdutch The macros \extrasdutch and \captionsafrikaans will perform all the ex\extrasafrikaans
\noextrasdutch
\noextrasafrikaans tra definitions needed for the Dutch language. The macros \noextrasdutch and noextrasafrikaans is used to cancel the actions of \extrasdutch and \captionsafrikaans.

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.

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

Both version of the language use the same set of shorthand definitions althoug the ' ij ' is not used in Afrikaans.

```
19.94\@namedef{extras\CurrentOption}{\languageshorthands{dutch}}
19.95 \expandafter\addto\csname extras\CurrentOption\endcsname{%
19.96 \bbl@activate{"}}
```

The 'umlaut' character should be positioned lower on all vowels in Dutch texts.
19.97 \expandafter $\backslash$ addto\csname extras $\backslash$ CurrentOption \endcsname\{\%
19.98 \umlautlow\umlautelow\}
19.99 \@namedef\{noextras \CurrentOption\}\{\%
19.100 \umlauthigh\}
> \dutchhyphenmins The dutch hyphenation patterns can be used with \lefthyphenmin set to 2 and \afrikaanshyphenmins \righthyphenmin set to 3 .
> 19.101 \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

```
19.102\def\@trema#1{\allowhyphens\discretionary{-}{#1}{\"{#1}}\allowhyphens}
```

Now we can define the doublequote macros: the tremas,

```
19.104\declare@shorthand{dutch}{"e}{\textormath{\@trema e}{\ddot e}}
19.105 \declare@shorthand{dutch}{"i}{\textormath
19.106 {\allowhyphens\discretionary{-}{i}{\"{\i}}\allowhyphens}%
19.107 {\ddot \imath}}
19.108\declare@shorthand{dutch}{"o}{\textormath{\@trema o}{\ddot o}}
19.109 \declare@shorthand{dutch}{"u}{\textormath{\@trema u}{\ddot u}}
    dutch quotes
19.110\declare@shorthand{dutch}{"'}{%
19.111 \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
19.112 \declare@shorthand{dutch}{"'}{%
19.113 \textormath{\textquotedblright}{\mbox{\textquotedblright}}}
    and some additional commands:
```

```
19.114 \declare@shorthand{dutch}{"-}{\nobreak-\bbl@allowhyphens}
```

19.114 \declare@shorthand{dutch}{"-}{\nobreak-\bbl@allowhyphens}
19.115\declare@shorthand{dutch}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
19.115\declare@shorthand{dutch}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
19.116 \declare@shorthand{dutch}{"|}{%
19.116 \declare@shorthand{dutch}{"|}{%
19.117 \textormath{\discretionary{-}{}{\kern.03em}}{}}
19.117 \textormath{\discretionary{-}{}{\kern.03em}}{}}
19.118 \declare@shorthand{dutch}{""}{\hskip\z@skip}
19.118 \declare@shorthand{dutch}{""}{\hskip\z@skip}
19.119 \declare@shorthand{dutch}{"y}{\textormath{\ij{}}{\ddot y}}
19.119 \declare@shorthand{dutch}{"y}{\textormath{\ij{}}{\ddot y}}
19.120 \declare@shorthand{dutch}{"Y}{\textormath{\IJ{}}{\ddot Y}}

```
19.120 \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 "/.
19.121 \declare@shorthand\{dutch\}\{"/\}\{\textormath
19.122 \{\bbl@allowhyphens $\backslash$ discretionary\{/\}\{\}\{/\}\bbl@allowhyphens\}\{\}\}
\- All that is left now is the redefinition of $\backslash$-. The new version of $\backslash$ - should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{EX}}$ can generate from the hyphenation patterns.

```
19.123\expandafter\addto\csname extras\CurrentOption\endcsname{%
19.124 \babel@save\-}
19.125 \expandafter\addto\csname extras\CurrentOption\endcsname{%
19.126 \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.

```
19.127\ldf@finish\CurrentOption
19.128 </code\
```


## 20 The English language

The file english. $\mathrm{dtx}^{14}$ 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.
20.1 〈*code〉
20.2 \LdfInit\CurrentOption\{date\CurrentOption\}

When this file is read as an option, i.e. by the command,englishcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@englishtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

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.

```
\ifx\l@english\@undefined
\ifx\l@UKenglish\@undefined
\ifx\l@british\@undefined
\ifx\l@american\@undefined
                    \ifx\l@USenglish\@undefined
                    \ifx\l@canadian\@undefined
                    \ifx\l@australian\@undefined
                    \ifx\l@newzealand\@undefined
                            \@nopatterns{English}
                            \adddialect\l@english0
                    \else
                        \let\l@english\l@newzealand
                    \fi
                    \else
                        \let\l@english\l@australian
                    \i
                    \else
                    \let\l@english\l@canadian
                    \fi
                    \else
                    \let\l@english\l@USenglish
                \fi
            \else
                \let\l@english\l@american
            \fi
            \else
                        \let\l@english\l@british
```

[^13]```
20.30 \fi
20.31 \else
    \else
    \fi
0.34 \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 \1@english was defined. Now we want to make sure that \1@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.

```
20.35\ifx\l@british\@undefined
20.36 \ifx\l@UKenglish\@undefined
20.37 \adddialect\l@british\l@english
0.38 \adddialect\l@UKenglish\l@english
0.39 \else
            \let\l@british\l@UKenglish
    \i
    \else
    \let\l@UKenglish\l@british
20.44 \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'.

```
20.45\ifx\l@american\@undefined
20.46 \ifx\l@USenglish\@undefined
```

When the patterns are not know as 'american' or 'USenglish' we add a "dialect".

```
20.47 \adddialect\1@american\l@english
20.48 \else
20.49 \let\1@american\l@USenglish
20.50 \fi
20.51 \else
```

Make sure that USenglish is known, even if the patterns were loaded as 'american'.

```
20.52 \ifx\l@USenglish\@undefined
20.53 \let\l@USenglish\l@american
20.54 \fi
20.55 \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.

```
20.56\ifx\l@canadian\@undefined
20.57 \adddialect\l@canadian\l@american
```

20.58 \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.

```
20.59 \ifx\l@australian\@undefined
20.60 \adddialect\l@australian\l@british
20.61 \fi
20.62\ifx\l@newzealand\@undefined
20.63 \adddialect\l@newzealand\l@british
20.64 \fi
```

\englishhyphenmins This macro is used to store the correct values of the hyphenation parameters \lefthyphenmin and \righthyphenmin. 20.65 \providehyphenmins\{\CurrentOption\}\{\tw@\thr@@\}

The next step consists of defining commands to switch to (and from) the English language.
\captionsenglish The macro \captionsenglish defines all strings used in the four standard document classes provided with $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$.
20.66 \@namedef\{captions \CurrentOption\}\{\%
20.67 \def\prefacename\{Preface\}\%
20.68 \def $\backslash r e f n a m e\{R e f e r e n c e s\} \%$
20.69 \def $\backslash$ abstractname\{Abstract\}\%
20.70 \def \bibname\{Bibliography\}\%
20.71 \def\chaptername\{Chapter\}\%
20.72 \def\appendixname\{Appendix\}\%
20.73 \def \contentsname\{Contents\}\%
20.74 \def \listfigurename\{List of Figures\}\%
20.75 \def $\backslash$ listtablename\{List of Tables\}\%
20.76 \def \indexname\{Index\}\%
20.77 \def\figurename\{Figure\}\%
20.78 \def\tablename\{Table\}\%
20.79 \def\partname\{Part\}\%
20.80 \def \enclname\{encl\}\%
20.81 \def\ccname\{cc\}\%
20.82 \def $\backslash$ headtoname\{To\}\%
20.83 \def \pagename\{Page\}\%
20.84 \def $\backslash$ seename\{see\}\%
20.85 \def $\backslash$ alsoname\{see also\}\%
20.86 \def $\backslash$ proofname\{Proof $\} \%$
20.87 \def \glossaryname\{Glossary\}\%
$20.88\}$
\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.

```
20.89 \def\bbl@tempa{british}
20.90 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{UK}\fi
20.91 \def\bbl@tempa{UKenglish}
20.92\ifx\CurrentOption\bbl@tempa\def\bbl@tempb{UK}\fi
20.93\def\bbl@tempa{american}
20.94\ifx\CurrentOption\bbl@tempa\def\bbl@tempb{US}\fi
```

```
20.95\def\bbl@tempa{USenglish}
20.96\ifx\CurrentOption\bbl@tempa\def\bbl@tempb{US}\fi
20.97 \def\bbl@tempa{canadian}
20.98\ifx\CurrentOption\bbl@tempa\def\bbl@tempb{US}\fi
20.99\def\bbl@tempa{australian}
20.100 \ifx\CurrentOption\bbl@tempa\def\bbl@tempb{AU}\fi
20.101 \def\bbl@tempa{newzealand}
20.102\ifx\CurrentOption\bbl@tempa\def\bbl@tempb{AU}\fi
```

The macro \dateenglish redefines the command \today to produce English dates.
$20.103 \backslash$ def $\backslash$ bbl@tempa\{UK\}
20.104 \ifx $\backslash$ bbl@tempa $\backslash$ bbl@tempb
20.105 \@namedef\{date $\backslash$ CurrentOption\}\{\%
$20.106 \backslash$ def $\backslash t o d a y\{\backslash i f c a s e \backslash d a y \backslash o r$
20.107 1st \or 2nd\or 3rd\or 4th\or 5th\or
20.108 6th\or 7th \or 8th\or 9th\or 10th \or
$20.109 \quad 11$ th $\backslash$ or 12 th $\backslash o r ~ 13 t h \backslash o r ~ 14 t h \backslash o r ~ 15 t h \backslash o r ~$
$20.110 \quad 16 t h \backslash o r ~ 17 t h \backslash o r ~ 18 t h \backslash o r ~ 19 t h \backslash o r ~ 20 t h \backslash o r ~$
20.111 21st $\backslash$ or 22nd \or 23rd\or 24th
$20.112 \quad 26$ th $\operatorname{lor} 27$ th $\backslash$ or $28 t h \backslash o r ~ 29 t h \backslash o r ~ 30 t h \backslash o r ~$
20.113 31st $\backslash$ fi~ ${ }^{\sim}$ ifcase\month $\backslash$ or
20.114 January \or February\or March\or April\or May\or June\or
20.115 July \or August\or September\or October\or November\or
20.116 December\fi\space \number\year\}\}
\dateaustralian Now, test for 'australian' or 'american'.
20.117 \else

The macro \dateaustralian redefines the command \today to produce Australian resp. New Zealand dates.

```
20.118 \def\bbl@tempa{AU}
20.119 \ifx\bbl@tempa\bbl@tempb
20.120 \@namedef{date\CurrentOption}{%
20.121 \def\today{\number\day~\ifcase\month\or
20.122 January\or February\or March\or April\or May\or June\or
20.123 July\or August\or September\or October\or November\or
20.124 December\fi\space \number\year}}
```

\dateamerican The macro \dateamerican redefines the command \today to produce American dates.

```
20.125 \else
20.126 \@namedef{date\CurrentOption}{%
20.127 \def\today{\ifcase\month\or
20.128 January\or February\or March\or April\or May\or June\or
20.129 July\or August\or September\or October\or November\or
20.130 December\fi \space\number\day, \number\year}}
20.131 \fi
20.132\fi
```

\extrasenglish The macro \extrasenglish will perform all the extra definitions needed for the \noextrasenglish 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.

```
20.133\@namedef{extras\CurrentOption}{}
20.134\@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.
20.135 \ldf@finish \CurrentOption
20.136 〈/code〉

## 21 The German language

The file germanb. $\mathrm{dtx}^{15}$ defines all the language definition macros for the German language as well as for the Austrian dialect of this language ${ }^{16}$.

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 ß (like \ss{}).
"z to produce the German B (like \ss{}).
"ck for ck to be hyphenated as k-k.
"ff for ff to be hyphenated as ff-f, this is also imple-
    mented 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 compund
    words with hyphen, e.g. x-""y).
"~ for a compound word mark without a breakpoint.
"= for a compound word mark with a breakpoint, allow-
    ing 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 .
When this file was read through the option germanb we make it behave as if german was specified.
21.1 \def $\backslash$ bbl@tempa\{germanb\}
21.2 \ifx $\backslash$ CurrentOption $\backslash$ bbl@tempa
21.3 \def\CurrentOption\{german\}
21.4 \fi

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

[^14]Table 5: More commands which produce quotes, defined by german.ldf

```
21.5 <*code>
21.6\LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e., by the command,germanwillbean'unknown'language,sowehavetomakeitknown.Sowecheckfortheexistenceof\1@germantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
21.7\ifx\l@german\@undefined
21.8 \@nopatterns{German}
21.9 \adddialect\l@german0
21.10\fi
```

For the Austrian version of these definitions we just add another language. 21.11 \adddialect\1@austrian\1@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 \captionsaustrian all strings used in the four standard document classes provided with $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$.

```
21.12\@namedef{captions\CurrentOption}{%
21.13\def\prefacename{Vorwort}%
21.14 \def\refname{Literatur}%
21.15\def\abstractname{Zusammenfassung}%
21.16 \def\bibname{Literaturverzeichnis}%
21.17 \def\chaptername{Kapitel}%
21.18\\def\appendixname{Anhang}%
21.19 \def\contentsname{Inhaltsverzeichnis}% % oder nur: Inhalt
21.20\\def\listfigurename{Abbildungsverzeichnis}%
21.21 \def\listtablename{Tabellenverzeichnis}%
21.22\def\indexname{Index}%
21.23\def\figurename{Abbildung}%
21.24 \def\tablename{Tabelle}% % oder: Tafel
21.25 \def\partname{Teil}%
21.26 \def\enclname{Anlage(n)}% % oder: Beilage(n)
21.27 \def\ccname{Verteiler}% % oder: Kopien an
21.28\def\headtoname{An}%
```

```
21.29\\def\pagename{Seite}%
21.30\\def\seename{siehe}%
21.31 \def\alsoname{siehe auch}%
21.32\def\proofname{Beweis}%
21.33\def\glossaryname{Glossar}%
21.34}
```

\dategerman The macro \dategerman redefines the command \today to produce German dates.

```
21.35\def\month@german{\ifcase\month\or
21.36 Januar\or Februar\or M\"arz\or April\or Mai\or Juni\or
21.37 Juli\or August\or September\or Oktober\or November\or Dezember\fi}
21.38\def\dategerman{\def\today{\number\day. ~\month@german
21.39 \space\number\year}}
```

\dateaustrian The macro \dateaustrian redefines the command \today to produce Austrian version of the German dates.
$21.40 \backslash$ def $\backslash$ dateaustrian\{ $\backslash$ def $\backslash$ today $\{\backslash$ number $\backslash$ day. $\sim \backslash i f n u m 1=\backslash m o n t h ~$
$21.41 \mathrm{~J} \backslash$ "anner\else \month@german $\backslash$ fi $\backslash$ space $\backslash$ number $\backslash y e a r\}\}$
\extrasgerman Either the macro \extrasgerman or the macros \extrasaustrian will per-
\extrasaustrian
\noextrasgerman \noextrasaustrian form all the extra definitions needed for the German language. The macro \noextrasgerman is used to cancel the actions of \extrasgerman.

For German (as well as for Dutch) the " character is made active. This is done once, later on its definition may vary.

```
21.42 \initiate@active@char{"}
21.43\@namedef{extras\CurrentOption}{%
21.44 \languageshorthands{german}}
21.45 \expandafter\addto\csname extras\CurrentOption\endcsname{%
21.46 \bbl@activate{"}}
```

Don't forget to turn the shorthands off again.
21.47 \addto\noextrasgerman\{\bbl@deactivate\{"\}\}

In order for $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ to be able to hyphenate German words which contain ' $\beta$ ' (in the OT1 position ${ }^{\wedge}$ - Y ) we have to give the character a nonzero \lccode (see Appendix H, the $\left.\mathrm{T}_{\mathrm{E}} \mathrm{Xbook}\right)$.
21.48 \expandafter\addto\csname extras\CurrentOption\endcsname\{\%
21.49 \babel@savevariable\{\lccode25\}\%
21.50 \lccode25=25\}

The umlaut accent macro $\backslash "$ is changed to lower the umlaut dots. The redefinition is done with the help of \umlautlow.
21.51 \expandafter $\backslash$ addto\csname extras \CurrentOption\endcsname\{\%
21.52 \babel@save\"\umlautlow\}
21.53 \@namedef\{noextras \CurrentOption\}\{\umlauthigh\}

The german hyphenation patterns can be used with \lefthyphenmin and \righthyphenmin set to 2.
$21.54 \backslash$ providehyphenmins\{\CurrentOption\}\{\tw@\tw@\}

For German texts we need to make sure that $\backslash$ frenchspacing is turned on.

21.56 \bbl@frenchspacing\}
21.57 \expandafter\addto\csname noextras \CurrentOption\endcsname\{\%
21.58 \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 ".

```
21.59 \begingroup \catcode'\"12
21.60\def\x{\endgroup
21.61 \def\@SS{\mathchar"7019 }
21.62 \def\dq{"}}
21.63\x
```

Now we can define the doublequote macros: the umlauts,
$21.64 \backslash$ declare@shorthand\{german\}\{"a\}\{\textormath\{\"\{a\}\allowhyphens\}\{\ddot a\}\}
$21.65 \backslash$ declare@shorthand\{german\}\{"o\}\{\textormath\{\"\{o\}\allowhyphens\}\{\ddot o\}\}
$21.66 \backslash$ declare@shorthand\{german\}\{"u\}\{\textormath\{\"\{u\}\allowhyphens\}\{\ddot u\}\}
21.67 \declare@shorthand\{german\}\{"A\}\{\textormath\{\"\{A\}\allowhyphens\}\{\ddot A\}\}
21.68 \declare@shorthand\{german\}\{"0\}\{\textormath\{\"\{0\}\allowhyphens\}\{\ddot 0\}\}
21.69 \declare@shorthand\{german\}\{"U\}\{\textormath\{\"\{U\}\allowhyphens\}\{\ddot U\}\}
tremas,
$21.70 \backslash$ declare@shorthand\{german\}\{"e\}\{\textormath\{\"\{e\}\}\{\ddot e\}\}
21.71 \declare@shorthand\{german\}\{"E\}\{\textormath\{\"\{E\}\}\{\ddot E\}\}
21.72 \declare@shorthand\{german\}\{"i\}\{\textormath\{\"\{\i\}\}\%
21.73 \{\ddot\imath\}\}
$21.74 \backslash$ declare@shorthand\{german\}\{"I\}\{\textormath\{\"\{I\}\}\{\ddot I\}\}
german es-zet (sharp s),
21.75 \declare@shorthand\{german\}\{"s\}\{\textormath\{\ss\}\{\@SS\{\}\}\}
21.76 \declare@shorthand\{german\}\{"S\}\{\SS\}
21.77 \declare@shorthand\{german\}\{"z\}\{\textormath\{\ss\}\{\@SS\{\}\}\}
21.78 \declare@shorthand\{german\}\{"Z\}\{SZ\}
german and french quotes,
21.79 \declare@shorthand\{german\}\{"'\}\{\glqq\}
21.80 \declare@shorthand\{german\}\{"'\}\{\grqq\}
21.81 \declare@shorthand\{german\}\{"<\}\{\flqq\}
21.82 \declare@shorthand\{german\}\{">\}\{\frqq\}
discretionary commands

```
21.83\declare@shorthand{german}{"c}{\textormath{\bbl@disc ck}{c}}
21.84 \declare@shorthand{german}{"C}{\textormath{\bbl@disc CK}{C}}
21.85\declare@shorthand{german}{"F}{\textormath{\bbl@disc F{FF}}{F}}
21.86 \declare@shorthand{german}{"l}{\textormath{\bbl@disc l{ll}}{l}}
```

21.87 \declare@shorthand\{german\}\{"L\}\{\textormath\{\bbl@disc L\{LL\}\}\{L\}\}
21.88 \declare@shorthand\{german\}\{"m\}\{\textormath\{\bbl@disc m\{mm\}\}\{m\}\}
21.89 \declare@shorthand\{german\}\{"M\}\{\textormath\{\bbl@disc M\{MM\}\}\{M\}\}
$21.90 \backslash$ declare@shorthand\{german\}\{"n\}\{\textormath\{\bbl@disc n\{nn\}\}\{n\}\}
21.91 \declare@shorthand\{german\}\{"N\}\{\textormath\{\bbl@disc N\{NN\}\}\{N\}\}
21.92 \declare@shorthand\{german\}\{"p\}\{\textormath\{\bbl@disc p\{pp\}\}\{p\}\}
$21.93 \backslash$ declare@shorthand\{german\}\{"P\}\{\textormath\{\bbl@disc P\{PP\}\}\{P\}\}
$21.94 \backslash$ declare@shorthand\{german\}\{"r\}\{\textormath\{\bbl@disc r\{rr\}\}\{r\}\}
21.95 \declare@shorthand\{german\}\{"R\}\{\textormath\{\bbl@disc R\{RR\}\}\{R\}\}
$21.96 \backslash$ declare@shorthand\{german\}\{"t\}\{\textormath\{\bbl@disc $t\{t t\}\}\{t\}\}$
21.97 \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.
21.98 \declare@shorthand\{german\}\{"f\}\{\textormath\{\bbl@discff\}\{f\}\}
21.99 \def $\backslash b b l @ d i s c f f\{\backslash p e n a l t y \backslash @ M$
21.100 \afterassignment \bbl@insertff \let\bbl@nextff= \}
21.101 \def $\backslash \mathrm{bbl@insertff}\{\%$
21.102 \if $\mathrm{f} \backslash \mathrm{bbl@nextff}$
$21.103 \backslash e x p a n d a f t e r \backslash @ f i r s t o f t w o \backslash e l s e \backslash e x p a n d a f t e r \backslash @ s e c o n d o f t w o \backslash f i$ 21.104 \{\relax\discretionary\{ff-\}\{f\}\{ff\}\allowhyphens\}\{f $\backslash$ bbl@nextff $\}\}$ 21.105 \let $\backslash \mathrm{bbl@nextff=f}$
and some additional commands:
21.106 \declare@shorthand\{german\}\{"-\}\{\nobreak\-\bbl@allowhyphens\}
21.107 \declare@shorthand\{german\}\{"|\}\{\%
21.108 \textormath\{\penalty $\backslash @ \backslash$ discretionary\{-\}\{\}\{\kern.03em\}\%
21.109 \allowhyphens\}\{\}\}
21.110 \declare@shorthand\{german\}\{""\}\{\hskip\z@skip\}
21.111 \declare@shorthand\{german\}\{"~\}\{\textormath\{\leavevmode\hbox\{-\}\}\{-\}\}
21.112 \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 compat$\backslash$ mdqoff ibility with german.sty.
\ck ${ }_{1.113}$ \def $\backslash$ mdqon $\{\backslash$ shorthandon\{" $\left.\}\right\}$
$21.114 \backslash \operatorname{def} \backslash \operatorname{mdqoff}\{\backslash$ shorthandoff\{"\}\}
21.115 \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.
21.116 \ldf@finish $\backslash$ CurrentOption 21.117 〈/code〉

## 22 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.

```
22.1\def\bbl@tempa{ngermanb}
22.2\ifx\CurrentOption\bbl@tempa
22.3\def\CurrentOption{ngerman}
22.4\fi
```

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

```
22.5 <*code〉
22.6\LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e., by the command,ngermanwillbean'unknown'language,sowehavetomakeitknown.Sowecheckfortheexistenceof\1@ngermantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
22.7\ifx\l@ngerman\@undefined
22.8 \@nopatterns{ngerman}
22.9 \adddialect\l@ngerman0
22.10\fi
```

For the Austrian version of these definitions we just add another language. 22.11 \adddialect $\backslash 1 @ n a u s t r i a n \backslash l @ n g e r m a n$

The next step consists of defining commands to switch to (and from) the German language.
\captionsngerman Either the macro \captionnsgerman or the macro \captionsnaustrian will de\captionsnaustrian fine all strings used in the four standard document classes provided with $\mathrm{HAT}_{\mathrm{E}} \mathrm{X}$.

```
@namedef{captions\CurrentOption}{%
\def\prefacename{Vorwort}%
\def\refname{Literatur}%
\def\abstractname{Zusammenfassung}%
\def\bibname{Literaturverzeichnis}%
\def\chaptername{Kapitel}%
\def\appendixname{Anhang}%
\def\contentsname{Inhaltsverzeichnis}% % oder nur: Inhalt
\def\listfigurename{Abbildungsverzeichnis}%
\def\listtablename{Tabellenverzeichnis}%
\({ }^{17}\) The file described in this section has version number v 2.6 m and was last revised on 2004/02/20.
```

```
22.22 \def\indexname{Index}%
22.23\def\figurename{Abbildung}%
22.24 \def\tablename{Tabelle}% % oder: Tafel
22.25\def\partname{Teil}%
22.26 \def\enclname{Anlage(n)}% % oder: Beilage(n)
22.27 \def\ccname{Verteiler}% % oder: Kopien an
22.28\def\headtoname{An}%
22.29\def\pagename{Seite}%
22.30\def\seename{siehe}%
22.31\def\alsoname{siehe auch}%
22.32\def\proofname{Beweis}%
22.33\def\glossaryname{Glossar}%
22.34 }
```

\datengerman The macro \datengerman redefines the command \today to produce German dates.

```
22.35\def\month@ngerman{\ifcase\month\or
22.36 Januar\or Februar\or M\"arz\or April\or Mai\or Juni\or
22.37 Juli\or August\or September\or Oktober\or November\or Dezember\fi}
22.38\def \datengerman{\\def \today{\number\day. ~\month@ngerman
22.39 \space\number\year}}
```

\dateanustrian The macro \datenaustrian redefines the command \today to produce Austrian version of the German dates.

```
22.40\def\datenaustrian{\def\today{\number\day. ~\ifnum1=\month
22.41 J\"anner\else \month@ngerman\fi \space\number\year}}
```

\extrasngerman Either the macro \extrasngerman or the macros \extrasnaustrian will per\extrasnaustrian \noextrasngerman form all the extra definitions needed for the German language. The macro \noextrasngerman is used to cancel the actions of \extrasngerman.

For German (as well as for Dutch) the " character is made active. This is done once, later on its definition may vary.

```
22.42 \initiate@active@char{"}
22.43\@namedef{extras\CurrentOption}{%
22.44 \languageshorthands{ngerman}}
22.45 \expandafter\addto\csname extras\CurrentOption\endcsname{%
22.46 \bbl@activate{"}}
```

    Don't forget to turn the shorthands off again.
    22.47 \addto $\backslash$ noextrasngerman\{ $\backslash$ bbl@deactivate\{" $\}\}$

In order for $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ to be able to hyphenate German words which contain ' $\beta$ ' (in the OT1 position ${ }^{\sim}$ Y) we have to give the character a nonzero \lccode (see Appendix H , the $\mathrm{T}_{\mathrm{E}} \mathrm{Xb}$ book).
22.48 \expandafter\addto\csname extras \CurrentOption\endcsname\{\%
22.49 \babel@savevariable\{\lccode25\}\%
22.50 \lccode25=25\}

The umlaut accent macro $\backslash "$ is changed to lower the umlaut dots. The redefinition is done with the help of \umlautlow.

```
22.51 \expandafter\addto\csname extras\CurrentOption\endcsname{%
22.52 \babel@save\"\umlautlow}
22.53\@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 .

```
22.54 \providehyphenmins{\CurrentOption}{\tw@\tw@}
```

For German texts we need to make sure that \frenchspacing is turned on.
22.55 \expandafter\addto\csname extras\CurrentOption\endcsname\{\%
22.56 \bbl@frenchspacing\}
22.57 \expandafter\addto\csname noextras\CurrentOption\endcsname\{\%
22.58 \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 $\backslash "$ can now be typed as ".

```
22.59 \begingroup \catcode'\"12
22.60\def\x{\endgroup
22.61\def\@SS{\mathchar"7019 }
22.62 \def\dq{"}}
22.63\x
```

Now we can define the doublequote macros: the umlauts,

```
22.64 \declare@shorthand{ngerman}{"a}{\textormath{\"{a}\allowhyphens}{\ddot a}}
```

22.65 \declare@shorthand\{ngerman\}\{"o\}\{\textormath\{\"\{o\}\allowhyphens\}\{\ddot o\}\}
22.66 \declare@shorthand\{ngerman\}\{"u\}\{\textormath\{\"\{u\}\allowhyphens\}\{\ddot u\}\}
22.67 \declare@shorthand\{ngerman\}\{"A\}\{\textormath\{\"\{A\}\allowhyphens\}\{\ddot A\}\}
22.68 declare@shorthand\{ngerman\}\{"O\}\{\textormath $\{\backslash "\{0\} \backslash$ allowhyphens $\}\{\backslash$ ddot 0$\}\}$
22.69 \declare@shorthand\{ngerman\}\{"U\}\{\textormath\{\"\{U\}\allowhyphens\}\{\ddot U\}\}
tremas,
22.70 \declare@shorthand\{ngerman\}\{"e\}\{\textormath\{\"\{e\}\}\{\ddot e\}\}
22.71 \declare@shorthand\{ngerman\}\{"E\}\{\textormath\{\"\{E\}\}\{\ddot E\}\}
22.72 \declare@shorthand\{ngerman\}\{"i\}\{\textormath\{\"\{\i\}\}\%
22.73 \{\ddot\imath\}\}
$22.74 \backslash$ declare@shorthand\{ngerman\}\{"I\}\{\textormath $\{\backslash$ "\{I\}\}\{\ddot I\}\}
german es-zet (sharp s),
22.75 \declare@shorthand\{ngerman\}\{"s\}\{\textormath\{\ss\}\{\@SS\{\}\}\}
22.76 \declare@shorthand\{ngerman\}\{"S\}\{\SS\}
$22.77 \backslash$ declare@shorthand\{ngerman\}\{"z\}\{\textormath\{\ss\}\{\@SS\{\}\}\}
22.78 \declare@shorthand\{ngerman\}\{"Z\}\{SZ\}
german and french quotes,
22.79 \declare@shorthand\{ngerman\}\{"'\}\{\glqq\}
22.80 \declare@shorthand\{ngerman\}\{"'\}\{\grqq\}

```
22.81\declare@shorthand{ngerman}{"<}{\flqq}
22.82 \declare@shorthand{ngerman}{">}{\\rqq}
and some additional commands:
22.83\declare@shorthand{ngerman}{"-}{\nobreak\-\bbl@allowhyphens}
22.84 \declare@shorthand{ngerman}{"|}{%
22.85 \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
22.86 \allowhyphens}{}}
22.87\declare@shorthand{ngerman}{""}{\hskip\z@skip}
22.88 \declare@shorthand{ngerman}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
22.89\declare@shorthand{ngerman}{"=}{\penalty\@M-\hskip\z@skip}
```

$\backslash$ mdqon All that's left to do now is to define a couple of commands for reasons of compat\mdqoff ibility with german.sty.
22.90 \def $\backslash$ mdqon $\{\backslash$ shorthandon\{" $\}\}$
22.91 \def $\backslash m d q o f f\{\backslash$ 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.

```
22.92\ldf@finish\CurrentOption
```

22.93 〈/code〉

## 23 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.
23.1 〈*code〉
23.2 \LdfInit\{breton\}\captionsbreton

When this file is read as an option, i.e. by the command,bretonwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof$\backslash1@breton$toseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
\ifx\l@breton\@undefined
    \@nopatterns{Breton}
    \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 $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
\addto\captionsbreton{%
    \def\prefacename{Rakskrid}%
    \def\refname{Daveenno\'u}%
    \def\abstractname{Dvierra\~n}%
    \def\bibname{Lennadurezh}%
    \def\chaptername{Pennad}%
    \def\appendixname{Stagadenn}%
    \def\contentsname{Taolenn}%
    \def\listfigurename{Listenn ar Figurenno\'u}%
    \def\listtablename{Listenn an taolenno\'u}%
    \def\indexname{Meneger}%
    \def\figurename{Figurenn}%
    \def\tablename{Taolenn}%
    \def\partname{Lodenn}%
    \def\enclname{Diello\'u kevret}%
    \def\ccname{Eilskrid da}%
    \def\headtoname{evit}
    \def\pagename{Pajenn}%
    \def\seename{Gwelout}%
```

    \({ }^{18}\) The file described in this section has version number \(v 1.0 \mathrm{~h}\) and was last revised on
    2005/03/29.
    ```
23.25\def\alsoname{Gwelout ivez}%
23.26 \def\proofname{Proof}% <-- needs translation
23.27 \def\glossaryname{Glossary}% <-- Needs translation
23.28}
```

\datebreton The macro \datebreton redefines the command \today to produce Breton dates.

```
23.29\def\datebreton{%
23.30\\def\today{\ifnum\day=1\relax 1\/$~{\rm a\tilde{n}}$\else
23.31 \number\day\fi \space a\space viz\space\ifcase\month\or
23.32 Genver\or C'hwevrer\or Meurzh\or Ebrel\or Mae\or Mezheven\or
23.33 Gouere\or Eost\or Gwengolo\or Here\or Du\or Kerzu\fi
23.34 \space\number\year}}
```

\extrasbreton The macro \extrasbreton will perform all the extra definitions needed for the \noextrasbreton 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.

```
23.35 \initiate@active@char{:}
```

23.36 \initiate@active@char\{;\}
23.37 \initiate@active@char\{!\}
23.38 \initiate@active@char\{?\}

We specify that the breton group of shorthands should be used.
23.39 \addto\extrasbreton\{\languageshorthands\{breton\}\}

These characters are 'turned on' once, later their definition may vary.

```
23.40 \addto\extrasbreton{%
23.41 \bbl@activate{:}\bbl@activate{;}%
23.42 \bbl@activate{!}\bbl@activate{?}}
```

Don't forget to turn the shorthands off again.
23.43 \addto ${ }^{\text {noextrasbreton\{ }}$ \%
23.44 \bbl@deactivate\{:\}\bbl@deactivate\{; \}\%
23.45 \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.
23.46 \addto \extrasbreton\{\bbl@frenchspacing\}
23.47 \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.

```
23.48 \declare@shorthand{breton}{;}{%
23.49 \ifmmode
23.50 \string;\space
23.51 \else\relax
```

In horizontal mode we check for the presence of a 'space' and replace it by a \thinspace.

```
23.52 \ifhmode
23.53 \ifdim\lastskip>\z@
23.54
23.55 \fi
3.56 \fi
23.57 \string;\spac
23.58 \fi}%
```

\breton@sh@:@ Because these definitions are very similar only one is displayed in a way that the \breton@sh@!@ definition can be easily checked.

```
23.59 \declare@shorthand{breton}{:}{%
23.60 \ifmmode\string:\space
23.61 \else\relax
23.62 \ifhmode
23.63 \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
23.64 \fi
23.65 \string:\space
23.66 \fi}
23.67 \declare@shorthand{breton}{!}{%
23.68 \ifmmode\string!\space
23.69 \else\relax
23.70 \ifhmode
23.71 \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
23.72 \fi
23.73 \string!\space
23.74 \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.

```
23.75 \declare@shorthand{breton}{?}{%
23.76 \ifmmode
23.77 \string?\space
23.78 \else\relax
23.79 \ifhmode
23.80 \ifdim\lastskip>\z@
23.81 \unskip
23.82 \kern\fontdimen2\font
\k.83 \kern-1.4\fontdimen3\font
23.84 \fi
23.85 \fi
23.86 \string?\space
23.87 \fi}
```

All that is left to do now is provide the breton user with some extra utilities. Some definitions for special characters.

```
23.88 \DeclareTextSymbol{\at}{OT1}{64}
```

```
23.89 \DeclareTextSymbol{\at}{T1}{64}
23.90\DeclareTextSymbolDefault{\at}{0T1}
23.91 \DeclareTextSymbol{\boi}{OT1}{92}
23.92 \DeclareTextSymbol{\boi}{T1}{16}
23.93\DeclareTextSymbolDefault{\boi}{0T1}
23.94\DeclareTextSymbol{\circonflexe}{0T1}{94}
23.95\DeclareTextSymbol{\circonflexe}{T1}{2}
23.96 \DeclareTextSymbolDefault{\circonflexe}{0T1}
23.97 \DeclareTextSymbol{\tild}{0T1}{126}
23.98\DeclareTextSymbol{\tild}{T1}{3}
23.99\DeclareTextSymbolDefault{\tild}{0T1}
23.100 \DeclareTextSymbol{\degre}{0T1}{23}
23.101 \DeclareTextSymbol{\degre}{T1}{6}
23.102 \DeclareTextSymbolDefault{\degre}{0T1}
```

The following macros are used in the redefinition of $\backslash$ ^ and $\backslash "$ to handle the letter i.

```
23.103\AtBeginDocument{%
23.104 \DeclareTextCompositeCommand{\^}{OT1}{i}{\^\i}
23.105 \DeclareTextCompositeCommand{\"}{0T1}{i}{\"\i}}
```

And some more macros for numbering.

```
23.106 \def \kentan{1\/${}~{\rm a\tilde{n}}$}
```

23.107 \def $\backslash \mathrm{eil}\{2 \backslash / \$\{ \} \sim\{\backslash \mathrm{rm}$ l\}\$\}
$23.108 \backslash \operatorname{def} \backslash r e\{\backslash / \$\{ \} \sim\{\backslash \mathrm{rm}$ re\} $\$\}$
23.109 \def $\backslash$ trede\{3\re\}
$23.110 \backslash$ def $\backslash$ pevare $\{4 \backslash r e\}$
23.111 \def \vet\{\/\$\{\}~\{\rm vet\}\$\}
23.112 \def $\backslash$ pempvet $\{5 \backslash$ vet $\}$

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document \} and resetting the category code of @ to its original value.
23.113 \ldf@finish\{breton\}
23.114 〈/code〉

## 24 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.

```
24.1 〈*code\rangle
24.2 \LdfInit{welsh}{captionswelsh}
```

When this file is read as an option, i.e. by the command,welshcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof$\backslash1@wel$shtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
\ifx\undefined\l@welsh
    \@nopatterns{welsh}
24.5 \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.
24.6 \providehyphenmins\{\CurrentOption\}\{\tw@\thr@@\}
\captionswelsh The macro \captionswelsh defines all strings used in the four standard documentclasses provided with $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$.

```
\def\captionswelsh{%
    \def\prefacename{Rhagair}%
    \def\refname{Cyfeiriadau}%
    \def\abstractname{Crynodeb}%
    \def\bibname{Llyfryddiaeth}%
    \def\chaptername{Pennod}%
    \def\appendixname{Atodiad}%
    \def\contentsname{Cynnwys}%
    \def\listfigurename{Rhestr Ddarluniau}%
    \def\listtablename{Rhestr Dablau}%
    \def\indexname{Mynegai}%
    \def\figurename{Darlun}%
    \def\tablename{Taflen}%
    \def\partname{Rhan}%
    \def\enclname{amgae\"edig}%
    \def\ccname{cop\"\i au}%
    \def\headtoname{At}% % 'at' on letters meaning 'to ( a person)'
        % 'to (a place)' is 'i' in Welsh
    \def\pagename{tudalen}%
    \def\seename{gweler}%
    \def\alsoname{gweler hefyd}%
```

[^15]```
24.28\\def\proofname{Prawf}%
24.29 \def\glossaryname{Rhestr termau}%
24.30 }
```

\datewelsh The macro \datewelsh redefines the command \today to produce welsh dates.

```
24.31 \def \datewelsh{%
24.32\def\today{\ifnum\day=1\relax 1\/$^{\mathrm{a\tilde{n}}}$\else
24.33\number\day\fi\space\ifcase\month\or
24.34 Ionawr\or Chwefror\or Mawrth\or Ebrill\or
24.35 Mai\or Mehefin\or Gorffennaf\or Awst\or
24.36 Medi\or Hydref\or Tachwedd\or Rhagfyr\fi
24.37 \space\number\year}}
```

\extraswelsh The macro \extraswelsh will perform all the extra definitions needed for the $\backslash$ noextraswelsh 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.

```
24.38 \addto\extraswelsh{}
24.39 \addto\noextraswelsh{}
```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of @ to its original value.
24.40 \ldf@finish\{welsh\}
24.41 〈/code〉

## 25 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.
$25.1\langle *$ code $\rangle$
25.2 \LdfInit\{irish\}\captionsirish

When this file is read as an option, i.e. by the command,irishcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@irishtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
\ifx\l@irish\@undefined
    \@nopatterns{irish}
25.5 \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 $\backslash$ lefthyphenmin and \righthyphenmin.
$25.6 \backslash$ providehyphenmins\{\CurrentOption\}\{\tw@\thr@@\}
\captionsirish The macro \captionsirish defines all strings used in the four standard documentclasses provided with $\mathrm{HAT}_{\mathrm{E}} \mathrm{X}$.

```
7 \addto\captionsirish{%
    \def\prefacename{R\'eamhr\'a}% <-- also "Brollach"
    \def\refname{Tagairt\'{\i}}%
    \def\abstractname{Achoimre}%
    \def\bibname{Leabharliosta}%
    \def\chaptername{Caibidil}%
    \def\appendixname{Aguis\'{\i}n}%
    \def\contentsname{Cl\'ar \'Abhair}%
    \def\listfigurename{L\'ear\'aid\'{\i}}%%
    \def\listtablename{T\'abla\'{\i}}%
    \def\indexname{Inn\'eacs}%
    \def\figurename{L\'ear\'aid}%
    \def\tablename{T\'abla}%
    \def\partname{Cuid}%
    \def\enclname{faoi iamh}%
    \def\ccname{cc}% abrv. 'c\'oip chuig'
    \def\headtoname{Go}%
    \def\pagename{Leathanach}%
    \def\seename{f\'each}%
    \def\alsoname{f\'each freisin}%
    \def\proofname{Cruth\'unas}%
```

[^16]```
25.28\def\glossaryname{Glossary}% <-- Needs translation
25.29 }
```

\dateirish The macro \dateirish redefines the command \today to produce Irish dates.

| $25.30 \backslash$ def \dateirish\{\% |  |
| :---: | :---: |
| 25.31 | \def \today\{\% |
| 25.32 | $\backslash$ number \day ${ }^{\text {space }}$ \ifcase\month\or |
| 25.33 | Ean\'air\or Feabhra\or M\'arta\or Aibre\'an\or |
| 25.34 | Bealtaine\or Meitheamh\or I\'uil\or L\'unasa\or |
| 25.35 | Me \'an F \'omhair ${ }^{\text {dor }}$ Deireadh F\'omhair ${ }^{\text {dor }}$ |
| 25.36 | M \'\{\i\} na Samhna\or M\'\{\i\} na Nollag\fi |
| 25.37 | \space \number\year\}\} |

\extrasirish The macro \extrasirish will perform all the extra definitions needed for the \noextrasirish 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.

```
25.38 \addto\extrasirish{}
25.39 \addto\noextrasirish{}
```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of @ to its original value.
25.40 \ldf@finish\{irish\}
25.41 〈/code〉

## 26 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.

```
26.1 〈*code>
26.2\LdfInit{scottish}\captionsscottish
```

When this file is read as an option, i.e. by the command,scottishcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@scottishtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
.3\ifx\l@scottish\@undefined
26.4 \@nopatterns{scottish}
26.5 \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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
26.6 \addto\captionsscottish\{\%
26.7 \def\prefacename\{Preface\}\% <-- needs translation
26.8 \def \(\backslash\) refname\{Iomraidh\}\%
26.9 \def \abstractname\{Br\'\{\i\}gh\}\%
\(26.10 \backslash\) def \(\backslash\) bibname\{Leabhraichean\}\%
26.11 \def\chaptername\{Caibideil\}\%
26.12 \def \appendixname\{Ath-sgr'\{\i\}obhadh\}\%
26.13 \def \contentsname\{Cl\'ar-obrach\}\%
26.14 \def\listfigurename\{Liosta Dhealbh \}\%
26.15 \def\listtablename\{Liosta Chl\‘ar\}\%
26.16 \def\indexname\{Cl\'ar-innse\}\%
26.17 \def\figurename\{Dealbh\}\%
26.18 \def \tablename\{Cl\‘ar\}\%
26.19 \def\partname\{Cuid\}\%
26.20 \def\enclname\{a-staigh\}\%
26.21 \def \ccname\{lethbhreac gu\}\%
\(26.22 \backslash\) def \(\backslash\) headtoname\{gu\}\%
26.23 \def\pagename\{t.d.\}\% abrv. 'taobh duilleag'
26.24 \def\seename\{see\}\% <-- needs translation
26.25 \def \(\backslash\) alsoname\{see also\}\% <-- needs translation
26.26 \def\proofname\{Proof\}\% <-- needs translation
26.27 \def\glossaryname\{Glossary\}\% <-- Needs translation
\(26.28\}\)
```

[^17]\datescottish The macro \datescottish redefines the command \today to produce Scottish dates.

```
2.29\def\datescottish{%
26.30 \def\today{%
26.31 \number\day\space \ifcase\month\or
26.32 am Faoilteach\or an Gearran\or am M\'art\or an Giblean\or
26.33 an C\'eitean\or an t-\'Og mhios\or an t-Iuchar\or
26.34 L\'unasdal\or an Sultuine\or an D\'amhar\or
26.35 an t-Samhainn\or an Dubhlachd\fi
26.36 \space \number\year}}
```

\extrasscottish The macro \extrasscottish will perform all the extra definitions needed for the \noextrasscottish Scottish language. The macro \noextrasscottish is used to cancel the actions of \extrasscottish. For the moment these macros are empty but they are defined for compatibility with the other language definition files.
26.37 \addto\extrasscottish\{\}
26.38 \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.
26.39 \ldf@finish\{scottish\}
26.40 〈/code〉

## 27 The Greek language

The file greek.dtx ${ }^{22}$ defines all the language definition macros for the Greek language, i.e., as it used today with only one accent, and the attribute $\pi о \lambda v \tau о \nu \kappa o ́$ ("Polutoniko") 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
\latintext
\textgreek
\textlatin
\textol

The commands \greektext and \latintext can be used to switch to greek or latin fonts. These are declarations.

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 typests 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, $\backslash$ permill, and \vardigamma.

### 27.1 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 I. 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" додотодько́. In that case we only need the ' and " symbols which are treated in the proper way.

### 27.2 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 primary 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 math-

[^18]ematical 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.
- 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 999 999. The command \greeknumeral makes it possible to typeset Greek numerals. There is also an
\Greeknumeral "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.

```
27.1 〈*code〉
27.2\LdfInit\CurrentOption{captions\CurrentOption}
```

When the option polutonikogreek was used, redefine \CurrentOption to prevent problems later on.
27.3 \gdef $\backslash$ CurrentOption\{greek\}\%

When this file is read as an option, i.e. by the command,greekcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\l@greektoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
27.4\ifx\l@greek\@undefined
27.5 \@nopatterns{greek}
27.6 \adddialect\l@greek0\fi
```

Now we declare the polutoniko language attribute.
27.7 \bbl@declare@ttribute\{greek\}\{polutoniko\}\{\%

This code adds the expansion of \extraspolutonikogreek to \extrasgreek and changes the definition of \today for Greek to produce polytonic month names.
27.8 \expandafter $\backslash$ addto\expandafter \extrasgreek
27.9 \expandafter\{\extraspolutonikogreek\}\%

```
27.10 \let\captionsgreek\captionspolutonikogreek
27.11 \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.

```
27.12 \let\l@polutonikogreek\l@greek
27.13 \let\datepolutonikogreek\dategreek
27.14 \let\extraspolutonikogreek\extrasgreek
27.15 \let\noextraspolutonikogreek\noextrasgreek
27.16 }
```

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 Beccari ${ }^{23}$. 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 $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$, and if it isn't we abort.

```
27.17\InputIfFileExists{lgrenc.def}{%
27.18 \message{Loading the definitions for the Greek font encoding}}{%
27.19 \errhelp{I can't find the lgrenc.def file for the Greek fonts}%
27.20 \errmessage{Since I do not know what the LGR encoding means^^J
27.21 I can't typeset Greek. ~ J
27.22 I stop here, while you get a suitable lgrenc.def file}\@@end
27.23 }
```

Now we define two commands that offer the possibility to switch between Greek and Roman encodings.
\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.

```
27.24\DeclareRobustCommand{\greektext}{%
27.25 \fontencoding{LGR}\selectfont
27.26\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.

```
27.27\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 $\backslash$ textol.

```
27.28\def\outlfamily{\usefont{LGR}{cmro}{m}{n}}
27.29\DeclareTextFontCommand{\textol}{\outlfamily}
```

The next step consists in defining commands to switch to (and from) the Greek language.

[^19]\greekhyphenmins This macro is used to store the correct values of the hyphenation parameters \lefthyphenmin and \righthyphenmin.
27.30 \% Yannis Haralambous has suggested this value
27.31 \providehyphenmins\{\CurrentOption\}\{\@ne\@ne\}
\captionsgreek The macro \captionsgreek defines all strings used in the four standard document classes provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
27.32 \addto\captionsgreek{%
27.33 \def\prefacename{Pr'ologos}%
27.34 \def\refname{Anafor'es}%
27.35\def\abstractname{Per'ilhyh}%
27.36 \def\bibname{Bibliograf'ia}%
27.37 \def\chaptername{Kef'alaio}%
27.38\def\appendixname{Par'arthma}%
27.39 \def\contentsname{Perieq'omena}%
27.40 \def\listfigurename{Kat'alogos Sqhm'atwn}%
27.41 \def\listtablename{Kat'alogos Pin'akwn}%
27.42 \def\indexname{Euret'hrio}%
27.43 \def\figurename{Sq'hma}%
27.44\def\tablename{P'inakas}%
27.45 \def\partname{M'eros}%
27.46 \def\enclname{Sunhmm'ena}%
27.47 \def\ccname{Koinopo'ihsh}%
27.48 \def\headtoname{Pros}%
27.49 \def\pagename{Sel'ida}%
27.50 \def\seename{bl'epe}%
27.51 \def\alsoname{bl'epe ep'ishs}%
27.52 \def\proofname{Ap'odeixh}%
27.53\def\glossaryname{Glwss'ari}%
27.54 }
```

\captionspolutonikogreek For texts written in the $\pi o \lambda v \tau o \nu \kappa o ́$ (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.

```
27.55\let\captionspolutonikogreek\captionsgreek
27.56 \addto\captionspolutonikogreek{%
27.57 \def\refname{>Anafor'es}%
27.58\def\indexname{E<uret'hrio}%
27.59\def\figurename{Sq~hma}%
27.60\\def\headtoname{Pr'os}%
27.61 \def\alsoname{bl'epe >ep'ishs}%
27.62 \def\proofname{>Ap'odeixh}%
27.63}
```

\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.

```
27.64\def\gr@month{%
27.65 \ifcase\month\or
27.66 Ianouar'iou\or Febrouar'iou\or Mart'iou\or April'iou\or
27.67 Ma'"iou\or Ioun'iou\or Ioul'iou\or Augo'ustou\or
27.68 Septembr'iou\or Oktwbr'iou\or Noembr'iou\or Dekembr'iou\fi}
27.69 \def\dategreek{%
27.70\\def\today{\number\day \space \gr@month\space \number\year}}
    \gr@c@greek
27.71 \def\gr@c@month{%
27.72 \ifcase\month\or >Ianouar'iou\or
27.73 Febrouar'iou\or Mart'iou\or >April'iou\or Ma"'iou\or
27.74 >Ioun'iou\or >Ioul'iou\or A>ugo'ustou\or Septembr'iou\or
27.75 >Oktwbr'iou\or Noembr'iou\or Dekembr'iou\fi}
```

$\backslash$ 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.

```
27.76\def\Grtoday{%
27.77 \expandafter\Greeknumeral\expandafter{\the\day}\space
27.78 \gr@c@month \space
27.79 \expandafter\Greeknumeral\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.

```
27.80 \addto\extrasgreek{\greektext}
27.81 \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).

```
27.82\def\gr@ill@value#1{%
27.83 \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.

```
27.84 \DeclareRobustCommand\anw@false{%
27.85 \DeclareRobustCommand\anw@print{}}
27.86 \DeclareRobustCommand\anw@true{%
27.87 \DeclareRobustCommand\anw@print{\anwtonos}}
27.88 \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.

```
27.89 \def\greeknumeral#1{%
```

If the value is negative or zero nothing is printed and a warning is issued.

```
27.90 \ifnum#1<\@ne\space\gr@ill@value{#1}%
27.91 \else
27.92 \ifnum#1<10\expandafter\gr@num@i\number#1%
27.93 \else
27.94 \ifnum#1<100\expandafter\gr@num@ii\number#1%
27.95 \else
```

We use the available shorthands for $1.000(\backslash @ m)$ and 10.000 ( $\backslash @ M)$ to save a few tokens.
$27.96 \backslash i f n u m \# 1<\backslash @ m$ expandafter $\backslash$ gr@num@iii\number\#1\%
27.97 \else
$27.98 \backslash$ ifnum\#1<\@M\expandafter $\backslash g r @ n u m @ i v \backslash n u m b e r \# 1 \%$
27.99 \else
27.100 \ifnum\#1<100000 expandafter\gr@num@v $\backslash$ number\#1\%
27.101 \else
$27.102 \backslash i f n u m \# 1<1000000 \backslash$ expandafter $\backslash g r @ n u m @ v i \backslash n u m b e r \# 1 \%$
27.103 \else

If the value is too large, nothing is printed and a warning is issued.

```
27.104 \space\gr@ill@value{#1}%
27.105 \fi
27.106 \fi
27.107 \fi
27.108 \fi
27.109 \fi
27.110 \fi
27.111 \fi
27.112}
```

\Greeknumeral The command \Greeknumeral prints uppercase greek numerals. The parsing is performed by the macro \greeknumeral.
27.113 \def \Greeknumeral\#1\{\%
27.114 \expandafter\MakeUppercase\expandafter\{\greeknumeral\{\#1\}\}\}
$\backslash g r e e k @ a l p h \quad$ In the previous release of this language definition the commands $\backslash$ greek@aplh and \greek@Alph \greek@Alph were kept just for reasons of compatibility. Here again they become meaningful macros. They are definited 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
27.115 \let $\backslash$ latin@alph \@alph
$27.116 \backslash$ let $\backslash$ 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.

```
27.117\def\greek@alph#1{\expandafter\greeknumeral\expandafter{\the#1}}
27.118\def\greek@Alph#1{\expandafter\Greeknumeral\expandafter{\the#1}}
    Now we can set up the switching.
```

27.119 \addto\extrasgreek\{\%
27.120 \let\@alph\greek@alph
27.121 \let\@Alph\greek@Alph\}
27.122 \addto\noextrasgreek $\{\%$
27.123 \let\@alph\latin@alph
27.124 \let $\backslash @ A l p h \backslash l a t i n @ A l p h\}$
\greek@roman
To prevent roman numerals being typeset in greek letters we need to adopt the \greek@Roman internal $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$ 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.

```
27.125\let\latin@roman\@roman
27.126\let\latin@Roman\@Roman
27.127 \def\greek@roman#1{\textlatin{\latin@roman{#1}}}
27.128\def\greek@Roman#1{\textlatin{\latin@Roman{#1}}}
27.129 \addto\extrasgreek{%
27.130 \let\@roman\greek@roman
27.131 \let\@Roman\greek@Roman}
27.132 \addto\noextrasgreek{%
27.133 \let\@roman\latin@roman
27.134 \let\@Roman\latin@Roman}
```

\greek@amp The greek fonts do not contain an ampersand, so the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ command $\backslash \&$ dosn't \ltx@amp give the expected result if we do not do something about it.

```
27.135 \let\ltx@amp\&
27.136 \def\greek@amp{\textlatin{\ltx@amp}}
27.137 \addto\extrasgreek{\let\&\greek@amp}
```

27.138 \addto \noextrasgreek\{\let $\backslash \& \backslash$ ltx@amp\}

What is left now is the definition of a set of macros to produce the various digits.
$\backslash \mathrm{gr@num@i}$ 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.

```
27.139 \def\gr@num@i#1{%
27.140 \ifcase#1\or a\or b\or g\or d\or e\or \stigma\or z\or h\or j\fi
27.141 \ifnum#1=\z@\else\anw@true\fi\anw@print}
27.142\def\gr@num@ii#1{%
27.143 \ifcase#1\or i\or k\or l\or m\or n\or x\or o\or p\or \qoppa\fi
27.144 \ifnum#1=\z@\else\anw@true\fi\gr@num@i}
```

```
27.145\def\gr@num@iii#1{%
27.146 \ifcase#1\or r\or sv\or t\or u\or f\or q\or y\or w\or \sampi\fi
27.147 \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.

```
\gr@num@viz7.148\def\gr@num@iv#1{%
```

27.149 \ifnum\#1=\z@\else\katwtonos $\backslash f i$
27.150 \ifcase\#1\or $a \backslash o r ~ b \backslash o r ~ g \backslash o r ~ d \backslash o r ~ e \backslash o r ~ \ s t i g m a \backslash o r ~ z \backslash o r ~ h \backslash o r ~ j \backslash f i ~$
27.151 \gr@num@iii\}
27.152 \def $\backslash \mathrm{gr@num@v} \mathrm{\# 1} \mathrm{\{ } \mathrm{\%}$
$27.153 \backslash i f n u m \# 1=\backslash z @ \backslash e l s e \backslash k a t w t o n o s \backslash f i$
27.154 \ifcase\#1 \or $i \backslash o r ~ k \backslash o r ~ l \backslash o r ~ m \backslash o r ~ n \backslash o r ~ x \backslash o r ~ o \backslash o r ~ p \backslash o r ~ \ q o p p a \backslash f i$
27.155 \gr@num@iv\}
27.156 \def $\backslash \mathrm{gr@num@vi} \mathrm{\# 1} \mathrm{\{ } \mathrm{\%}$
27.157 \katwtonos

27.159 \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 $\sim$ is normaly used to produce an unbreakable space. The command $\backslash \sim$ normally produces a tilde accent. For polytonic Greek we change the definition of $\backslash \sim$ to produce the tilde character itself, making sure it has category code 12 .

```
27.160 \begingroup
27.161 \@ifundefined{active@char\string!}{}{\catcode'!=12\relax}
27.162 \catcode'\ = =12
27.163 \lccode'\!=`\~
27.164 \lowercase{\def\x{\endgroup
27.165 \def\greek@tilde{!}}\x}
27.166 \addto\extraspolutonikogreek{%
27.167\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 wich is different from that of an initial or medial sigma; the ' $v$ 'after an isolated sigma fools the ligature mechanism in order to typeset $\sigma$ in place of $\varsigma$. 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.
27.168 \addto\extrasgreek\{\%
27.169 \babel@savevariable\{\lccode'v\}\lccode'v='v\%
27.170 \babel@savevariable\{\lccode'\’\}\lccode'\'='<br>,\%
27.171 \babel@savevariable\{\lccode‘\"\}\lccode‘\"='\"\}

```
27.172 \addto\extraspolutonikogreek{%
27.173 \babel@savevariable{\lccode'\<}\lccode`\<='\<%
27.174 \babel@savevariable{\lccode`\>}\lccode`\>=`\>>%
27.175 \babel@savevariable{\lccode'\~}\lccode"\~='\ ~%
27.176 \babel@savevariable{\lccode"\|}\lccode'\|='\\/%
27.177 \babel@savevariable{\lccode'\'}\lccode'\'='\'}
```

And in order to get rid of all accents and breathings when a string is \uppercased we also change a number of uppercase codes.

```
27.178 \addto\extrasgreek{%
27.179 \babel@savevariable{\uccode'\"}\uccode`\"='\"%
27.180 \babel@savevariable{\uccode'\'}\uccode'\'=159} %% 159 == ~ - 9f
27.181 \addto\extraspolutonikogreek{%
27.182 \babel@savevariable{\uccode`\~ }\uccode`\~}=159
27.183 \babel@savevariable{\uccode'\>}\uccode'\>=159%
27.184 \babel@savevariable{\uccode'\<}\uccode'\<==159%
27.185 \babel@savevariable{\uccode'\|}\uccode'\|='\\%
27.186 \babel@savevariable{\uccode'\'}\uccode`\'=159}
```

For this to work we make the character ${ }^{\sim} 9 f$ a shorthand that expands to nothing. In order for this to work we need to make a character look like ^^9f in $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 's eyes. The trick is to have another character and assign it a different lowercase code. The execute the macros needed in a \lowercase environment. Usually the tile ~ character is used for such purposes. Before we do this we save it's original lowercase code to restore it once we're done.

```
27.187 \@tempcnta=\lccode`\~
27.188 \lccode`\~}=15
27.189 \lowercase{%
27.190 \initiate@active@char{~}%
27.191 \declare@shorthand{greek}{~}}{}
27.192\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.

```
27.193\addto\extraspolutonikogreek{\languageshorthands{greek}}%
```

27.194 \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 usefull, such as the $\epsilon v \rho \omega$ symbol, etc.
27.195 \DeclareTextCommand\{\anwtonos\}\{LGR\}\{\char"FE\relax\}
27.196 DeclareTextCommand\{\katwtonos\}\{LGR\}\{\char"FF\relax\}
27.197 \DeclareTextCommand\{\qoppa\}\{LGR\}\{\char"12\relax\}
27.198 \DeclareTextCommand\{\stigma\}\{LGR\}\{\char"06\relax\}
27.199 \DeclareTextCommand\{\sampi\}\{LGR\}\{\char"1B\relax\}
$27.200 \backslash$ DeclareTextCommand\{\Digamma\}\{LGR\}\{\char"C3\relax\}
27.201 \DeclareTextCommand\{\ddigamma\}\{LGR\}\{\char"93\relax\}
27.202 \DeclareTextCommand\{\vardigamma\}\{LGR\}\{\char"07\relax\}
27.203 \DeclareTextCommand\{\euro\}\{LGR\}\{\char"18\relax\}
27.204 \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.
$27.205 \% \backslash$ def $\backslash$ fnum@figure $\{\backslash$ figurename $\backslash$ nobreakspace $\backslash$ thefigure $\}$
$27.206 \% \backslash$ def $\backslash$ fnum@table $\{\backslash$ tablename $\backslash$ nobreakspace $\backslash$ 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.
27.207 \ldf@finish\{\CurrentOption\}
27.208 〈/code〉

## 28 The French language

The file frenchb. $\mathrm{dtx}^{24}$, derived from frenchy.sty, defines all the language definition macros for the French language.

Customization 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.

This file has been designed to be used with $\mathrm{EA}_{\mathrm{E}} \mathrm{X} 2 \varepsilon, \mathrm{I}_{\mathrm{E}} \mathrm{T}_{\mathrm{E}} \mathrm{X}-2.09$ and $\mathrm{Plain}_{\mathrm{E}} \mathrm{X}$ formats. If you are still using $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}-2.09$, you should consider switching to $\mathrm{LATEX}_{2}$ !

The command \selectlanguage\{french\} switches to the French language ${ }^{25}$, with the following effects:

1. French hyphenation patterns are made active;
2. 'double punctuation' is made active for correct spacing in French;
3. \today prints the date in French;
4. the caption names are translated into French ( $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ only);
5. the default items in itemize environment are set to '-' instead of $\bullet$, and all vertical spacing and glue is deleted, a hook to reset standard $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ spacing is provided ( $\backslash$ FrenchItemizeSpacingfalse); it is possible to change '-' to something else ('-' for instance) by redefining \FrenchLabelItem; apart from the global hook $\backslash$ FrenchLabelItem, it is also possible to change the 'labelitems' at any level ( 1 to 4) in French, using the standard IATEX syntax, for instance: \{•\}; in order to be effective in French, the redefinitions have to be made when French is the current language (i.e. after the $\backslash$ begin\{document\}), the changes are saved when switching to another language and will be remembered of, when switching back to French;
6. vertical spacing in general LATEX lists is shortened, a hook to reset standard LATEX settings is provided ( $\backslash$ FrenchListSpacingfalse) ;
7. the first paragraph of each section is indented ( $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ only);
8. the space after \dots is removed in French.

Some commands are provided in frenchb to make typesetting easier:

1. French quotation marks can be entered using the commands $\backslash o g$ and $\backslash f g$ which work in $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}, \mathrm{IAT}_{\mathrm{E}} \mathrm{X}-2.09$ and PlainT $\mathrm{T}_{\mathrm{E}} \mathrm{X}$, their appearance depending on what is available to draw them; if you use ${ }^{\mathrm{A}} \mathrm{T}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ and T1-encoding you can also enter them as <<~French quotation marks~>> but then don't

[^20]forget the unbreakable spaces, ( kslash\mathrm{og}\)and$\backslash\mathrm{fg}$provideforcorrectlinebreaksandbetterhorizontalspacing).$\backslash\mathrm{og}$and$\backslash\mathrm{fg}$canbeusedoutsideFrench,theyprovidethenEnglishquotes"and".2.Acommand\upisprovidedtotypesetsuperscriptslike$M\backslashup\{me\}$(abbreviationfor"Madame"),$1\backslashup\{er\}$(for"premier").3.Familynamesshouldbetypesetinsmallcapitalsandneverbehyphenated,themacro\bsc(boxedsmallcaps)doesthis,e.g.,Leslie~$\backslash$bsc\{Lamport$\}$willproduceLeslieLamport.4.Commands\primo,\secundo,\tertioand\quartomaybeusedtoenumerateinlists.5.Abbreviationsfor"Numéro"and"numéro"areobtainedviathecommands$\backslash$No,\no.6.Twocommandsareprovidedtotypesetthesymbolfor"degré":\degreprintstherawcharacterand\degresshouldbeusedtotypesettemperatures(e.g.,"20~$\backslash$degresC"withanunbreakablespace),orforalcohols'strengths(e.g.,"$45\backslash$degres"withnospaceinFrench).7.Inmathmodethecommahastobesurroundedwithbracestoavoidaspuriousspacebeinginsertedafterit,indecimalnumbersforinstance(seethe$\mathrm{T}_{\mathrm{E}}\mathrm{Xbook}$p.134).Thecommand\DecimalMathCommamakesthecommabeanordinarycharacterinFrenchonly(nospaceadded);asacounterpart,if$\backslash$DecimalMathCommaisactive,anexplicitspacehastobeaddedinlistsandintervals:$\$[0,\backslash1]\$$,$\$(x,\backslashy)\$$.$\backslash$StandardMathCommaswitchesbacktothestandardbehaviourofthecomma.8.Acommand$\backslash$nombreisprovidedtoeasilytypesetnumbers:itworksbothintextandinmathmode:inputting\nombre\{3141,592653\}willformatthisnumberproperlyaccordingtothecurrentlanguage(FrenchornonFrench):eachsliceofthreedigitswillbeseparatedeitherwithacommainEnglishorwithaspaceinFrench(ifyoupreferathinspaceinsteadofanormalspace,justaddthecommand\ThinSpaceInFrenchNumberstothepreambleofyourdocument,ortofrenchb.cfg).Thecommand$\backslash$nombreisacontributionofVincentJalbyusingideasofDavidCarlisleincomma.sty.9.frenchbhasbeendesignedtotakeadvantageofthexspacepackageifpresent:adding\usepackage\{xspace\}inthepreamblewillforcemacroslike$\backslash\mathrm{fg}$,\ier,\ieme,\dots,...,torespectthespacesyoutypeafterthem,forinstancetyping'1\ierjuin'willprint'1erjuin'(noneedforaforcedspaceafter1\ier).10.Twocommands,\FrenchLayoutand\StandardLayout(tobeusedonlyinthepreamble)areprovidedtounifythelayoutofmultilingualdocuments(itmainlyconcernslists)regardlessthecurrentlanguage(seesection29.10fordetails).11.Twocommands\AddThinSpaceBeforeFootnotesand$\backslash$FrenchFootnotesenabletocustomizethelayoutoffootnotesfollowingtheFrenchIN'sspecifications;thefirstone,whichcanonlybeusedbeforethe\begin\{document\},}addsathinspaceintherunningtextbeforethenumberorsymbolcallingthefootnote;thesecondonemodifiesthefootnote'slayout:ittypesetsleadingnumbersas'1.'insteadof'${}^{1}$'.\FrenchFootnoteshasnoeffectonfootnotesnumberedwithsymbols(asinthe\thankscommand).Noneofthesecommandsareactivebydefault,bothcanbebestaddedtofrenchb.cfgortothepreambleofthedocument.\FrenchFootnotesand$\backslash$StandardFootnotes,whichcancelstheeffectof$\backslash$FrenchFootnotesforthetextcomingnext,canbeusedanywhereinthedocument.Thecommand$\backslash$StandardFootnotesmaybeusefulwhensomefootnotesarenumberedwithletters(insideminipagesforinstance).$\backslash$AddThinSpaceBeforeFootnotesand$\backslash$FrenchFootnotesactonallfootnoteswhateverthecurrentlanguageis.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

All commands previously available in francais.ldf have been included in frenchb.ldf for compatibility, sometimes with updated definitions.

The french package, by Bernard Gaulle, was not designed to run with babel (although the latest versions claim to be babel compatible), but rather as a standalone package for the French language. It provides many more functionalities (like \lettrine, \sommaire...) not available in frenchb, at the cost of a much greater complexity and possible incompatibilities with other languages.

As french is known to produce the best layout available for French typography, I have borrowed many ideas from Bernard's file. I did my best to help users of both packages (french and frenchb) to exchange their sources files easily (see the example configuration file below in section 28 ), with one exception which affects the way French quotation marks are entered: frenchb uses macros ( $\backslash \mathrm{og}$ and $\backslash \mathrm{fg}$ ) while french uses active characters (<< and >>).

French typographic rules specify that some white space should be present before 'double punctuation' characters. These characters are ; ! ? and :. In order to get this white space automatically, the category code of these characters is made \active. In French, the user should input these four characters preceded with a space, 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 the rule in French typography). It's up to the user to add or not a space after 'double punctuation' characters: usually a space is necessary, but not always (before a full point or a closing brace for instance), so this cannot done automatically.

In (rare) cases where no space should be added before a 'double punctuation', either use \string; \string: \string! \string? instead of ; : ! ?, or switch locally to English. For instance you can type C\string:TEX or \begin\{otherlanguage\}\{english\}\{C:TEX\}\end\{otherlanguage\} to avoid the } space before : in a MS-DOS path.

Some users dislike this 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 frenchb.cfg, or anywhere in a document) frenchb 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 $\backslash$ AutoSpaceBeforeFDP switches back to the default behaviour of frenchb.

Once you have built your format, a good precaution would be to perform some basic tests about hyphenation in French. For $\mathrm{EAT}_{\mathrm{E}} 2_{\varepsilon}$ I suggest this:

- run the following file, with the encoding suitable for your machine ( $m y$ encoding will be latin1 for UNIX machines and PCs running Windows, applemac for Macintoshs, or cp850 for PCs running DOS).

```
%%% Test file for French hyphenation.
\documentclass{article}
\usepackage[my-encoding]{inputenc}
\usepackage[T1]{fontenc} % Use EC fonts for French
%\usepackage{aeguill} % Uncomment this line and
                                    % comment out the preceeding one
                                    % to use AE virtual fonts.
```

\usepackage[francais]\{babel\}$\backslash$begin\{document\}\showhyphens\{signalcontainer\'ev\'enementalg\‘ebre\}\showhyphens\{signalcontainerévénementalgèbre\}\end\{document\}}undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

- check the hyphenations proposed by $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 EC fonts with built-in accented characters or M1TEX with CM fonts avoids this type of mismatch.
frenchb has been improved using helpful suggestions from many people, the main contributions came from Vincent Jalby. Thanks to all of them!


## Changes

First version released: 1.1 as of 1996/05/31 part of babel-3.6beta.
Changes in version 1.1b: update for babel-3.6.
Changes in version 1.2: new command \nombre to format numbers; removed command \fup borrowed from the french package ( $\backslash$ up does a better job in $\operatorname{LA}_{E} \mathrm{X} 2_{\varepsilon}$ ); also removed aliases $\backslash$ french and \english (frenchb.cfg is a better place for these).

Changes in version 1.3:

- The 'xspace' package, when present, now controls spacing after all (sensible) macros, formerly 'xspace' worked on \fg (suggested by Vincent Jalby);
- spacing after opening and before closing guillemets improved as suggested by Thierry Bouche;
- a replacement for poor looking guillemets in OT1 encoding (math fonts are used to emulate them) is provided if the file ot2wncyr.fd is found: this file defines an OT2 encoding using AMS Cyrillic fonts; these have builtin guillemets, they are text fonts, are free, and are distributed as METAFONT and Type1 fonts (good for pdftex!), the replacement was suggested by Gérard Degrez; if the files ot2wncyr.fd, wncy*.mf, wncy*.tfm, and wncy*.pfb aren't available on your system, you should get them from CTAN; if your system complains about missing wncy*.tfm fonts, it means your $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ system is incomplete, as a quick fix, you can either remove ot2wncyr.fd or add the command \LasyGuillemets to the preamble of your document or to frenchb.cfg, then frenchb will behave as it did in the previous versions. In case Cyrillic guillemets do not fit, it is possible to pick French guillemets from any font using the command $\backslash$ FrenchGuillemetsFrom which requires 4 arguments \FrenchGuillemetsFrom\{coding\}\{font\}\{char-left\}\{char-right\}, for instance, add $\backslash$ FrenchGuillemetsFrom\{T1\}\{ppl\}\{19\}\{20\} to the preamble of your document or to frenchb.cfg, to pick French guillemets from Palatino (fontname $=$ ppl, char 19 and 20 are left and right guillemets in T1-encoding). This was suggested by Michel Bovani.
- the environment 'itemize' has been redesigned in French according to specifications from Jacques André and Thierry Bouche; it is possible to switch back to the settings of version 1.2: add $\backslash$ FrenchItemizeSpacingfalse after loading frenchb in the preamble (this can be useful to process old documents);
- two switches have been added to go back to standard $\mathrm{AAT}_{\mathrm{E}} \mathrm{X}$ list spacing $\backslash$ FrenchItemizeSpacingfalse and \FrenchListSpacingfalse;
- $\backslash$ nombre now properly handles signs in $\mathrm{AT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$;
- definition of \dots changed in French;
- in French, with the standard $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ classes, captions in figures and tables are printed with an endash instead of a colon.


## Changes in version 1.4:

- the redefinition of $\backslash @$ makecaption is changed not to overwrite the changes made by some classes (koma-script, amsart, ua-thesis...) as pointed out by Werner Lemberg;
- a hook, \FrenchGuillemetsFrom, is provided to pick French guillemets from any font (suggested by Michel Bovani, works only in OT1-encoding);
- a hook, \FrenchLabelItem, is provided to enable marks other than '-' ( $\backslash$ textendash) in French lists (also suggested by Michel Bovani);
- \DecimalMathComma, \StandardMathComma, \ThinSpaceInFrenchNumbers added;
- \FrenchLayout and \StandardLayout added;
- an example of customization file frenchb.cfg is included in frenchb.dtx.


## Changes in version 1.5:

- The settings for spacing in French lists are no longer tuned at the \@trivlist level but within \list; this enables the users to provide their own settings for the lists they define with \list (suggested by P. Pichaureau). Although the layout of the standard lists has not changed, some lists, based directly on \@trivlist or \trivlist, could possibly be typeset differently when upgrading from version 1.4 to 1.5. The command $\backslash$ FrenchOldTrivlisttrue could then, be useful to process older documents.
- Apart from the global hook $\backslash$ FrenchLabelItem, it is now also possible to change the 'labelitems' at any level (1 to 4) in French, using the standard ${ }^{\mathrm{A}} \mathrm{T}_{\mathrm{E}} \mathrm{X}$ syntax, for instance: \{•\}.
- The internal name for the French language has been changed from frenchb to french, it means that \captionsfrench, \datefrench, \extrasfrench, \noextrasfrench, are to be used now instead of \captionsfrenchb, \datefrenchb, \extrasfrenchb, \noextrasfrenchb.
- From version 1.5 f on, it is possible to customize the space inserted before a colon in French; its default value is \space (as in previous versions of frenchb), but it can now be changed to \thinspace by redefining $\backslash$ Fcolonspace.


## Changes in version 1.6:

- Two new commands are provided: \AddThinSpaceBeforeFootnotes and $\backslash$ FrenchFootnotes; they enable to typeset footnotes according to the French IN's specifications (suggested by Jacques André).
- frenchb will make use of \textdegree (TS1 encoding) to typeset degrees whenever possible.
- Default guillemets have been changed in OT1 encoding only in version 1.6c: the former ones were picked up in the wncyr fonts, which forced the cyrillic encoding to OT2 when babel was loaded with option frenchb before a cyrillic language (russian, ukrainian or bulgarian). Now, LM fonts for T1-encoding are available, we pick up the guillemets from them, for OT1encoding.
- The config file frenchb. cfg is now loaded only when $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ is used (from version 1.6 g ).
- The command \degres now works also in math mode (from version 1.6 g ), but not in bold versions (it is a text character basically).


## Customizing frenchb: an example of configuration file

28.1 〈*cfg〉
$28.2 \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \%$
$28.3 \%$ If you want to customize frenchb, please DO NOT hack into the code,
$28.4 \% \%$ copy this file into a directory searched by TeX, preferably a
$28.5 \%$ personal one on multi-user systems, and customize it as you like.
$28.6 \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \%$
$28.7 \% \%$ WARNING: If you exchange your documents with colleagues using
$28.8 \% \%$ a different TeX installation, it is best NOT TO HAVE a frenchb.cfg
$28.9 \%$ file, and add instead the customization commands to the preamble
$28.10 \%$ of your documents after babel and frenchb have been loaded.
$28.11 \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \% \%$
$28.12 \%$
$28.13 \%$ Uncomment the following line if you dislike frenchb automatically
$28.14 \% \%$ adding a space before French double punctuation (see frenchb.dtx
$28.15 \%$ for more information).
$28.16 \%$
$28.17 \% \%$ NoAutoSpaceBeforeFDP
$28.18 \%$
$28.19 \%$ frenchb inserts a normal space before a colon in French (following
$28.20 \%$ I.N. specifications), but some people prefer a thinspace instead;
$28.21 \%$ uncomment the following line if you prefer a thinspace.
$28.22 \%$
$28.23 \% \%$ renewcommand $\{\backslash$ Fcolonspace $\}\{\backslash$ thinspace $\}$
$28.24 \%$
$28.25 \%$ Uncomment the following line to force lasy font emulation for
$28.26 \%$ French guillemets in OT1 encoding; Polish guillemets are
$28.27 \% \%$ FAR BETTER LOOKING, you should normally leave this line commented out. $28.28 \%$
$28.29 \% \%$ LasyGuillemets
$28.30 \% \%$
$28.31 \%$ The following option is for backward compatibility ONLY
$28.32 \%$ (it might conflict with the russian option of babel).
$28.33 \%$ Uncomment the following line if you want to get back to $28.34 \%$ the default guillemets used in versions \$<\$ 1.6b for OT1-encoding $28.35 \%$ (no effect in T1-encoding).
$28.36 \%$
$28.37 \% \%$ \CyrillicGuillemets
$28.38 \% \%$
$28.39 \% \%$ If you dislike Polish guillemets in OT1-encoding, you can easily $28.40 \%$ pick up yours from other fonts, examples below: Palatino or
$28.41 \%$ cyrillic CMR T2A encoded font.
$28.42 \% \%$ The four arguments are: encoding, font name, character codes of $28.43 \%$ the opening and closing guillemets in the selected font.
$28.44 \% \%$
$28.45 \% \%$ FrenchGuillemetsFrom $\{$ T1\} $\{$ ppl $\}\{19\}\{20\}$
$28.46 \% \%$ FrenchGuillemetsFrom\{T2A\}\{cmr\}\{190\}\{191\}
$28.47 \% \%$
$28.48 \% \%$ Uncomment the following line if you don't want frenchb to change $28.49 \% \%$ LaTeX standard settings for vertical spacing in 'itemize' lists. $28.50 \% \%$ LaTeX standard settings were used by frenchb in versions < 1.3, $28.51 \%$ so you may want to uncomment the following line to process older $28.52 \% \%$ documents.
$28.53 \% \%$
$28.54 \% \%$ (FrenchItemizeSpacingfalse
$28.55 \%$
$28.56 \%$ frenchb tunes vertical spaces in all lists (at \list level), $28.57 \%$ uncomment the following line if you prefer LaTeX standard settings $28.58 \% \%$ (this flag is new in version 1.3 ). When \ifFrenchItemizeSpacing is $28.59 \%$ 'true' (default setting), this flag has no effect on itemize lists. $28.60 \% \%$
$28.61 \% \% \backslash$ FrenchListSpacingfalse
$28.62 \% \%$
$28.63 \% \%$ The command \FrenchLabelItem can be used to set the 'labelitems' $28.64 \% \%$ for all levels in French. For instance, to get long dashes, use: $28.65 \%$
$28.66 \% \%$ renewcommand $\{\backslash$ FrenchLabelItem $\}\{$ textemdash $\}$
$28.67 \% \%$
$28.68 \% \%$ You can also have different 'labelitems' for each level in French: $28.69 \%$
$28.70 \% \%$ \{—\}
$28.71 \% \%$ \{\bfseries $\backslash$ textendash\}
$28.72 \% \%$ \{•\}
$28.73 \% \%$ renewcommand $\{\backslash$ Frlabelitemiv $\}\}$
$28.74 \% \%$
$28.75 \%$ Uncomment the following line if you want a thinspace to be added $28.76 \% \%$ in the running text before the footnote number or symbol.
$28.77 \% \%$
$28.78 \% \%$ \AddThinSpaceBeforeFootnotes
28.79 \% \%
$28.80 \%$ Uncomment the following line if you want footnotes to be typeset $28.81 \%$ according the specifications of the French 'Imprimerie Nationale'. $28.82 \% \%$
$28.83 \% \%$ \FrenchFootnotes
$28.84 \%$
$28.85 \%$ Uncomment the following line if you want the command \nombre to $28.86 \%$ insert thin spaces (1/6em) instead of normal spaces (0.3em) $28.87 \%$ between slices of 3 digits when typesetting numbers in French. $28.88 \% \%$
$28.89 \% \%$ \ThinSpaceInFrenchNumbers
$28.90 \% \%$
$28.91 \%$ If you are bored to type $\{$,$\} every time you want a decimal comma in$ $28.92 \%$ math mode, uncomment the following line (and don't forget to add a $28.93 \%$ forced space after the comma in lists and intervals: $\$(x, \backslash y) \$$ and $28.94 \% \%$ [0,\1]\$). I suggest NOT TO UNCOMMENT the following line, $28.95 \%$ and to switch between the decimal comma and the standard comma $28.96 \%$ using \DecimalMathComma and \StandardMathComma inside the document. $28.97 \% \%$
$28.98 \% \%$ \DecimalMathComma
$28.99 \%$
$28.100 \%$ If you do not want the appearance of lists to depend on the current $28.101 \%$ language, add ONE of these two commands: \FrenchLayout (for a
$28.102 \%$ document mainly in French) or \StandardLayout (for a document
$28.103 \% \%$ mainly in English).
$28.104 \% \%$
$28.105 \%$ \% \FrenchLayout
$28.106 \% \%$ StandardLayout
$28.107 \% \%$
$28.108 \% \%$ With the standard LaTeX classes (article.cls, report.cls, book.cls)
$28.109 \%$ you can change the separator used in figure and table captions in
$28.110 \% \%$ French with \CaptionSeparator, for instance (the default value of
$28.111 \% \%$ \CaptionSeparator in French is '\space–\space'):
28.112 \% \%
$28.113 \% \%$ addto \captionsfrench\{ \def $\backslash$ CaptionSeparator $\{\backslash$ space $\backslash$ textemdash $\backslash$ space $\}\}$ $28.114 \% \%$
$28.115 \% \%$ You might want to change some of the translations of caption names, $28.116 \% \%$ you can do it this way, for instance:
28.117 \% \%
$28.118 \% \%$ addto $\backslash$ captionsfrench\{\def $\backslash$ figurename\{\{\scshape Figure $\}\}\}$
$28.119 \% \%$ addto\captionsfrench\{\def \tablename\{\{\scshape Table\}\}\}
$28.120 \% \%$ \addto\captionsfrench\{\def $\backslash$ proofname\{Preuve\}\}
$28.121 \% \%$
$28.122 \% \%$ If French guillemets are available on your keyboard, you can use
$28.123 \%$ them instead of the commands log and \fg: REPLACE << and >> in
$28.124 \% \%$ the following code by your ready-made guillemets (2 occurrences $28.125 \%$ for each) AND uncomment the resulting code.
$28.126 \%$ WARNINGS :
$28.127 \% \%$ 1) This will reduce the portability of your source files!
$28.128 \% \%$ 2) This adds two active characters.

```
28.129%% 3) If you add this stuff to the preamble of a document, you'll
28.130 %%
need to wrap it in \makeatletter ... \makeatother.
28.131 %%
28.132%% \ifx\mule@def \undefined
28.133%% \initiate@active@char{<<}
28.134%% \initiate@active@char{>>}
28.135%% \AtBeginDocument{%
28.136%% \def<<{\og\ignorespaces}%
28.137%% \def>>{{\fg}}%
28.138%% }%
28.139%% \else %%%% For the CJK package, see MULEenc.sty.
28.140%% \mule@def{11}{\og\ignorespaces}
28.141%% \mule@def{27}{{\fg}}
28.142 %% \fi
28.143 %%
28.144%% The rest is for compatibility with Bernard Gaulle's french.sty:
28.145%% if you want to typeset with babel/frenchb files originally written
28.146 %% for french.sty, you will probably want to uncomment these lines.
28.147%%
28.148%% Uncomment these lines if you want the aliases \numero and \Numero
28.149%% (as in french.sty) for \no and \No.
28.150 %%
28.151%%\let\numero=\no
28.152%%\let\Numero=\No
28.153 %%
28.154%% Uncomment the following lines if you want to substitute '\bsc' to
28.155 %% either (or both) of french.sty macros '\fsc' and '\lsc'.
28.156 %%
28.157 %%\let\fsc=\bsc
28.158%%\let\lsc=\bsc
28.159 %%
28.160%% Uncomment the following line if you want to substitute '\up' to
28.161 %% french.sty macro '\fup'.
28.162 %%
28.163 %%\let\fup=\up
28.164 %%
28.165%% Uncomment the following lines if you intend to use commands \french
28.166%% and \english to switch between languages, as in french.sty.
28.167 %%
28.168%%\def\french{\leavevmode\selectlanguage{french}}
28.169%%\def\english{\leavevmode\selectlanguage{english}}
28.170 %%
28.171%% Uncomment the following lines if you intend to use the command
28.172%% \AllTeX to refer to all flavours of TeX, (this command has been
28.173 %% made popular by french.sty).
28.174%%
28.175 %%\newcommand*{\\AllTeX}{%
28.176 %%
28.177 %%
28.178 </cfg>
```

(L\kern-.36em\raise.3ex\hbox\{\sc a\} \kern-.15em)\% T\kern-. $1667 \mathrm{em} \backslash$ lower. $7 \mathrm{ex} \backslash$ hbox $\{\mathrm{E}\} \backslash$ kern-. 125 emX$\}$

## $29 \mathrm{~T}_{\mathrm{E}} \mathrm{Xnical}$ details

### 29.1 Initial setup

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

```
29.1 <*code\rangle
29.2%%
29.3\LdfInit\CurrentOption\datefrench
```

\ifLaTeX To check the format in use (plain, $\mathrm{L}_{\mathrm{A}} \mathrm{T}_{\mathrm{E}} \mathrm{X}$ or $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ ), we'll need two new 'if'.
\ifLaTeXe 29.4 \newif $\backslash$ ifLaTeX
29.5 \ifx $\backslash$ magnification\@undefined $\backslash$ LaTeXtrue $\backslash f i$
29.6 \newif $\backslash i f L a T e X e$
29.7 \ifx\@compatibilitytrue\@undef ined $\backslash e l s e \backslash L a T e X e t r u e \backslash f i$
\if@Two@E We will need another 'if' : \if@Two@E is true if and only if $\mathrm{LAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ is running not in compatibility mode. It is used in the definitions of the command \nombre and \up. The definition is somewhat complicated, due to the fact that \if@compatibility is not recognized as a $\backslash$ if in $\mathrm{EA}_{\mathrm{E}} \mathrm{X}-2.09$ based formats.

```
29.8\newif\if@Two@E \@Two@Etrue
29.9 \def\@FI@{\fi}
29.10\ifx\@compatibilitytrue\@undefined
29.11\@Two@Efalse \def\@FI@{\relax}
29.12\else
29.13\if@compatibility \@Two@Efalse \fi
29.14\@FI@
```

Check if hyphenation patterns for the French language have been loaded in language.dat; we allow for the names 'french', 'francais', 'canadien' or 'acadian'. The latter two are both names used in Canada for variants of French that are in use in that country.

```
29.15\ifx\l@french\@undefined
29.16 \ifx\l@francais\@undefined
29.17 \ifx\l@canadien\@undefined
29.18 \ifx\l@acadian\@undefined
29.19 \@nopatterns{French}
29.20 \adddialect\l@french0
29.21 \else
29.22 \let\l@french\l@acadian
29.23 \fi
29.24 \else
29.25 \let\l@french\l@canadien
29.26 \fi
29.27 \else
29.28 \let\l@french\l@francais
29.29 \fi
29.30\fi
```

After the statement above \1@french is sure to always be defined.
The internal name for the French language will be 'french'; 'francais' and 'frenchb' will be synonymous for 'french': first let both names use the same hyphenation patterns. Later we will have to set aliases for \captionsfrench, \datefrench, \extrasfrench and \noextrasfrench. As French uses the standard values of \lefthyphenmin (2) and $\backslash$ righthyphenmin (3), no special setting is required here.

```
29.31 \def\CurrentOption{french}
29.32\ifx\l@francais\@undefined
29.33 \let\l@francais\l@french
29.34\fi
29.35\ifx\l@frenchb\@undefined
29.36 \let\l@frenchb\l@french
29.37\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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$.

```
29.38\ifx\l@canadien\@undefined
29.39 \let\l@canadien\l@french
29.40 \fi
29.41 \ifx\l@acadian\@undefined
29.42 \let\l@acadian\l@french
29.43\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 francais are synonymous. As some users who choose frenchb or francais as option of babel, might customize \captionsfrenchb or \captionsfrancais in the preamble, we merge their changes at the \begin\{document\} when they do so (only possible in } \mathrm { IA } _ { \mathrm { E } } \mathrm { X } 2 \varepsilon ). The other three commands for \date, \extras, and \noextras are not likely to be modified by end-users. The other variants of languages are defined by checking if the relevant option was used and then adding one extra level of expansion.

```
29.44\def\datefrancais{\datefrench}
29.45\def\extrasfrancais{\extrasfrench}
    \def\noextrasfrancais{\noextrasfrench}
    \def\datefrenchb{\datefrench}
    \def\extrasfrenchb{\extrasfrench}
    \def\noextrasfrenchb{\noextrasfrench}
    \ifLaTeXe
    \AtBeginDocument{\let\captions@French\captionsfrench
    \ifx\captionsfrenchb\@undefined
    \let\captions@Frenchb\relax
    \else
    \let\captions@Frenchb\captionsfrenchb
    \fi
    \ifx\captionsfrancais\@undefined
```

```
29.58 \let\captions@Francais\relax
2 9 . 5 9
29.60
29.61
29.62
29.63
29.64
9.65
29.66
29.67
29.68\else
29.69\def\captionsfrancais{\captionsfrench}
29.70 \def\captionsfrenchb{\captionsfrench}
29.71 \fi
29.72 \@ifpackagewith{babel}{canadien}{%
29.73\def\captionscanadien{\captionsfrench}%
29.74 \def\datecanadien{\datefrench}%
29.75 \def\extrascanadien{\extrasfrench}%
29.76\def\noextrascanadien{\noextrasfrench}%
29.77 }{}
29.78 \@ifpackagewith{babel}{acadian}{%
29.79\def\captionsacadian{\captionsfrench}%
29.80 \def\dateacadian{\datefrench}%
29.81 \def\extrasacadian{\extrasfrench}%
29.82 \def\noextrasacadian{\noextrasfrench}%
29.83 }{}
```

\extrasfrench The macro \extrasfrench will perform all the extra definitions needed for the \noextrasfrench French language. The macro \noextrasfrench is used to cancel the actions of \extrasfrench.

In French "apostrophe" is used in hyphenation in expressions like l'ambulance (French 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 to be reset to null when exiting French.

```
29.84\@namedef{extras\CurrentOption}{\lccode'\'='\'}
29.85\@namedef{noextras\CurrentOption}{\lccode`\'=0}
```

It is best to use $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ 's font changing commands, and to emulated those we need when they are not available, as in PlainTEX or $\mathrm{LA}_{\mathrm{E}} \mathrm{X}-2.09$. Be aware that old commands \sc, \it, etc. exist in $\mathrm{HA}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$, but they behave like they did in $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}-2.09$ (i.e., they switch back to \normalfont instead of keeping the other font attributes unchanged).

```
29.86 \ifx\scshape\@undefined
29.87 \ifx\sc\@undefined
            \let\scshape\relax
    \else
        \(\backslash\) let \(\backslash s c s h a p e \backslash s c\)
    \(\backslash f i\)
29.92 \fi
```

```
9.93\ifx\emph\@undefined
29.94\ifx\em\@undefined
9.95\\let\emph\relax
29.96 \else
9.97\\def\emph#1{\em #1}
29.98 \fi
29.99\fi
```


### 29.2 Caption names

The next step consists of defining the French equivalents for the $\mathrm{LA}_{\mathrm{E}} \mathrm{X}$ caption names.

In French, captions in figures and tables should be printed with endash ('') instead of the standard ' $\because$ '. If another separator is preferred, see in the example configuration file above (section 28) how to change the default value of \CaptionSeparator.

We first store the standard definition of \@makecaption (e.g., the one provided in article.cls, report.cls, book.cls which is frozen for $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ according to Frank Mittelbach), in \STD@makecaption. '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

```
ifLaTeXe
    \long\def\STD@makecaption#1#2{%
        \vskip\abovecaptionskip
        \sbox\@tempboxa{#1: #2}%
        \ifdim \wd\@tempboxa >\hsize
            #1: #2\par
        \else
            \global \@minipagefalse
            \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
        \fi
        \vskip\belowcaptionskip}
    \def\CaptionSeparator{\string:\space}
    \long\def\FB@makecaption#1#2{%
        \vskip\abovecaptionskip
        \sbox\@tempboxa{#1\CaptionSeparator #2}%
        \ifdim \wd\@tempboxa >\hsize
            #1\CaptionSeparator #2\par
        \else
            \global \@minipagefalse
            \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
        \fi
        \vskip\belowcaptionskip}
    \AtBeginDocument{%
```

```
29.123 \ifx\@makecaption\STD@makecaption
29.124 \let\@makecaption\FB@makecaption
29.125 \else
29.126 \ifx\@makecaption\@undefined\else
29.127 \PackageWarning{frenchb.ldf}%
29.128 {The definition of \protect\@makecaption\space
29.129 has been changed,\MessageBreak
29.130 frenchb will NOT customize it;\MessageBreak reported}%
29.131 \fi
29.132 \fi
29.133\let\FB@makecaption\relax
29.134 \let\STD@makecaption\relax}
29.135 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
29.136 \def\CaptionSeparator{\string:\space}}
29.137\fi
```

\captionsfrench The macro \captionsfrench defines all strings used in the four standard document classes provided with $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$. Some authors do not like some of these names; it is easy to change them in the preamble after loading frenchb (or in your file frenchb.cfg), e.g \addto\captionsfrench\{\def\figurename\{Figure\}\} will print 'Figure' in Roman instead of 'Fig.'.

```
38\ifLaTeX
```

39 \@namedef\{captions \CurrentOption\}\{\%
29.140 \def \refname\{R\'ef\'erences\}\%
29.141 \def\abstractname\{R\'esum\'e\}\%
29.142 \def \bibname\{Bibliographie\}\%
$29.143 \backslash$ def $\backslash p r e f a c e n a m e\{\operatorname{Pr} \backslash ' e f a c e\} \%$
29.144 \def\chaptername\{Chapitre\}\%
29.145 \def $\backslash$ appendixname\{Annexe\}\%
29.146 \def \contentsname\{Table des mati\'eres\}\%
29.147 \def \listfigurename\{Table des figures\}\%
29.148 \def \listtablename\{Liste des tableaux\}\%
29.149 \def \indexname\{Index\}\%
29.150 \def \figurename\{\{\scshape Fig.\}\}\%
29.151 \def $\backslash$ tablename\{\{\scshape Tab.\}\}\%
$29.152 \backslash$ def $\backslash$ CaptionSeparator\{\space–\space\}\%
"Première partie" instead of "Part I"
$29.153 \backslash$ def $\backslash$ partname $\{$ protect $\backslash @$ Fpt partie\}\%
$29.154 \backslash$ def \@Fpt\{\{\ifcase\value\{part\}\or Premi\'ere\or Deuxi\'eme\or
29.155 Troisi\'eme\or Quatri\'eme\or Cinqui\'eme\or Sixi\'eme\or
29.156 Septi\'eme\or Huiti\'eme\or Neuvi\'eme\or Dixi\'eme\or Onzi\'eme\or
29.157 Douzi\'eme\or Treizi\'eme\or Quatorzi\'eme\or Quinzi\'eme\or
29.158 Seizi\'eme\or Dix-septi\'eme\or Dix-huiti\'eme\or Dix-neuvi\'eme\or
29.159 Vingti\'eme\fi\}\space\def\thepart\{\}\}\%
29.160 \def $\backslash$ pagename\{page\}\%
29.161 \def \seename\{\{\emph\{voir\}\}\}\%
29.162 \def \alsoname\{\{\emph\{voir aussi\}\}\}\%
29.163 \def \enclname\{P.~J. \}\%

```
29.164 \def\ccname{Copie \'a }%
29.165 \def\headtoname{}%
29.166 \def\proofname{D\'emonstration}% for AMS-\LaTeX
29.167 \def\glossaryname{Glossaire}%
29.168 }
29.169 \fi
```


### 29.3 Punctuation

The 'double punctuation' characters (; ! ? and :) have to be made \active for an automatic control of the amount of space to insert before them.

```
29.170 \initiate@active@char{:}
29.171 \initiate@active@char{;}
29.172 \initiate@active@char{!}
29.173 \initiate@active@char{?}
```

We specify that the French group of shorthands should be used.

```
29.174 \expandafter\addto\csname extras\CurrentOption\endcsname{%
```

29.175 \languageshorthands\{french\}\}

These characters are 'turned on' once, later their definition may vary.

```
29.176 \expandafter\addto\csname extras\CurrentOption\endcsname{%
```

29.177 \bbl@activate\{:\}\bbl@activate\{;\}\%
29.178 \bbl@activate\{!\}\bbl@activate\{?\}\}
29.179 \expandafter $\backslash$ addto $\backslash c s n a m e ~ n o e x t r a s \backslash C u r r e n t O p t i o n \backslash e n d c s n a m e\{\% ~$
$29.180 \backslash$ bbl@deactivate\{:\}\bbl@deactivate\{;\}\%
29.181 \bbl@deactivate\{!\}\bbl@deactivate\{?\}\}

One more thing \extrasfrench needs to do is to make sure that $\backslash$ frenchspacing is in effect. If this is not the case the execution of \noextrasfrench will switch it off again.

```
29.182 \expandafter\addto\csname extras\CurrentOption\endcsname{%
29.183 \bbl@frenchspacing}
29.184 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
29.185 \bbl@nonfrenchspacing}
```

\french@sh@;@ We have to tune the amount of white 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 $\backslash$ FDP@thinspace which will be defined, up to the user's wishes, as an automatic added thin space, or as \@empty.
29.186 \declare@shorthand\{french\}\{;\}\{\%
29.187 \ifhmode
$29.188 \backslash$ ifdim $\backslash l a s t s k i p>\backslash z @$
$29.189 \backslash u n s k i p \backslash p e n a l t y \backslash @ M \backslash$ thinspace
$29.190 \backslash$ else
$29.191 \backslash$ FDP@thinspace
29.192 \fi
29.193 \fi

```
                    Now we can insert a ; character.
29.194 \string;}
\french@sh@!@ Because these definitions are very similar only one is displayed in a way that the \(\backslash f r e n c h @ s h @ ? @\) definition can be easily checked.
29.195 \declare@shorthand\{french\}\{!\}\{\%
29.196 \ifhmode
29.197 \ifdim\lastskip>\z@
29.198 \unskip\penalty\@M\thinspace
29.199 \else
\(29.200 \quad \backslash\) FDP@thinspace
29.201 \fi
29.202 \fi
29.203 \string!\}
29.204 \declare@shorthand\{french\}\{?\}\{\%
29.205 \ifhmode
29.206 \ifdim\lastskip>\z@
29.207 \unskip\penalty\@M\thinspace
29.208 \else
\(29.209 \quad \backslash\) FDP@thinspace
29.210 \fi
29.211 \fi
29.212 \string?\}
\(\backslash f r e n c h @ s h @: @ ~ T h e ~ ' ~: ' ~ r e q u i r e s ~ a ~ n o r m a l ~ s p a c e ~ b e f o r e ~ i t, ~ i n s t e a d ~ o f ~ a ~ t h i n s p a c e . ~\)
29.213 \declare@shorthand\{french\}\{:\}\{\%
29.214 \ifhmode
29.215 \ifdim\lastskip>\z@
\(29.216 \backslash\) unskip \(\backslash\) penalty \(\backslash @ \backslash\) Fcolonspace
\(29.217 \quad \backslash e l s e\)
\(29.218 \quad \backslash\) FDP@colonspace
29.219 \fi
\(29.220 \quad \backslash f i\)
29.221 \string:\}
\(\backslash\) Fcolonspace \(\backslash\) FDP@thinspace and \(\backslash\) FDP@space are defined as unbreakable spaces by
```

$\backslash$ AutoSpaceBeforeFDP $\backslash$ NoAutoSpaceBeforeFDP
$\backslash A u t o S p a c e B e f o r e F D P$ or as \@empty by \NoAutoSpaceBeforeFDP. Default is \AutoSpaceBeforeFDP

```
29.222 \newcommand{\Fcolonspace}{\space}
```

29.222 \newcommand{\Fcolonspace}{\space}
29.223 \def\AutoSpaceBeforeFDP{%
29.223 \def\AutoSpaceBeforeFDP{%
29.224 \def\FDP@thinspace{\penalty\@M\thinspace}%
29.224 \def\FDP@thinspace{\penalty\@M\thinspace}%
29.225<br>def\FDP@colonspace{\penalty\@M\Fcolonspace}}
29.225<br>def\FDP@colonspace{\penalty\@M\Fcolonspace}}
29.226 \def\NoAutoSpaceBeforeFDP{\let\FDP@thinspace\@empty
29.226 \def\NoAutoSpaceBeforeFDP{\let\FDP@thinspace\@empty
29.227 \let\FDP@colonspace\@empty}
29.227 \let\FDP@colonspace\@empty}
29.228 \AutoSpaceBeforeFDP

```
29.228 \AutoSpaceBeforeFDP
```

\system@sh@:@ \system@sh@!@ \system@sh@?@ \system@sh@;@

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.
229 \declare@shorthand\{system\}\{:\}\{\string:\}
29.230 \declare@shorthand\{system\}\{!\}\{\string!\}
29.231 \declare@shorthand\{system\}\{?\}\{\string?\}
29.232 \declare@shorthand\{system\}\{;\}\{\string;\}

### 29.4 French quotation marks

EC/LM fonts and standard PostScript fonts have built-in guillemets, we of course use them. CM fonts have no French guillemets built-in, so we have to emulate them:

- if a file 't1lmr.fd' is found, we expect the LM fonts to be present on the system, so we pick up French guillemets from them; before version 1.6b, French guillemets were borrowed from the 'wncyr' fonts, this has been changed because it resulted in a conflict with the russian option of babel. Former Cyrillic guillemets are still available through the command \CyrillicGuillemets (for backward compatibility).
- otherwise we use math symbols, either $\mathrm{A}^{\mathrm{A}} \mathrm{T}_{\mathrm{E}}$ 's 'lasy' font if available, or $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ symbols \ll and \gg otherwise;

The standard names for French quotation marks are \guillemotleft and \guillemotright, these commands are defined in t1enc.def for T1-encoding. In $\mathrm{FAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ formats, we first define these commands for OT1-encoding (for CM fonts), using either the Polish ' $\mathrm{pl}^{*}$ ' fonts or LATEX's 'lasy' fonts. Other local text encodings should define them too, but if they don't, the OT1 definitions will be used. The standard PostScript fonts should always be used together with T1encoding (not OT1), for proper hyphenation and built-in (better looking) French quotation marks.

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. \guill@spacing which does the spacing, has been fine tuned by Thierry Bouche.

The top macros for quotation marks will be called $\log$ ("ouvrez guillemets") and $\backslash \mathrm{fg}$ ("fermez guillemets").

The top level definitions for French quotation marks are switched on and off through the \extrasfrench \noextrasfrench mechanism. Outside French, \og and $\backslash f g$ will typeset standard English opening and closing double quotes.

As \DeclareTextCommand cannot be used after the \begin\{document\} we in- } troduce internal definitions \begin@guill and \end@guill.

We'll try to be smart to users of D. CARLISLE's xspace package: if this package is loaded there will be no need for $\}$ or $\backslash$ to get a space after $\backslash f g$.
\guillemotleft $\operatorname{In} \operatorname{LAT}_{E} X 2_{\varepsilon}$ we provide a dummy definition for $\backslash o g$ and $\backslash f g$, just to display an $\backslash$ guillemotright error message in case $\backslash \mathrm{og}$ or $\backslash$ fg have been defined elsewhere.
log $_{9.233} \backslash$ newcommand $\{\backslash$ og $\}\{\backslash @$ empty $\}$
$\backslash$ g $_{9.234}$ \newcommand $\{\backslash$ fg $\}\{\backslash$ @empty $\}$
$\backslash$ FrenchGuillemetsFrom
29.235
29.236 $\quad$ ifLaTeXe $\backslash$ FrenchGuillemetsFrom\#1\#2\#3\#4\{\%
\CyrillicGuillemets
\PolishGuillemets9.243 \LasyGuillemetฐ9.244
\bbl@frenchguillemet89.245
\bbl@nonfrenchguillemet39.246
29.247
29.248
29.249
29.250
29.251
29.252
29.253
29.254
29.255
29.256
29.257
29.258
29.259
29.260
29.261
29.262
29.263
29.264
\def \CyrillicGuillemets\{\FrenchGuillemetsFrom\{0T2\}\{wncyr\}\{60\}\{62\}\}
\def $\backslash$ PolishGuillemets\{\FrenchGuillemetsFrom\{T1\}\{lmr\}\{19\}\{20\}\}
\def \LasyGuillemets\{\%
$\backslash$ DeclareTextCommand\{\guillemotleft\}\{OT1\}\{\hbox\{\%
$\backslash$ fontencoding\{U\}\fontfamily\{lasy\}\selectfont (\kern-0.20em(\}\}\%
\DeclareTextCommand\{\guillemotright\}\{0T1\}\{\hbox\{\%
\fontencoding\{U\}\fontfamily\{lasy\}\selectfont) \kern-0.20em) \}\}\}
$\backslash I f F i l e E x i s t s\{t 1 l \mathrm{mr} . f d\}\{\backslash$ PolishGuillemets\}\{\LasyGuillemets\}
\DeclareTextSymbolDefault\{\guillemotleft\}\{0T1\}
\DeclareTextSymbolDefault\{\guillemotright\}\{0T1\}
\def \guill@spacing\{\penalty $\backslash @ \backslash$ hskip. $8 \backslash$ fontdimen2 $\backslash$ font plus. $3 \backslash$ fontdimen3\font minus. $8 \backslash$ fontdimen4\font\}
$\backslash \text { DeclareRobustCommand*\{\begin@guill\}\{\leavevmode }$
\guillemotleft\penalty\@M\guill@spacing\}
$\backslash$ DeclareRobustCommand*\{\end@guill\}\{\ifdim\lastskip>\z@\unskip\fi
$\backslash$ penalty $\backslash @ M \backslash$ guill@spacing $\backslash$ guillemotright $\backslash x$ space $\}$
$\backslash$ AtBeginDocument $\{\backslash i f x \backslash x$ space $\backslash$ @undefined $\backslash l e t \backslash x s p a c e \backslash r e l a x \backslash f i\}$
$\backslash \text { def } \backslash \text { bbl@frenchguillemets\{\renewcommand\{\og\}\{\begin@guill\}\% }$
\{\end@guill\}\}
\def\bbl@nonfrenchguillemets\{\{‘‘\}\%
\{\ifdim\lastskip>\z@\unskip\fi , ,\}\}
For Plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$, and $\mathrm{LA}_{\mathrm{E}} \mathrm{X}-2.09$ we define \begin@guill and \end@guill using math symbols \ll and \gg.
265 \else
29.266
29.267
29.268
29.269
29.270
29.271
29.272

| $\backslash$ def \begin@guill\{\leavevmode\raise0.25ex $\{\leavevmode\raise0.25ex}$ |
| :---: |
| $\backslash$ hbox $\{\$ \backslash$ scriptscriptstyle\ll\$\}\% |
| \penalty $¢ @ M \backslash$ hskip.8\fontdimen2\font |
| plus.3\fontdimen3\font |
| minus.3\fontdimen4\font\} |
| $\backslash d e f \backslash e n d @ g u i l l\{\backslash i f d i m \ l a s t s k i p>\backslash z @ \backslash u n s k i p \backslash p e n a l t y \backslash @ M \backslash f i$ |
| \penalty $¢ @ M \backslash$ hskip.8\fontdimen2\font |

```
29.273 plus.3\fontdimen3\font minus.3\fontdimen4\font
29.274 \raise0.25ex\hbox{$\scriptscriptstyle\gg$}}
29.275 \let\xspace\relax
29.276 \def\bbl@frenchguillemets{\let\og\begin@guill
29.277 \let\fg\end@guill}
29.278\\def\bbl@nonfrenchguillemets{\def\og{``}%
29.279 \def\fg{\ifdim\lastskip>\z@\unskip\fi ',}}
29.280 \fi
29.281 \expandafter\addto\csname extras\CurrentOption\endcsname{%
29.282 \bbl@frenchguillemets}
29.283\expandafter\addto\csname noextras\CurrentOption\endcsname{%
29.284 \bbl@nonfrenchguillemets}
```


### 29.5 French lists

\bbl@frenchlistspacing \bbl@nonfrenchlistspacing

Vertical spacing in general lists should be shorter in French texts than the defaults provided by $L^{A} T_{E} X$. 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 redefine \list (up to version 1.4, the more internal command \@trivlist was redefined, this ensured that all lists would have common settings, but didn't allow the users to provide their own settings, as pointed out by P. Pichaureau). If standard LaTeX settings are preferred, it is easy to issue the command $\backslash$ FrenchListSpacingfalse in the preamble or in frenchb.cfg. Please note that changing the flag $\backslash$ FrenchListSpacing will not take effect immediately, but next time language French is switched on.

The amount of vertical space before and after a list is given by \topsep + $\backslash$ 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 0 pt, but will be noticeable when \parskip is not null.

Of course, this code is only for $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
29.285\newif\ifFrenchListSpacing \FrenchListSpacingtrue
```

29.286 \newif \ifFrenchOldTrivlist
29.287 \ifLaTeX
29.288 \let $\backslash l i s t 0 R I \backslash l i s t$
\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\}. }
29.289 \let\endlistORI\endlist
$29.290 \backslash$ def $\backslash$ FR@listsettings $\{\%$
29.291 \setlength\{- sep\}\{0.4ex plus 0.2ex minus 0.2ex\}\%
29.292 \setlength\{\parsep\}\{0.4ex plus 0.2ex minus 0.2ex\}\%
\(29.293 \backslash\) setlength \(\{\) topsep\}\{0.8ex plus 0.4 ex minus 0.4 ex\} \(\%\)
\(29.294 \backslash\) setlength\{\partopsep\}\{0.4ex plus 0.2ex minus 0.2ex\}\%
\parskip is of type 'skip', its mean value only (not the glue) should be substracted
from \topsep and added to \partopsep, so convert \parskip to a 'dimen' using \@tempdima.


```
29.295 \@tempdima=\parskip
29.296 \addtolength{\topsep}{-\@tempdima}%
29.297 \addtolength{\partopsep}{\@tempdima}}%
29.298\def\listFR#1#2{\listORI{#1}{\FR@listsettings #2}}%
```

Keep the \@trivlist redefinition from version 1.4 (only for compatibility reasons). The flag \ifFrenchOldTrivlist could happen to be useful to process older documents.

```
29.299 \let\@trivlistORI\@trivlist
29.300 \def\@trivlistFR{\FR@listsettings\@trivlistORI}
29.301 \def\bbl@frenchlistspacing{\ifFrenchListSpacing\let\list\listFR\fi
29.302 \ifFrenchOldTrivlist\let\@trivlist\@trivlistFR
2 9 . 3 0 3 ~ \ l e t \ l i s t \ l i s t O R I \ f i \}
29.304 \def\bbl@nonfrenchlistspacing{\let\list\listORI
2 9 . 3 0 5 ~ \ l e t \ @ t r i v l i s t \ @ t r i v l i s t O R I \} ~
29.306 \expandafter\addto\csname extras\CurrentOption\endcsname{%
29.307 \bbl@frenchlistspacing}
29.308 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
29.309 \bbl@nonfrenchlistspacing}
29.310\fi
```

\bbl@frenchitemize \bbl@nonfrenchitemize

The layout of French itemize-lists has changed between version 1.2 and 1.3. Jacques André and Thierry Bouche pointed out to me that vertical spacing between items, before and after the list, should null with no glue added. Moreover horizontal spacing was not correct either: the item labels of a first level list should be vertically aligned on the paragraph's first character (at \parindent from the left margin). Checking the book "Lexique des règles typographiques en usage à l'Imprimerie nationale" confirmed their points, so the itemize environment has been totally redefined for French in version 1.3.

Several people pointed out to me that the layout of lists can either be set by the current language or be tuned independently of the language as part of the layout of the document. So hooks should be provided to switch off the special settings made in frenchb. Setting $\backslash$ FrenchItemizeSpacingfalse after loading frenchb in the preamble switches back to settings of version 1.2 (this can be useful to process old documents). If you want all list spacings to be exactly what they are in standard $\mathrm{ET}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$, then add also $\backslash$ FrenchListSpacingfalse in the preamble after loading frenchb. Both switches can also be set in frenchb.cfg. Please note that changing the flags $\backslash$ FrenchItemizeSpacing and $\backslash$ FrenchListSpacing will not take effect immediately, but next time language French is switched on.

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 defaults to ' - ' and can be changed using 

```
29.313 \newcommand{\FrenchLabelItem}{\textendash}
29.314 \newcommand{\Frlabelitemi}{\FrenchLabelItem}
29.315 \newcommand{\Frlabelitemii}{\FrenchLabelItem}
29.316 \newcommand{\Frlabelitemiii}{\FrenchLabelItem}
29.317 \newcommand{\Frlabelitemiv}{\FrenchLabelItem}
29.318 \def\bbl@frenchitemize{%
29.319 \let\@ltiORI\labelitemi
29.320 \let\@ltiiORI\labelitemii
29.321 \let\@ltiiiORI\labelitemiii
29.322 \let\@ltivORI\labelitemiv
29.323 \let\itemizeORI\itemize
29.324 \let\labelitemi\Frlabelitemi
29.325 \let\labelitemii\Frlabelitemii
29.326 \let\labelitemiii\Frlabelitemiii
29.327 \let\labelitemiv\Frlabelitemiv
29.328 \ifFrenchItemizeSpacing
```

Make sure - ize uses \@trivlistORI even with \FrenchOldTrivlisttrue.
\(29.329 \backslash\) def \(\backslash i t e m i z e\{\backslash l e t \backslash @ t r i v l i s t \backslash @ t r i v l i s t O R I\)
29.330 \ifnum \@itemdepth >\thr@@\@toodeep\else
\(29.331 \backslash\) advance \@itemdepth\@ne
29.332 \edef \@itemitem\{labelitem\romannumeral\the\@itemdepth\}\%
29.333 \expandafter
29.334 \listORI
29.335 \csname\@itemitem\endcsname
29.336 \{\settowidth\{\labelwidth\}\{\csname\@itemitem\endcsname\}\%
29.337 \setlength\{\leftmargin\}\{\labelwidth\}\%
29.338 \addtolength\{\leftmargin\}\{\labelsep\}\%
29.339 \ifnum \@listdepth=0
29.340 \setlength\{
- indent\}\{\parindent\}\%
29.341 \else
29.342 \addtolength\{\leftmargin\}\{\parindent\}\%
29.343 \fi
29.344 \setlength\{
- sep\}\{\z@\}\%
29.345 \setlength\{\parsep\}\{\z@\}\%
29.346 \setlength\{\topsep\}\{\z@\}\%
29.347
\parskip is of type 'skip', its mean value only (not the glue) should be substracted from \topsep and added to \partopsep, so convert \parskip to a 'dimen' using \@tempdima.


| 29.348 | \@tempdima=\parskip |
| :--- | :---: |
| 29.349 | \addtolength\{\topsep\}\{-\@tempdima\}\% |
| 29.350 | \addtolength\{\partopsep\}\{\@tempdima\}\}\% |
| 29.351 | fi\}\% |
| 29.352 | $\backslash$ fi\} |

The user's changes in labelitems are saved when leaving French for further use when switching back to French.


```
29.355 \let\Frlabelitemiii\labelitemiii
29.356
29.357
29.358
2 9 . 3 5 9
29.360
29.361
29.362
29.363
29.364
29.365
29.366\fi
\let\Frlabelitemiv\labelitemiv
\let\labelitemi\@ltiORI
\let\labelitemii\@ltiiORI
\let\labelitemiii\@ltiiiORI
\let\labelitemiv\@ltivORI
\let\itemize\itemizeORI}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
\bbl@frenchitemize}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
    \bbl@nonfrenchitemize}
```


### 29.6 French indentation of sections

\bbl@frenchindent In French the first paragraph of each section should be indented, this is another $\backslash$ bbl@nonfrenchindent difference with US-English. Add this code only in $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$.

```
29.367 \ifLaTeX
29.368 \let\@aifORI\@afterindentfalse
29.369 \def\bbl@frenchindent{\let\@afterindentfalse\@afterindenttrue
                                    \@afterindenttrue}
    \def\bbl@nonfrenchindent{\let\@afterindentfalse\@aifORI
                \@afterindentfalse}
    \expandafter\addto\csname extras\CurrentOption\endcsname{%
        \bbl@frenchindent}
    \expandafter\addto\csname noextras\CurrentOption\endcsname{%
        \bbl@nonfrenchindent}
    \i
```


### 29.7 Formatting footnotes

$\backslash$ AddThinSpaceBeforeFootnotes
$\backslash$ FrenchFootnotes \StandardFootnotes
$\backslash$ AddThinSpaceBeforeFootnotes redefines \@footnotemark (whose definition is saved at the \begin\{document\} in order to include any customization that pack- } ages might have done) to add a thinspace before the number or symbol calling a footnote (any space typed in is removed first). \AddThinSpaceBeforeFootnotes has no effect on the layout of the footnote itself. Note that \thanks in \maketitle also relies on \@footnotemark, so the thinspace is added there too.

```
78 \newif\ifthinspace@FN
29.379 \newif\ifFR@fntlayout
80 \newdimen\parindentFFN
81 \ifLaTeXe
29.382 \newcommand{\AddThinSpaceBeforeFootnotes}{\thinspace@FNtrue}
29.383\AtBeginDocument{\let\@footnotemarkORI\@footnotemark
29.384 \def\@footnotemarkFR{\leavevmode\unskip\unkern
                                    \,\@footnotemarkORI}%
    \ifthinspace@FN
                                \let\@footnotemark\@footnotemarkFR
    \fi}
```

We then define \@makefntextFR, 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 textsuperscripts) 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 defined at the \begin\{document\}, as the maximum of } $\backslash$ parindent and 1.5 em .

```
29.389\def\ftnISsymbol{\@fnsymbol\c@footnote}
29.390 \long\def\@makefntextFR#1{\ifx\thefootnote\ftnISsymbol
                                \@makefntextORI{#1}%
29.391
29.392
2 9 . 3 9 3 ~ \ p a r i n d e n t = \ p a r i n d e n t F F N
2 9 . 3 9 4 ~ \ r u l e \ z @ \ f o o t n o t e s e p ~
29.395 \setbox\@tempboxa\hbox{\@thefnmark}%
29.396 \ifdim\wd\@tempboxa>\z@
\llap{\@thefnmark}.\kern.5em
29.398 \fi #1
29.399 \fi}%
```

We now define two commands \FrenchFootnotes and \StandardFootnotes which select French or Standard layout for footnotes; we have to save the standard definition of \@makefntext at the \begin\{document\}, and then redefine } $\backslash$ @makefntext according to the value of an internal flag set by $\backslash$ FrenchFootnotes and reset by \StandardFootnotes.

```
29.400 \AtBeginDocument{\parindentFFN=\parindent
29.401 \ifdim\parindentFFN<1.5em\parindentFFN=1.5em\fi
29.402 \let\@makefntextORI\@makefntext
29.403 \long\def\@makefntext#1{%
29.404 \ifFR@fntlayout
29.405 \@makefntextFR{#1}%
29.406 \else
\@makefntextORI{#1}%
\fi
                                    }%
    }
    \newcommand{\FrenchFootnotes}{\FR@fntlayouttrue}
    29.412 \newcommand{\StandardFootnotes}{\FR@fntlayoutfalse}
29.413 \fi
```


### 29.8 Formatting numbers

\DecimalMathComma
\StandardMathComma

As mentioned in the $\mathrm{T}_{\mathrm{E}} \mathrm{Xb}$ book p . 134, the comma is of type $\backslash$ 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.

```
29.414 \newcount\std@mcc
29.415 \newcount\dec@mcc
29.416 \std@mcc=\mathcode`\,
29.417\dec@mcc=\std@mcc
29.418 \@tempcnta=\std@mcc
29.419 \divide\@tempcnta by "1000
29.420\multiply\@tempenta by "1000
29.421 \advance\dec@mcc by -\@tempcnta
.422 \newcommand{\DecimalMathComma}{\iflanguage{french}%
23 {\mathcode'\,=\dec@mcc}{}%
\addto\extrasfrench{\mathcode'\,=\dec@mcc}}
\newcommand{\StandardMathComma}{\mathcode'\,=\std@mcc
\addto\extrasfrench{\mathcode'\,=\std@mcc}}
27 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
\mathcode`\,=\std@mcc}
```

In English the decimal part starts with a point and thousands should be separated by a comma: an approximation of $1000 \pi$ should be inputed as $\$ 3\{\} ,141.592\{\} 653 \$,$$ in math mode and as 3,141.592,653$ in text

In French the decimal part starts with a comma and thousands should be separated by a space; the same approximation of $1000 \pi$ should be inputed as $\$ 3 \backslash ; 141\{\} ,592 \backslash ; 653 \$$ in math mode and as something like $3 \sim 141,592^{\sim} 653$ in text: in math mode, 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 (French or non-French).
\nombre accepts an optional argument which happens to be useful with the package 'dcolumn', it specifies the decimal separator used in the source code: $\backslash$ newcolumntype $\{d\}\{D\{\},\{\backslash$ decimalsep $\}\{-1\}\}$
\begin\{tabular\}\{d\}\hline }
3,14 <br>
\nombre[,]\{123,4567\} <br>
\nombre[,]\{9876,543\}<br>\hhline
\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 French ( $\backslash \mathrm{f}$ @thousandsep).

The commands \decimalsep and \thousandsep get default definitions (for the English language) when frenchb is loaded; these definitions will be updated when the current language is switched to or from French.

```
29\mathchardef\m@comma=\dec@mcc
30\def\@comma@{\ifmmode\m@comma\else,\fi}
\def\f@thousandsep{\ifmmode\mskip5.5mu\else\penalty\@M\kern.3em\fi}
\def\ThinSpaceInFrenchNumbers{\def\f@thousandsep{%
\ifmmode\mskip3mu\else\penalty\@M\kern.16667em\fi}}
34 \newcommand{\decimalsep}{.}
\newcommand{\thousandsep}{\@comma@}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
    \def\decimalsep{\@comma@}%
    \def\thousandsep{\f@thousandsep}}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
    \def\decimalsep{.}%
    \def\thousandsep{\@comma@}}
```

Signs can now be entered inside \nombre. When \nombre is used in text-mode, signs should be text symbols to get the series, shape... from the current text-font. When signs are not available in text-mode, we provide some defaults.

```
29.442 \providecommand{\textminus}{\textendash}
29.443 \providecommand{\textplusminus}{\ensuremath{\pm}}
29.444\providecommand{\textminusplus}{\ensuremath{\mp}}
29.445\def\fb@minus{\ifmmode-\else\textminus\fi}
29.446\def\fb@plusminus{\ifmmode\pm\else\textplusminus\fi}
29.447\def\fb@minusplus{\ifmmode\mp\else\textminusplus\fi}
```

The decimal separator used when inputing a number with \nombre has to be a comma. \nombre splits the inputed 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 $\mathrm{HT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ only) the optional argument of $\backslash$ nombre.

```
48\if@Two@E
\newcommand{\nombre}[2][\decimalsep]{\def\@decimalsep{#1}%
29.450 \@@nombre#2\@empty,\@empty,\@nil}
.451 \else
    \def\@decimalsep{\decimalsep}
    \newcommand{\nombre}[1]{\@nombre#1\@empty,\@empty,\@nil}
54 \fi
55\def\@firstofmany#1#2,{#1}
456 \def\@@nombre#1,#2,#3\@nil{%
29.457 \def\nb@sign{}%
29.458 \edef\nb@first{\@firstofmany #1\@empty,}%
29.459 \edef\nb@suite{\@secondoftwo #1\@empty,}%
29.460 \if+\nb@first \def\nb@sign{+}\fi
29.461 \if-\nb@first \def\nb@sign{\fb@minus}\fi
29.462 \expandafter\ifx\nb@first\pm \def\nb@sign{\fb@plusminus}\fi
```

```
29.463 \expandafter\ifx\nb@first\mp \def\nb@sign{\fb@minusplus}\fi
29.464 \ifx\@empty\nb@sign
29.465 \let\@tmp\nb@suite\edef\nb@suite{\nb@first\@tmp}%
29.466
29.467 \nb@sign\expandafter\@nombre\nb@suite#2,#3\@nil}
29.468\def\@nombre#1,#2,#3\@nil{%
29.469 \ifx\@empty#2%
29.470 \@integerpart{#1}%
29.471 \else
29.472 \@integerpart{#1}\@decimalsep\@decimalpart{#2}%
29.473
```

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.
29.474 \def \@decimalpart\#1\{\@@decimalpart\#1\@empty\@empty\@empty\}
29.475 \def \@@decimalpart\#1\#2\#3\#4\{\#1\#2\#3\%
29.476 \ifx\@empty\#4\%
29.477 \else
29.478 \thousandsep $\backslash$ expandafter\@@decimalpart\expandafter\#4\%
29.479 \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.
$29.480 \backslash$ def $\backslash$ @integerpart\#1 \{\@@integerpart\{\}\#1\@empty $\backslash @ e m p t y \backslash @ e m p t y\}$
29.481 \def \@@integerpart\#1\#2\#3\#4\{\%
29.482 \ifx\@empty\#2\%
\@addthousandsep\#1 \relax
\else
\ifx\@empty\#3\%
\@addthousandsep\@empty\@empty\#1\#2\relax
\else
\ifx\@empty\#4\%
\@addthousandsep\@empty\#1\#2\#3\relax
\else
\@@integerpartafterfi\{\#1\#2\#3\#4\}\% $\backslash f i$
\fi
\fi\}
def \@@integerpartafterfi\#1 fi $^{\text {\fi\fi\{\fi\fi\fi\@@integerpart }}$ \#\#1\}\}
\def \@addthousandsep\#1\#2\#3\#4\{\#1\#2\#3\%
\if\#4\relax
$\backslash e l s e$
\thousandsep $\backslash$ expandafter \@addthousandsep\expandafter\#4\%
\fi\}

### 29.9 Dots...

$\mathrm{LAT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ 's standard definition of $\backslash$ dots in text-mode is ··· 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 $\backslash$ Frtextellipsis for French (in $\mathrm{IT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ only).

The $\backslash$ if construction in the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ definition of $\backslash$ dots doesn't allow the use of xspace (xspace is always followed by a $\backslash f i$ ), so we use the AMS-EATEX construction of \dots; this has to be done 'AtBeginDocument' not to be overwritten when amsmath.sty is loaded after babel

## \Frtextellipsis

29.501 \ifLaTeXe
29.502 \DeclareTextCommandDefault\{\Frtextellipsis\}\{\%
29.503 . \kern\fontdimen3\font. \kern\fontdimen3\font. \xspace\}
$\backslash$ 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 $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ definitions 'AtBeginDocument', if amsmath has not been loaded. \Mdots@ doesn't change when switching from/to French, while \Tdots@ is \Frtextellipsis in French and \Tdots@ORI otherwise.
29.504 \newcommand\{\Tdots@ORI\}\{\@xp···\}
29.505 \newcommand\{\Tdots@\}\{\Tdots@ORI\}
29.506 \newcommand\{\Mdots@\}\{\@xp\mdots@\}
$29.507 \backslash$ AtBeginDocument\{\DeclareRobustCommand\{\dots\}\{\relax
29.508 \csname\ifmmode M\else T\fi dots@\endcsname\}\%
$29.509 \quad$ \ifx $\backslash$ @xp $\backslash @ u n d e f i n e d \backslash l e t \backslash @ x p \backslash r e l a x \backslash f i$
29.510
29.511
$29.512 \backslash$ def $\backslash \mathrm{bbl@frenchdots} \mathrm{\{ } \mathrm{\backslash let} \mathrm{\backslash Tdots@} \mathrm{\backslash Frtextellipsis} \mathrm{\}}$
29.513 \def \bbl@nonfrenchdots\{\let\Tdots@\Tdots@ORI\}
29.514 \expandafter\addto\csname extras \CurrentOption\endcsname\{\%
\bbl@frenchdots\}
\expandafter\addto\csname noextras\CurrentOption\endcsname\{\%
\bbl@nonfrenchdots\}
29.518 \fi

### 29.10 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 29.5) and the indentation of the first paragraph of sections (see 29.6)
$\backslash$ FrenchLayout In a multilingual document, a unified look, either the French one or the standard
look. Both of these commands can only be used before the \begin\{document\}. } $\backslash$ StandardLayout can also be of interest for class designers who do not want frenchb to interfere with their global layout choices.

As of version 1.4, \FrenchLayout and \StandardLayout only affect:

- the layout of lists (see 29.5),
- the indentation of the first paragraph of sections (see 29.6).
29.519 \ifLaTeXe
29.520 \newcommand\{\FrenchLayout\}\{\%
29.521 \{\%
$29.522 \quad$ \PackageWarning\{frenchb.ldf\}\%
$29.523 \quad\{\backslash$ protect $\backslash$ StandardLayout $\backslash$ space makes no change $\backslash$ MessageBreak
29.524 ( $\backslash$ protect $\backslash$ FrenchLayout $\backslash$ space has been selected before
29.525 \MessageBreak for the whole document)\}\%
29.526 \}\%
$29.527 \backslash$ AtBeginDocument $\{$ \bbl@frenchitemize $\backslash$ bbl@frenchlistspacing
29.528 \bbl@frenchindent\}\%
29.529 \let\bbl@nonfrenchitemize\relax
$29.530 \backslash$ let $\backslash$ bbl@nonfrenchlistspacing $\backslash r e l a x$
29.531 \let $\backslash$ bbl@nonfrenchindent $\backslash r e l a x\}$
29.532 \newcommand\{\StandardLayout\}\{\%
29.533 \{\%
$29.534 \backslash$ PackageWarning\{frenchb.ldf\}\%
$29.535 \quad\{\backslash$ protect $\backslash$ FrenchLayout $\backslash$ space makes no change $\backslash$ MessageBreak
29.536 ( $\backslash$ protect $\backslash$ StandardLayout $\backslash$ space has been selected before \MessageBreak for the whole document)\}\%
29.537 \}
29.539 \let\@ltiORI\labelitemi
29.540 \let\@ltiiORI\labelitemii
29.541 \let\@ltiiiORI\labelitemiii
29.542 \let\@ltivORI\labelitemiv
29.543 \let- izeORI
- ize
\(29.544 \backslash\) let \(\backslash \mathrm{bbl@frenchitemize} \mathrm{\backslash relax}\)
\(29.545 \quad \backslash l e t \backslash b b l @ f r e n c h l i s t s p a c i n g \backslash r e l a x\)
29.546 \let \bbl@frenchindent \(\backslash\) relax \(\}\)
29.547 \@onlypreamble\FrenchLayout
29.548 \@onlypreamble\StandardLayout
29.549 \fi


### 29.11 Extra utilities

All that is left to do now is to provide the French user with some extra utilities.
\up \up eases the typesetting of superscripts like ' 1 er '. \up relies on \textsuperscript when available (i. e., in $\mathrm{AT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ ).
\up@size The internal macro \up@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.

```
29.550 \ifx\sevenrm\@undefined
29.551 \ifx\@ptsize\@undefined
29.552 \let\up@size\small
29.553 \else
29.554 \ifx\selectfont\@undefined
```

In this case the format is the original $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}-2.09$ :

| 29.555 | \ifcase\@ptsize |
| :---: | :---: |
| 29.556 | \let\up@size\ixpt\or |
| 29.557 | \let\up@size\xpt\or |
| 29.558 | \let\up@size\xipt |
| 29.559 | \fi |

When \selectfont is defined we probably have NFSS available:

```
29.560 \else
    \ifcase\@ptsize
                        \def\up@size{\fontsize\@ixpt{10pt}\selectfont}\or
                                \def\up@size{\fontsize\@xpt{11pt}\selectfont}\or
                                \def\up@size{\fontsize\@xipt{12pt}\selectfont}
                \fi
                \fi
    \i
29.568\else
```

If we end up here it must be a plain based $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ format, so:
29.569 \let\up@size\sevenrm
29.570 \fi

Now we can define $\backslash u p$. When $\mathrm{EAT}_{\mathrm{E}} 2_{\varepsilon}$ runs in compatibility mode ( $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}-2.09$
emulation), \textsuperscript is also defined, but does no good job, so we give two different definitions for \up using \if@Two@E

```
29.571 \if@Two@E
```

29.572 \DeclareRobustCommand*\{\up\}[1]\{\textsuperscript\{\#1\}\}
29.573 \else
$29.574 \backslash$ DeclareRobustCommand*\{\up\}[1]\{\leavevmode\raise1ex $\backslash$ hbox $\{\backslash$ up@size\#1\}\}
29.575 \fi
\ieme is provided for compatibility with francais.sty, the other 5 for compatibility with french.sty:

```
29.576 \def\ieme{\up{\lowercase{e}}\xspace}
```

29.577 \def \iemes\{\up\{\lowercase\{es\}\}\xspace\}
29.578 \def $\backslash i e r\{\backslash u p\{\backslash l o w e r c a s e\{e r\}\} \backslash x s p a c e\}$
29.579 \def \iers\{\up\{\lowercase\{ers\}\}\xspace\}
$29.580 \backslash$ def $\backslash$ iere $\{\backslash u p\{\backslash l o w e r c a s e\{r e\}\} \backslash x s p a c e\}$
29.581 \def $\backslash i e r e s\{\backslash u p\{\backslash l o w e r c a s e\{r e s\}\} \backslash x s p a c e\}$
\No And some more macros for numbering, first two support macros.
\n§9.582 \DeclareRobustCommand*\{\FrenchEnumerate\}[1] \{\%
\prim29.583 \#1 \up\{\lowercase\{o\}\}\kern+.3em\}
\fprimoz9.584 \DeclareRobustCommand*\{\FrenchPopularEnumerate\}[1] \{\%
29.585 \#1\up\{\lowercase\{o\}\}) \kern+.3em\}

Typing \primo should result in ' $1^{\circ}$ ',

```
29.586\def\primo{\FrenchEnumerate1}
```

29.587 \def $\backslash$ secundo\{ $\backslash$ FrenchEnumerate 2$\}$
29.588 \def $\backslash$ tertio $\backslash \backslash$ FrenchEnumerate3\}
29.589 \def \quarto $\{\backslash$ FrenchEnumerate4\}
while typing $\backslash$ fprimo) gives ( 1 ') .
$29.590 \backslash$ def $\backslash$ fprimo) $\{\backslash$ FrenchPopularEnumerate 1$\}$
29.591 \def $\backslash$ fsecundo) $\{\backslash$ FrenchPopularEnumerate2 $\}$
$29.592 \backslash$ def $\backslash$ ftertio) $\{\backslash$ FrenchPopularEnumerate3\}
29.593 \def $\backslash$ fquarto) $\{\backslash$ FrenchPopularEnumerate 4$\}$

Let's provide two macros for the common abbreviations of "Numéro".

```
29.594 \DeclareRobustCommand*{\No}{N\up{\lowercase{o}}\kern+.2em}
```

$29.595 \backslash$ DeclareRobustCommand*\{\no\}\{n\up\{\lowercase\{o\}\}\kern+. 2 em$\}$
$\backslash$ 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; this is a simpler implementation of commands \fsc and \lsc from french.sty: no automatic uppercase/lowercase conversion is performed. Usage: Jean~\bsc\{Duchemin\}.
$29.596 \backslash$ DeclareRobustCommand*\{\bsc\}[1]\{\leavevmode $\backslash$ hbox $\{\backslash$ scshape \#1\}\}
Some definitions for special characters. The first eight are mandatory for \oe etc. to work properly in moving arguments, the others just for convenience. 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 $\backslash$ boi may not be used in math mode, its name in math mode is \backslash.
\degre needs a special treatment: it is \char6 in T1-encoding and \char23 in OT1-encoding, both can be accessed by the command $\backslash r\}$ for ring accent.

## .597 \ifLaTeXe

29.598 \DeclareTextSymbol\{\ae\}\{T1\}\{230\}
29.599 \DeclareTextSymbol\{\ae\}\{0T1\}\{26\}
29.600 \DeclareTextSymbol\{\oe\}\{T1\}\{247\}
29.601 \DeclareTextSymbol\{\oe\}\{OT1\}\{27\}
29.602 \DeclareTextSymbol\{\AE\}\{T1\}\{198\}
29.603 \DeclareTextSymbol\{\AE\}\{OT1\}\{29\}
29.604 \DeclareTextSymbol\{\OE\}\{T1\}\{215\}
29.605 \DeclareTextSymbol\{\OE\}\{OT1\}\{30\}
29.606 \DeclareTextSymbol\{\at\}\{T1\}\{64\}
29.607 \DeclareTextSymbol\{\at\}\{OT1\}\{64\}
29.608 \DeclareTextSymbol\{\circonflexe\}\{T1\}\{94\}
29.609 \DeclareTextSymbol\{\circonflexe\}\{0T1\}\{94\}
29.610 \DeclareTextSymbol\{\tild\}\{T1\}\{126\}
29.611 \DeclareTextSymbol\{\tild\}\{OT1\}\{126\}
29.612 \DeclareTextSymbolDefault\{\at\}\{0T1\}
29.613 \DeclareTextSymbolDefault\{\circonflexe\}\{OT1\}
29.614 \DeclareTextSymbolDefault\{\tild\}\{0T1\}

```
29.615 \DeclareRobustCommand*{\boi}{\textbackslash}
29.616 \DeclareRobustCommand*{\degre}{\r{}}
29.617 \else
29.618 \def\T@one{T1}
29.619 \ifx\f@encoding\T@one
29.620 \newcommand{\degre}{\char6}
29.621 \else
29.622 \newcommand{\degre}{\char23}
29.623 \fi
29.624 \newcommand{\at}{\char64}
29.625 \newcommand{\circonflexe}{\char94}
29.626 \newcommand{\tild}{\char126}
29.627 \newcommand{\boi}{{$\backslash$}}
29.628\fi
```

\degres We now define a macro \degres 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 \degres to 0.3 em , this lets the symbol 'degree' stick to the preceding (e.g., $45 \backslash$ degres ) or following character (e.g., 20~ $\backslash$ degres C ).

If the $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ Companion fonts are available (textcomp.sty), we pick up \textdegree from them instead of using emulating ‘degrees' from the $\backslash r\}$ accent. Otherwise we overwrite the (poor) definition of \textdegree given in latin1.def applemac.def etc. (called by inputenc.sty) by our definition of \degres. We also advice the user (once only) to use TS1-encoding.

```
ifLaTeXe
    \def\Warning@degree@TSone{%
        \PackageWarning{frenchb.ldf}{%
            Degrees would look better in TS1-encoding:
            \MessageBreak add \protect
            \usepackage{textcomp} to the preamble.
            \MessageBreak Degrees used}}
    \AtBeginDocument{\expandafter\ifx\csname M@TS1\endcsname\relax
                \DeclareRobustCommand*{\degres}{%
                            \leavevmode\hbox to 0.3em{\hss\degre\hss}%
                    \Warning@degree@TSone
                            \global\let\Warning@degree@TSone\relax}%
                            let\textdegree\degres
                                \else
                                \DeclareRobustCommand*{\degres}{%
                            \hbox{\UseTextSymbol{TS1}{\textdegree}}}%
                            \fi}
\else
    \DeclareRobustCommand*{\degres}{%
        \leavevmode\hbox to 0.3em{\hss\degre\hss}}
```

29.649 \fi

The following macros are used in the redefinition of $\backslash^{\wedge}$ and $\backslash "$ to handle the letter i: they allow users to type simply $\backslash$ ~i and $\backslash$ "i instead of $\backslash \wedge\{\backslash i\}$ and $\backslash "\{\backslash i\}$.

MlTEX's macros dealing with accents conflict with those of $\operatorname{LAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$, so we check whether \csubinverse is defined or not. If \csubinverse is defined, we are in $\mathrm{MlT}_{\mathrm{E}} \mathrm{X}$.

```
29.650 \ifLaTeXe
29.651 \AtBeginDocument{%
29.652 \ifx\csubinverse\@undefined
29.653 \DeclareTextCompositeCommand{\~}{OT1}{i}{\^\i}%
29.654 \DeclareTextCompositeCommand{\"}{OT1}{i}{\"\i}%
29.655 \fi}
29.656 \fi
```


### 29.12 Date and clean up

\datefrench The macro \datefrench redefines the command \today to produce French dates.

```
29.657\@namedef{date\CurrentOption}{%
29.658 \def\today{\number\day
29.659 \ifnum1=\day {\ier}\fi
29.660 \space \ifcase\month
29.661 \or janvier\or f\'evrier\or mars\or avril\or mai\or juin\or
29.662 juillet\or ao\^ut\or septembre\or octobre\or novembre\or
29.663 d\'ecembre\fi
29.664 \space \number\year}}
```

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 $\backslash$ CurrentOption cannot be used: we first load 'frenchb.cfg' (in LaTeX2e only), then call \ldf@quit $\backslash$ CurrentOption. The macrospace used by some control sequences we do not need any longer, is freed.

```
29.665 \ifLaTeXe
29.666 \loadlocalcfg{frenchb}
29.667 \fi
29.668 \let\T@one\relax
29.669 \let\@FI@\relax
29.670 \let\ifLaTeX\@undefined
29.671 \let\LaTeXtrue\@undefined
29.672\let\LaTeXfalse\@undefined
29.673 \let\ifLaTeXe\@undefined
29.674 \let\LaTeXetrue\@undefined
29.675 \let\LaTeXefalse\@undefined
29.676 \ldf@quit\CurrentOption
29.677 </code\
```


## 30 The Italian language

The file italian.dtx ${ }^{26}$ 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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ format was built; in case it was not, read the information coming with your distribution of the $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\backslash$ finalhyphendemerits is set to such a high value that hyphenation is prohibited between the last two lines of a paragraph.
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 and 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 " $\mid$ 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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ distribution.

[^21]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.

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 30.3 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 char-
    acters represented with control sequences or accented
    ones.
"" inserts open quotes ".
"< inserts open guillemots.
"> inserts closed guillemots.
"/ equivalent to \slash
```

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.

```
30.1 <*code>
30.2 \LdfInit{italian}{captionsitalian}%
```

When this file is read as an option, i.e. by the command,italianwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@italiantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
30.3\ifx\l@italian\@undefined
30.4 \@nopatterns{Italian}%
30.5 \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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
30.6 \addto\captionsitalian{%
30.7 \def\prefacename{Prefazione}%
30.8 \def\refname{Riferimenti bibliografici}%
```

```
                    30.9 \def\abstractname{Sommario}%
\def\bibname{Bibliografia}%
\def\chaptername{Capitolo}%
\def\appendixname{Appendice}%
\def\contentsname{Indice}%
\def\listfigurename{Elenco delle figure}%
\def\listtablename{Elenco delle tabelle}%
\def\indexname{Indice analitico}%
\def\figurename{Figura}%
\def\tablename{Tabella}%
\def\partname{Parte}%
\def\enclname{Allegati}%
\def\ccname{e~p.~c.}%
\def\headtoname{Per}%
\def\pagename{Pag.}% % in Italian the abbreviation is preferred
\def\seename{vedi}%
\def\alsoname{vedi anche}%
\def\proofname{Dimostrazione}%
\def \glossaryname{Glossario}%
}%
```

\dateitalian The macro \dateitalian redefines the command \today to produce Italian dates.

```
30.29\def\dateitalian{%
30.30 \def\today{\number\day~\ifcase\month\or
30.31 gennaio\or febbraio\or marzo\or aprile\or maggio\or giugno\or
30.32 luglio\or agosto\or settembre\or ottobre\or novembre\or
30.33 dicembre\fi\space \number\year}}%
```

\italianhyphenmins The italian hyphenation patterns can be used with both \lefthyphenmin and $\backslash$ righthyphenmin set to 2 .
30.34 \providehyphenmins\{\CurrentOption\}\{\tw@\tw@\}
\extrasitalian Lower the chance that clubs or widows occur.
\noextrasitalian 30.35 \addto\extrasitalian\{\%
30.36 \babel@savevariable\clubpenalty
30.37 \babel@savevariable\widowpenalty
30.38 \clubpenalty3000\widowpenalty3000\}\%

Never ever break a word between the last two lines of a paragraph in italian texts.
30.39 \addto\extrasitalian\{\%
30.40 \babel@savevariable\finalhyphendemerits
30.41 \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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ finds the following hyphenation points nel-1'al-tez-za instead of none.
30.42 \addto\extrasitalian\{\%
30.43 \lccode''='' $\} \%$

### 30.1 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 " $\mid$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ command $\backslash$-.

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 ${ }^{\sim}{ }^{\sim} \mathrm{W}$. 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 ${ }^{\sim} \mathrm{W}$, 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 ".

[^22]\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.

```
30.48\declare@shorthand{italian}{"}{%
30.49 \ifmmode
30.50 \def\it@next{',}%
30.51 \else
30.52 \def\it@next{\futurelet\it@temp\it@cwm}%
30.53\fi
30.54 \it@next
30.55 }%
```

\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 $I_{12}$ : if so a compound word separator is inserted and the $I$ 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.

```
\(30.56 \backslash\) def \(\backslash i t @ @ c w m\{\backslash n o b r e a k \backslash d i s c r e t i o n a r y\{-\}\}\} \backslash h s k i p \backslash z @ s k i p\} \%\)
\(30.57 \backslash d e f \backslash i t @ @ o c a p \# 1\{\backslash i t @ o c a p\} \backslash d e f \backslash i t @ @ c c a p \# 1\{\backslash i t @ c c a p\} \%\)
\DeclareRobustCommand*\{\it@cwm\}\{\let\it@@next\relax
\ifcat\noexpand\it@temp a\%
\def\it@@next\{\it@@cwm\}\%
30.61 \else
30.62 \if \(\backslash\) noexpand \(\backslash i t @ t e m p ~ \ s t r i n g l \%\)
30.63
30.64
\if \noexpand \it@temp \string<\%
30.66 \def \(\backslash i t @ @ n e x t\{\backslash i t @ @ o c a p\} \%\)
30.67 \else
30.68 \if \(\backslash\) noexpand \(\backslash i t @ t e m p ~ \ s t r i n g>\%\)
30.69 \def\it@@next\{\it@@ccap\}\%
\(30.70 \backslash e l s e\)
30.71 \if \noexpand \(\backslash i t @ t e m p \backslash s t r i n g / \%\)
30.72 \def \(\backslash i t @ n e x t\{\backslash\) slash\@gobble\}\%
30.73 \else
30.74 \ifx\it@temp"\%
30.75
\def\it@@next\{‘‘\@gobble\}\%
```

```
30.76 \fi
30.77 \fi
3.78 \fi
30.79 \fi
30.80 \fi
30.81 \fi
30.82\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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ 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 " $\mid$ may be used that performs exactly as " normally does, except that the $\mid$ sign is removed from the token input list: kilo"|\{\"o\}rsted gets hyphenated as ki-lo-ör-sted.

### 30.2 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 $V_{\mathrm{rms}}$ 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 $\mathbf{A}^{\mathrm{T}}$ 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 $1^{\mathrm{a}}$ or the old fashioned obsolete abbreviation $\mathrm{F}^{\mathrm{lli}}$ 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 30.83 \@ifpackageloaded{units}{}{%
    30.84 \DeclareRobustCommand*{\bbl@unit}[1]{%
        \textormath{\,\mbox{#1}}{\,\mathrm{#1}}}%
    }%
\DeclareRobustCommand*{\bbl@ap}[1]{%
    \textormath{\textsuperscript{#1}}{~{\mathrm{#1}}}}%
\DeclareRobustCommand*{\bbl@ped} [1] {%
    \textormath{$_{\mbox{\fontsize\sf@size\z@
    30.91 \selectfont#1}}$}{_\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.

```
0.92 \@ifpackageloaded{units}{}{%
0.93 \addto\extrasitalian{%
    \babel@save\unit\let\unit\bbl@unit}%
    }%
\addto\extrasitalian{%
    \babel@save\ap\let\ap\bbl@ap
    \babel@save\ped\let\ped\bbl@ped
    }%
```


### 30.3 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 ~ (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 $T_{E X} X$ sequences
when writing a file to disk (for the sake of file portability), and to transform the standard $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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...

### 30.4 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 $\mathrm{L}_{\mathrm{E}} \mathrm{X} \mathrm{X} 2_{\varepsilon}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $0 T 2 \mathrm{cmr} . f \mathrm{fd}$ font description file. Actually the presence of this file does not guarantee the completeness of your $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ implementation; should $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ 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 id 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 \it@ccap by means of the $\mathrm{I}^{\mathrm{A}} \mathrm{T}_{\mathrm{E}} \mathrm{X}$ symbols; each one of these macro sets actually redefines the control sequences \it@ocap and \it@ccap that are the ones effectively activated by the shortcuts "< and ">.
30.100 \def $\backslash$ LtxSymbCaporali $\{\%$
30.101 \DeclareRobustCommand*\{\it@ocap\}\{\mbox\{\%
$30.102 \backslash$ fontencoding\{U\}\fontfamily\{lasy\}\selectfont (\kern-0.20em (\}\%
30.103 \ignorespaces\}\%
$30.104 \backslash$ DeclareRobustCommand*\{\it@ccap\}\{\ifdim\lastskip>\z@\unskip\fi
30.105 \mbox\{\%
30.106
\fontencoding\{U\}\fontfamily\{lasy\}\selectfont)\kern-0.20em)\}\}\%
30.107 \}\%

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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ special symbols is used. In any case the user can force the use of the Cyrillic guillemets substitution by issuing the declaration $\backslash$ CyrillicCaporali just before the \begin\{document\} statement; in alternative } the user can specify with
$\backslash$ CaporaliFrom $\{\langle$ encoding $\rangle\}\{\langle$ family $\rangle\}\{\langle$ opening number $\rangle\}\{\langle$ closing number $\rangle\}$
the encoding and family of the font $\mathrm{s} / \mathrm{he}$ 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 di pdflatex, for example.

```
30.108\def\CaporaliFrom#1#2#3#4{%
```

30.111
30.112
30.113
30.11
30.11
30.116
30.117
30.109 \DeclareFontEncoding\{\#1\}\{\}\{\}\%
30.110 \DeclareTextCommand\{\it@ocap\}\{T1\}\{\%
\DeclareFontEncoding\{\#1\}\{\}\{\}\%
\DeclareTextCommand\{\it@ocap\}\{T1\}\{\% $\{\backslash$ fontencoding\{\#1\}\fontfamily\{\#2\}\selectfont\char\#3\ignorespaces\}\}\%
\DeclareTextCommand\{\it@ccap\}\{T1\}\{\ifdim\lastskip>\z@\unskip\fi\% $\left\{\backslash\right.$ fontencoding\{\#1\} ${ }^{\text {fontfamily\{\#2\} }}$ \selectfont $\backslash$ char\#4\}\}\%
\DeclareTextCommand\{\it@ocap\}\{0T1\}\{\% $\{\backslash$ fontencoding\{\#1\}\fontfamily\{\#2\}\selectfont\char\#3\ignorespaces\}\}\%
\DeclareTextCommand\{\it@ccap\}\{0T1\}\{\ifdim\lastskip>\z@\unskip\fi\% $\{\backslash$ fontencoding\{\#1\}\fontfamily\{\#2\}\selectfont $\backslash$ 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.

```
30.118\def\get@ae#1#2#3!{\def\bbl@ae{#1#2}}%
30.119\def\@ifT@one@noCap{\expandafter\get@ae\f@family!%
30.120\def\bbl@temp{ae}\ifx\bbl@ae\bbl@temp\expandafter\@firstoftwo\else
30.121 \expandafter\@secondoftwo\fi}%
```

We set another couple of boolean variables for testing the existence of the CBgreek or the Cyrillic fonts
30.122 \newif \if@CBgreekEncKnown
30.123 \IfFileExists\{lgrcmr.fd\}\%
$30.124 \quad\{\backslash @ C B g r e e k E n c K n o w n t r u e\}\{\backslash @ C B g r e e k E n c K n o w n f a l s e\}$
30.125 \newif \if@CyrEncKnown
30.126 \IfFileExists\{ot2cmr.fd\}\%
$30.127 \quad\{\backslash @ C y r E n c K n o w n t r u e\}\{\backslash @ C y r E n c K n o w n f a l s e\} \%$
\CBgreekCaporali Next we define the macros \CBgreekCaporali, \T@unoCaporali, and \CyrillicCaporali; $\backslash$ CyrillicCaporali with the latter one we test the loaded class, and if it's slides nothing gets done. In \T@unoCaporali any case each one of these declarations, if used, must be specified in the preamble.

```
30.128\def\CBgreekCaporali{\@ifclassloaded{slides}{%
30.130 \DeclareRobustCommand*{\it@ccap}%
30.131 {\ifdim\lastskip>\z@\unskip\fi
30.132 {\fontencoding{LGR}\selectfont))}}%
30.133 \DeclareRobustCommand*{\it@ocap}%
                                    {{\fontencoding{LGR}\selectfont((}\ignorespaces}}%
    {\LtxSymbCaporali}}%
    {\DeclareFontEncoding{LGR}{}{}%
    \DeclareRobustCommand*{\it@ccap}%
    {\ifdim\lastskip>\z@\unskip
    \fi{\fontencoding{LGR}\selectfont))}}%
        \DeclareRobustCommand*{\it@ocap}%
    {{\fontencoding{LGR}\selectfont((}\ignorespaces}}%
        }%
    \def\CyrillicCaporali{\@ifclassloaded{slides}{\relax}%
    {\DeclareFontEncoding{0T2}{}{}%
    \DeclareRobustCommand*{\it@ccap}%
                            {\ifdim\lastskip>\z@\unskip\fi
                            {\fontencoding{0T2}\selectfont\char62\relax}}%
            \DeclareRobustCommand*{\it@ocap}%
                            {{\fontencoding{0T2}\selectfont\char60\relax}\ignorespaces}}}%
\@onlypreamble{\CBgreekCaporali}\@onlypreamble{\CyrillicCaporali}%
def\T@unoCaporali{\DeclareRobustCommand*{\it@ocap}{<<\ignorespaces}%
                        \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
30.153\ifx\cf@encoding\bbl@t@one
30.154 \@ifT@one@noCap{%
30.155 \if@CBgreekEncKnown
30.156 \CBgreekCaporali
30.157 \else
30.158 \if@CyrEncKnown
30.159 \CyrilicCaporali
30.160 \else
3 0 . 1 6 1 ~ \ L t x S y m b C a p o r a l i ~
30.162 \fi
30.163 \fi}%
30.164 {\T@unoCaporali}%
But if the default encoding is not the T1 one, then the substitutions must be
performed.
30.165\else
30.166 \if@CBgreekEncKnown
30.167 \CBgreekCaporali
30.168 \else
30.169 \if@CyrEncKnown
3 0 . 1 7 0 ~ \ C y r i l i c C a p o r a l i ~
30.171 \else
\LtxSymbCaporali
    \fi
\i
30.175 \fi
```


### 30.5 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.
30.176 \ldf@finish\{italian\}\%
30.177 〈/code〉

## 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).

## 31 The Latin language

The file latin.dtx ${ }^{27}$ defines all the language-specific macros for the Latin language both in modern and medieval spelling.

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 32, 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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$, 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 ${ }^{A} T_{E} X$ format by 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.

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\{latin\}\{captionslatin\}

When this file is read as an option, i.e. by the command,latinwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\l@latintoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

[^23]${ }^{\text {i }}$ i inserts the breve accent as 1 ; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures $æ$ and œ.
$=\mathrm{a}$ inserts the macron accent as $\overline{\mathrm{a}}$; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures æ and œ.
" inserts a compound word mark where hyphenation is legal; the next character must not be a medieval ligature $æ$ or $\propto$, nor an accented letter (foreign names).
" | 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 ${ }^{\text {- }}$ and $=$ are active only when the language attribute withprosodicmarks has been declared, otherwise they are disabled; see section 32 for more details.

```
\ifx\l@latin\@undefined
    \@nopatterns{Latin}
    \adddialect\l@latin0\fi
```

Now we declare the medieval language attribute.

```
\bbl@declare@ttribute{latin}{medieval}{%
    \addto\captionslatin{\def\prefacename{Pr{\ae}fatio}}%
    \def\november{Nouembris}%
    \expandafter\addto\expandafter\extraslatin
    \expandafter{\extrasmedievallatin}%
    }
```

The third typesetting style withprosodicmarks is defined here

```
1.12\bbl@declare@ttribute{latin}{withprosodicmarks}{%
```

31.13 \expandafter\addto\expandafter\extraslatin
31.14 \expandafter\{\extraswithprosodicmarks\}\%
31.15 \}

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 language ${ }^{28}$.
\captionslatin The macro \captionslatin defines all strings used in the four standard document classes provided with $\mathrm{L}^{\mathrm{A}} \mathrm{TEX}_{\mathrm{E}}$.

```
1.16\@namedef{captionslatin}{%
31.17\def\prefacename{Praefatio}%
31.18\def\refname{Conspectus librorum}%
31.19\def\abstractname{Summarium}%
31.20\def\bibname{Conspectus librorum}%
31.21 \def\chaptername{Caput}%
```

[^24]```
31.22\\def\appendixname{Additamentum}%
31.23\def\contentsname{Index}%
31.24\def\listfigurename{Conspectus descriptionum}%
31.25 \def\listtablename{Conspectus tabularum}%
31.26 \def\indexname{Index rerum notabilium}%
1.27 \def\figurename{Descriptio}%
1.28\def\tablename{Tabula}%
31.29\def\partname{Pars}%
31.30 \def\enclname{Adduntur}% Or " Additur" ? Or simply Add.?
1.31 \def\ccname{Exemplar}% Use the recipient's dative
31.32 \def\headtoname{\ignorespaces}% Use the recipient's dative
31.33\def\pagename{Charta}%
31.34\def\seename{cfr.}%
31.35 \def\alsoname{cfr.}% R.Tabacco never saw "cfr. atque" or similar forms
31.36 \def\proofname{Demonstratio}%
31.37\def\glossaryname{Glossarium}%
31.38 }
```

In the above definitions there are some points that might change in the future or that require a minimum of attention from the typesetter.

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 $\backslash$ 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 Nouembris. This is taken care of by using a control sequence which can be redefined when the attribute 'medieval' is selected.
```
31.39 \def\datelatin{%
31.40\def\november{Novembris}%
31.41 \def\today{%
31.42 {\check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont
31.43\uppercase\expandafter{\romannumeral\day}}\nobreakspace
```

```
31.44 \ifcase\month\or
31.45 Ianuarii\or Februarii\or Martii\or Aprilis\or Maii\or Iunii\or
31.46 Iulii\or Augusti\or Septembris\or Octobris\or \november\or
31.47 Decembris\fi
31.48 \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 $a b$ 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 $\backslash$ righthyphenmin set to 2.
31.49 \providehyphenmins\{\CurrentOption\}\{\tw@\tw@\}
\extraslatin For this language the \clubpenalty, \widowpenalty are set to rather high val\noextraslatin ues and \finalhyphendemerits is set to such a high value that hyphenation is prohibited between the last two lines of a paragraph.

```
31.50 \addto\extraslatin\{\%
31.51 \babel@savevariable\clubpenalty
31.52 \babel@savevariable\widowpenalty
31.53 \clubpenalty 3000 \widowpenalty3000\}
```

Never ever break a word between the last two lines of a paragraph in latin texts.
31.54 \addto\extraslatin\{\%
$31.55 \backslash$ babel@savevariable\finalhyphendemerits
31.56 \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.
31.57 \addto \extrasmedievallatin\{\%
31.58 \babel@savevariable\{\lccode'\V\}\%
31.59 \babel@savevariable\{\uccode‘\u\}\%
31.60 \lccode' $\backslash \mathrm{V}={ }^{\text {' }}$ \u $\backslash \mathrm{uccode}{ }^{\left.\text {' } \backslash u={ }^{\prime} \backslash \mathrm{V}\right\}}$
$\backslash$ SetLatinLigatures We need also the lccodes for æ and œ; since they occupy different positions in the OT1 $\mathrm{T}_{\mathrm{E}}$-fontencoding compared to the T 1 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，as the uncial ＇$u$＇shape of the lower case＇$v$＇and the above ligatures were used until the middle of the XVII century．Since my opinion（CB）may be wrong，I managed to set up the instruments and it is up to the typesetter to use them or not．

```
31.61 \addto\extrasmedievallatin\{\%
    \babel@savevariable\{\lccode‘\~^e6\}\% T1 \ae
    \babel@savevariable\{\uccode‘\~^e6\}\% T1 \ae
    \babel@savevariable\{\lccode‘\~~c6\}\% T1 \AE
    \babel@savevariable\{\lccode‘\~~f7\}\% T1 \oe
    \babel@savevariable\{\uccode‘\へ~f7\}\% T1 \OE
    \babel@savevariable\{\lccode‘\~~d7\}\% T1 \OE
    \babel@savevariable\{\lccode‘\~~1a\}\% OT1 \ae
    \babel@savevariable\{\uccode‘\~~1a\}\% OT1 \ae
    \babel@savevariable\{\lccode‘\へ~1d\}\% OT1 \AE
    \babel@savevariable\{\lccode‘\~~1b\}\% OT1 \oe
    \babel@savevariable\{\uccode'\へ~1b\}\% OT1 \OE
    \babel@savevariable\{\lccode‘\へ~1e\}\% OT1 \OE
    \SetLatinLigatures\}
\(\backslash\) providecommand \(\backslash\) SetLatinLigatures \(\{\%\)
    \def \@tempA\{T1\}
```



```
        \catcode‘\\~~c6=11 \lccode"\\~~c6="\\~~e6 \% \AE
        \catcode‘\^^f7=11 \lccode‘\へ~f7=‘\\~~f7 \uccode‘\へ~f7=‘\\~~d7 \% \oe
        \catcode‘\\~~d7=11 \lccode‘\\~~d7=‘\\~~f7 \% \OE
    \(\backslash e l s e\)
            \catcode‘\\~~1a=11 \lccode"\へ~1a="\\~~1a \uccode"\へ~1a="\\~~1d \% \ae
            \catcode‘\\~~1d=11 \lccode‘\\~~1d=‘\\~~1a \% \AE (~~])
            \catcode‘\\~~1b=11 \lccode"\\~~1b="\\~~1b \uccode"\へ~1b=‘\\~~1e \% \oe
            \catcode‘\^~1e=11 \lccode‘\\~~1e=‘\\~~1b \% \OE (~~~)
    \fi
    \let\@tempA\@undefined
    \}
```

With the above definitions we are sure that \MakeUppercase works properly and \MakeUppercase\｛C\｛\ae\}sar\} correctly 'yields 'CÆSAR"; correspondingly $\backslash$ MakeUppercase\｛Heluetia\} correctly yields "HELVETIA".

## 32 Latin shortcuts

For writing dictionaries or didactic texts we defined a third language attribute，or a third typesetting style，where 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 the necessary code for allowing hyphenation in the rest of the word．

It must be remarked that the active characters ~ 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 may be switched on and off at will, and in particular that they are off by default if the attribute withprosodicmarks has not been set.

| $\backslash$ ProsodicMarksOn \ProsodicMarksOff | We begin by adding to the normal typesetting style the definitions of the commands \ProsodicMarksOn and \ProsodicMarksOff that should produce messages when the third style is not declared: |
| :---: | :---: |
|  | 32.1 \addto\extraslatin\{\def $\backslash$ ProsodicMarksOn\{\% |
|  | 32.2 \GenericError\{(latin)\@spaces\@spaces\@spaces\@spaces\}\% |
|  | 32.3 \{Latin language error: \string $\backslash$ ProsodicMarksOn\space |
|  | 32.4 is defined by setting the $\backslash$ MessageBreak |
|  | 32.5 language attribute to 'withprosodicmarks' ${ }^{\text {M }}$ MessageBreak |
|  | 32.6 If you continue you are likely to encounter $\backslash$ MessageBreak |
|  | 32.7 fatal errors that I can't recover\}\% |
|  | 32.8 \{See the Latin language description in the babel |
|  | 32.9 documentation for explanation\}\{\@ehd\}\}\} |
|  | 2.10 \addto\extraslatin\{\let $\backslash$ ProsodicMarksOff $\backslash$ relax\} |

Then we temporarily set the caret and the equals sign to active characters so that they can receive their definitions:

```
32.11 \catcode'\= \active
32.12 \catcode`\^ \active
```

and we add the necessary declarations to the macros that are being activated when the Latin language and its typesetting styles are declared:

```
32.13\addto\extraslatin{\languageshorthands{latin}}%
32.14 \addto\extraswithprosodicmarks{\bbl@activate{^}}%
32.15 \addto\extraswithprosodicmarks{\bbl@activate{=}}%
32.16 \addto\noextraswithprosodicmarks{\bbl@deactivate{^}}%
32.17 \addto\noextraswithprosodicmarks{\bbl@deactivate{=}}%
32.18 \addto\extraswithprosodicmarks{\ProsodicMarks}
```

$\backslash$ ProsodicMarks Next we define the defining macro for the active characters

```
32.19 \def\ProsodicMarks{%
32.20\def\ProsodicMarksOn{\catcode'\^ 13\catcode'\= 13\relax}%
32.21 \def\ProsodicMarksOff{\catcode'\^ 7\catcode'\= 12\relax}%
```

Notice that with the above redefinitions of the commands $\backslash$ ProsodicMarksOn and $\backslash$ 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 the switch them off within the environment, being conscious that their deactivation remains local to the environment.

```
32.22 \initiate@active@char{^}%
32.23 \initiate@active@char{=}%
32.24 \declare@shorthand{latin}{`a}{%
32.25 \textormath{\u{a}\allowhyphens}{\hat{a}}}%
32.26 \declare@shorthand{latin}{^e}{%
32.27 \textormath{\u{e}\allowhyphens}{\hat{e}}}%
32.28\declare@shorthand{latin}{^i}{%
32.29 \textormath{\u{\i}\allowhyphens}{\hat{\imath}}}%
32.30\declare@shorthand{latin}{^o}{%
32.31 \textormath{\u{o}\allowhyphens}{\hat{o}}}%
32.32 \declare@shorthand{latin}{^u}{%
32.33 \textormath{\u{u}\allowhyphens}{\hat{u}}}%
32.34 \declare@shorthand{latin}{=a}{%
32.35\textormath{\={a}\allowhyphens}{\bar{a}}}%
32.36 \declare@shorthand{latin}{=e}{%
32.37 \textormath{\={e}\allowhyphens}{\bar{e}}}%
32.38 \declare@shorthand{latin}{=i}{%
32.39 \textormath{\={\i}\allowhyphens}{\bar{\imath}}}%
32.40\declare@shorthand{latin}{=o}{%
32.41 \textormath{\={o}\allowhyphens}{\\operatorname{bar{o}}}%}
32.42 \declare@shorthand{latin}{=u}{%
32.43\textormath{\={u}\allowhyphens}{\\bar{u}}}%
32.44}
```

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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ commands such as $\backslash \mathrm{u}\{\mathrm{I}\}$ for typesetting I , or $\backslash=\{\mathrm{A}\}$ for typesetting $\overline{\mathrm{A}}$.

Finally we restore the caret and equals sign initial default category codes.

```
32.45 \catcode`\= 12\relax
32.46 \catcode`\^ 7\relax
```

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.

## 33 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.

```
33.1 \initiate@active@char{"}%
33.2 \addto\extraslatin{\bbl@activate{"}%
33.3}
```

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.

```
\declare@shorthand{latin}{"}{%
    \ifmmode
        \def\lt@@next{''}%
    \else
        \def\lt@@next{\futurelet\lt@temp\lt@cwm}%
    \fi
    \lt@@next
3.11 }%
```

In text mode the \lt@next control sequence is such that upon its execution a temporary variable \lı@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 $\backslash l t @ c w m$ 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 I, in which case it is gobbled and a discretionary break is introduced.

```
33.12\def\lt@allowhyphens{\nobreak\discretionary{-}{}{}\hskip\z@skip}
\newcommand*{\lt@cwm}{\let\lt@n@xt\relax
    \ifcat\noexpand\lt@temp a%
        \let\lt@n@xt\lt@allowhyphens
    \else
            \if\noexpand\lt@temp\stringl%
                \def\lt@n@xt{\lt@allowhyphens\@gobble}%
            \i
    \fi
    \lt@n@xt}%
```

Attention: the category code comparison does not work if the temporary control sequence \lit@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 " I 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 ${ }^{29}$ 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 $\backslash$ begin\{document\} and resetting the category code of © to its original value.

[^25]$33.22 \backslash l d f @ f i n i s h\{l a t i n\}$
33.23 〈/code〉

## 34 The Portuguese language

The file portuges.dtx ${ }^{30}$ defines all the language-specific macros for the Portuguese language as well as for the Brasilian 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 $\ll$ ).
" $>$ for French right double quotes (similar to $\gg$ ).
$\backslash$ - like the old $\backslash$-, 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.

```
34.1〈*Code\rangle
34.2\LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e. by the command,portugeswillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof$\backslash1@portuges~to~see~whether~we~have~to~do~some-~$thinghere.Sinceitispossibletoloadthisfilewithanyofthefollowingfouroptionstobabel:portuges,portuguese,brazilandbrazilianwealsoallowthatthehyphenationpatternsareloadedunderanyofthesefournames.Wejusthavetofindoutwhichonewasused.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
\ifx\l@portuges \@undefined
    \ifx\l@portuguese\@undefined
        \ifx\l@brazil\@undefined
            \ifx\l@brazilian\@undefined
                    \@nopatterns\{Portuguese\}
                    \adddialect\1@portuges0
            \else
                    \let\l@portuges\l@brazilian
            \(\backslash f i\)
        \else
            \let\l@portuges\l@brazil
        \(\backslash f i\)
    \else
        \let\1@portuges \l@portuguese
```

[^26]```
34.17 \fi
```

34.18 \fi

By now \1@portuges is defined. When the language definition file was loaded under a different name we make sure that the hyphenation patterns can be found.

```
34.19\expandafter\ifx\csname 1@\CurrentOption\endcsname\relax
34.20 \expandafter\let\csname l@\CurrentOption\endcsname\l@portuges
34.21 \fi
```

Now we have to decide whether this language definition file was loaded for Portuguese or Brasilian 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.

```
34.22 \def\bbl@tempa{portuguese}
34.23\ifx\CurrentOption\bbl@tempa
34.24 \let\bbl@tempb\@empty
34.25\else
34.26 \def\bbl@tempa{portuges}
34.27 \ifx\CurrentOption\bbl@tempa
            \let\bbl@tempb\@empty
    \else
                \def\bbl@tempb{brazil}
        \fi
        \i
34.33\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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

| 34.34 | \@namedef\{captions $\backslash$ CurrentOption\} |
| :--- | :---: |
| 34.35 | \def $\backslash$ prefacename\{Pref $\backslash$ 'acio\}\% |
| 34.36 | \def $\backslash$ refname\{Refer\^encias\}\% |
| 34.37 | \def $\backslash$ abstractname\{Resumo\}\% |
| 34.38 | \def $\backslash$ bibname\{Bibliografia\}\% |
| 34.39 | \def $\backslash$ chaptername\{Cap\'\{\i\}tulo\}\% |
| 34.40 | \def\appendixname\{Ap\`endice\}\% |

Some discussion took place around the correct translations for 'Table of Contents' and 'Index'. the translations differ for Portuguese and Brasilian 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 apeared with 'Índice' at the begining 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'.

| 34.41 | \def \contentsname\{Conte\'udo\}\% |
| :---: | :---: |
| 34.42 | \def \listfigurename\{Lista de Figuras\}\% |
| 34.43 | \def \listtablename\{Lista de Tabelas\}\% |
| 34.44 | \def \indexname\{\’Indice\}\% |
| 34.45 | $\backslash$ def $\backslash$ figurename\{Figura\}\% |
| 34.46 | \def \tablename\{Tabela\}\% |
| 34.47 | \def \partname\{Parte\}\% |
| 34.48 | \def \enclname\{Anexo\}\% |
| 34.49 | \def \ccname\{Com c\'opia a\}\% |
| 34.50 | $\backslash$ def \headtoname\{Para\}\% |
| 34.51 | $\backslash$ def $\backslash$ pagename $\{$ P \}  'agina\}\%  |
| 34.52 | \def \seename\{ver\}\% |
| 34.53 | \def \alsoname\{ver tamb\'em\}\% |

An alternate term for 'Proof' could be 'Prova'.
$\backslash$ def $\backslash$ proofname $\{$ Demonstra $\backslash c\{c\} \backslash \sim a 0\} \%$
\def \glossaryname\{Gloss\’ario\}\%
\}
\dateportuges The macro \dateportuges redefines the command \today to produce Portuguese dates.

```
34.57 \@namedef{date\CurrentOption}{%
34.58\\def\today{\number\day\space de\space\ifcase\month\or
34.59 Janeiro\or Fevereiro\or Mar\c{c}o\or Abril\or Maio\or Junho\or
34.60 Julho\or Agosto\or Setembro\or Outubro\or Novembro\or Dezembro%
34.61 \fi
34.62 \space de\space\number\year}}
34.63\else
```

For the Brasilian version of these definitions we just add a "dialect".

```
34.64 \expandafter
34.65 \adddialect\csname l@\CurrentOption\endcsname\l@portuges
```

\captionsbrazil The "captions" are different for both versions of the language, so we define the macro \captionsbrazil here.

| 34.66 | $\backslash @$ namedef\{captions\CurrentOption\}\{\% |
| :---: | :---: |
| 34.67 | \def \prefacename\{Pref\'acio\}\% |
| 34.68 | \def \refname\{Refer\^encias\}\% |
| 34.69 | \def \abstractname\{Resumo\}\% |
| 34.70 | \def\bibname\{Refer\^encias Bibliogr\'aficas\}\% |
| 34.71 | \def \chaptername\{Cap\'\{\i\}tulo\}\% |
| 34.72 | \def \appendixname\{Ap\^endice\}\% |
| 34.73 | \def \contentsname\{Sum\'ario\}\% |
| 34.74 | $\backslash d e f \backslash l i s t f i g u r e n a m e\{L i s t a ~ d e ~ F i g u r a s\} \% ~$ |
| 34.75 | $\backslash d e f \backslash l i s t t a b l e n a m e\{L i s t a ~ d e ~ T a b e l a s\} \% ~$ |
| 34.76 | \def \indexname\{\'Indice Remissivo\}\% |
| 34.77 | \def \figurename\{Figura\}\% |
| 34.78 | \def \tablename\{Tabela\}\% |
| 34.79 | \def \partname\{Parte\}\% |

```
34.80\\def\enclname{Anexo}%
34.81 \def\ccname{C\'opia para}%
34.82 \def\headtoname{Para}%
34.83 \def\pagename{P\'agina}%
34.84 \def\seename{veja}%
34.85 \def\alsoname{veja tamb\'em}%
34.86\\def\proofname{Demonstra\c{c}\~ao}%
34.87\\def\glossaryname{Glossary}% <-- Needs translation
34.88
}
```

\datebrazil The macro \datebrazil redefines the command \today to produce Brasilian dates, for which the names of the months are not capitalized.

```
34.89 \@namedef{date\CurrentOption}{%
34.90\\def\today{\number\day\space de\space\ifcase\month\or
34.91 janeiro\or fevereiro\or mar\c{c}o\or abril\or maio\or junho\or
34.92 julho\or agosto\or setembro\or outubro\or novembro\or dezembro%
34.93 \fi
34.94 \space de\space\number\year}}
34.95\fi
```

$\backslash$ portugeshyphenmins Set correct values for \lefthyphenmin and \righthyphenmin. $34.96 \backslash$ providehyphenmins $\{\backslash$ CurrentOption $\}\{$ tw@ $\backslash$ thr@@\}
\extrasportuges The macro \extrasportuges will perform all the extra definitions needed for the \noextrasportuges 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.

```
34.97\initiate@active@char{"}
34.98\@namedef{extras\CurrentOption}{\languageshorthands{portuges}}
34.99 \expandafter\addto\csname extras\CurrentOption\endcsname{%
34.100 \bbl@activate{"}}
```

Don't forget to turn the shorthands off again.
34.101 \addto\noextrasportuges\{\bbl@deactivate\{"\}\}

First we define access to the guillemets for quotations,

```
34.102 \declare@shorthand{portuges}{"<}{%
34.103 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
34.104 \declare@shorthand{portuges}{">}{%
34.105 \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.
34.106 \declare@shorthand\{portuges\}\{"-\}\{\nobreak-\bbl@allowhyphens\}
34.107 \declare@shorthand\{portuges\}\{""\}\{\hskip\z@skip\}

And we want to have a shorthand for disabling a ligature.

```
34.108 \declare@shorthand{portuges}{"|}{%
34.109 \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

\- All that is left now is the redefinition of $\backslash$-. The new version of $\backslash$ - should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ can generate from the hyphenation patterns.

```
34.110 \expandafter\addto\csname extras\CurrentOption\endcsname{%
34.111 \babel@save\-}
34.112 \expandafter\addto\csname extras\CurrentOption\endcsname{%
34.113\def\-{\allowhyphens\discretionary{-}{}{}\allowhyphens}}
```

\ord We also provide an easy way to typeset ordinals, both in the male (\ord or \ro)
$\backslash$ ro and the female (orda or $\backslash \mathrm{ra}$ ) form.
\ordga.114 \def $\backslash o r d\{\$ \sim\{\backslash$ rm o\}\$\}
\ra4.115 \def\orda\{\$~\{\rm a\}\$\}
$34.116 \backslash$ let $\backslash r o \backslash o r d \backslash$ let $\backslash r a \backslash o r d a$
The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of © to its original value.
34.117 \ldf@finish\CurrentOption
34.118 〈/code〉

## 35 The Spanish language

The file spanish.dtx ${ }^{31}$ defines all the language-specific macros for the Spanish language.

Custumization is made following mainly the books on the subject by José Martínez de Sousa. By typesetting spanish.dtx directly you will get the full documentation (regrettably is in Spanish only, but it is pretty long). References in this part refers to that document. There are several aditional features documented in the Spanish version only.

This style provides:

- Translations following the International $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$ conventions, as well as \today.
- Shorthands listed in Table 9. Examples in subsection 3.4 are illustrative. Note that "~ has a special meaning in spanish different to other languages, and is used mainly in linguistic contexts.

```
'a acute accented a. Also for: e, i, o, u (both lowercase
    and uppercase).
'n \tilde{n}\mathrm{ (also uppercase).}
~n n}\mathrm{ n (also uppercase). Deprecated.
"u ü (also uppercase).
"a Ordinal numbers (also "A, "o, "0).
"c ç (also uppercase).
"rr rr, but -r when hyphenated
"- Like \-, but allowing hyphenation in the rest the
        word.
"= Like -, but allowing hyphenation in the rest the
        word.
"~ The hyphen is repeated at the very beginning of the
        next line if the word is hyphenated at this point.
"" Like "- but producing no hyphen sign.
~ _ Like - but with no break after the hyphen. Also for:
        en-dashes (~--) and em-dashes (~---).
"/ A slash slightly lowered, if necessary.
"| disable ligatures at this point.
"< Left guillemets.
"> Right guillemets.
<< \begin{quoting}. (See text.)
>> \end{quoting}. (See text.)
```

Table 9: Extra definitions made by file spanish.ldf

[^27]- \deactivatetilden deactivates the ${ }^{\sim} \mathrm{n}$ and ${ }^{\sim} \mathrm{N}$ shorthands.
- In math mode a dot followed by a digit is replaced by a decimal comma.
- Spanish ordinals and abbeviations with \sptext as, for instance, 1 \sptext\{er\}. 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 \selectspanish selects the spanish language and its shorthands. (Intended for the preamble.)
- \frenchspacing is used.
- Ellipsis are best typed ... or, within a sentence, \...
- There is a small space before $\backslash \%$.
- \lisc provides lowercase small caps. (See subsection 3.10.)

Just in case spanish is the main language, the group \layoutspanish 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 Spanish rules.
- Both \alph and $\backslash$ 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 $\backslash \mathrm{gg}$.
- \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 tooljust $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ it and follow the instructions.
- There is a dot after section numbers in titles and toc.

This group is ignored if you write \selectspanish* in the preamble.
Some additional commands are provided to be used in the spanish.cfg file:

- With \es@activeacute acute accents are always active, overriding the default babel behaviour.
- \es@enumerate sets the labels to be used by enumerate. The same applies to \es@itemize and itemize.
- \es@operators stores the operator commands. All of them are canceled with

```
\let\es@operators\relax
```

The commands \deactivatequoting, \deactivatetilden and \selectspanish may be used in this file, too.

A subset of these commands is provided for use in Plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ (with \input spanish.sty).

### 35.1 The Code

This file provides definition for both $\mathrm{IAT}_{E} \mathrm{X} 2_{\varepsilon}$ and non $\mathrm{IAT}_{E} \mathrm{X} 2_{\varepsilon}$ formats.
Identify the ldf file.

```
35.1 \*code\
```

35.2 \ProvidesLanguage\{spanish.ldf\}
$35.3 \quad[2005 / 03 / 31 \mathrm{v} 4.2 \mathrm{~b}$ Spanish support from the babel system]

The macro dfInittakescareofpreventingthatthisfileisloadedmorethanonce,checkingthecategorycodeofthe@sign,etc.Whenthisfileisreadasanoption,i.e.bythe\usepackagecommand,spanishwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@spanishtoseewhetherwehavetodosomethinghere.35.4\LdfInit\{spanish\}\captionsspanish35.5\ifx\undefined$\backslash1@$spanish35.6\@nopatterns\{Spanish\}35.7\adddialect\1@spanish035.8\fiundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

We define some tools which will be used in that style file: (1) we make sure that ~ is active, (2) \es@delayed delays the expansion of the code in conditionals
(in fact, quite similar to \bbl@afterfi).
35.9 \edef $\backslash$ es@savedcatcodes $\{\%$
$35.10 \backslash$ catcode ${ }^{\text {}} \backslash$ noexpand $\backslash$ ~ $=\backslash$ the $\backslash$ catcode ${ }^{\prime} \backslash$ ~
35.11 \catcode' $\backslash$ noexpand $\backslash$ "=\the $\backslash$ catcode" $\backslash$ " $\}$
35.12 \catcode' ) $^{\sim}=$ = active
35.13 \catcode‘\"=12
$5.14 \backslash$ long $\backslash$ def $\backslash e s @ d e l a y e d \# 1 \backslash t h e n \# 2 \backslash e l s e \# 3 \backslash f i\{\%$
35.15 \#1\%
5.16 \expandafter\@firstoftwo
35.17 \else
55.18 \expandafter \@secondoftwo
35.19 \fi
35.20 \{\#2\}\{\#3\}\}

Two tests are introduced. The first one tells us if the format is $\operatorname{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$, and the second one if the format is plain or any other. If both are false, the format is LETEX2.09.
35.21 \es@delayed
35.22 \expandafter\ifx\csname documentclass $\backslash e n d c s n a m e \backslash r e l a x \backslash t h e n$
35.23 \let\ifes@LaTeXe\iffalse
35.24 \else
35.25 \let\ifes@LaTeXe\iftrue
35.26 \fi
35.27 \es@delayed
35.28 \expandafter\ifx\csname newenvironment \endcsname\relax $\backslash$ then
35.29 \let\ifes@plain\iftrue
$35.30 \backslash e l$ se
35.31 \let\ifes@plain\iffalse
35.32 \fi

Translations for captions.
35.33 \addto\captionsspanish\{\%
35.34 \def $\backslash$ prefacename\{Prefacio\}\%
35.35 \def $\backslash$ refname\{Referencias\}\%
35.36 \def $\backslash$ abstractname\{Resumen\}\%
35.37 \def\bibname\{Bibliograf\'\{\i\}a\}\%
35.38 \def\chaptername\{Cap\’\{\i\}tulo\}\%
35.39 \def\appendixname\{Ap\'endice\}\%
35.40 \def $\backslash$ listfigurename\{\'Indice de figuras\}\%
35.41 \def \listtablename\{\'Indice de cuadros\}\%
35.42 \def \indexname\{\'Indice alfab\'etico\}\%
35.43 \def\figurename\{Figura\}\%
35.44 \def $\backslash t a b l e n a m e\{C u a d r o\} \%$
35.45 \def \partname\{Parte\}\%
35.46 \def \enclname\{Adjunto\}\%
35.47 \def \ccname\{Copia a\}\%
35.48 \def $\backslash$ headtoname\{A\}\%
35.49 \def\pagename\{P\’agina\}\%
35.50 \def \seename\{v\'ease\}\%
35.51 \def \alsoname\{v\'ease tambi\'en\}\%
35.52 \def\proofname\{Demostraci\’on\}\%
$35.53 \backslash$ def \glossaryname\{Glosario\}\}
35.54
35.55 \expandafter\ifx $\backslash c s n a m e ~ c h a p t e r \backslash e n d c s n a m e \backslash r e l a x$
35.56 \addto\captionsspanish\{\def \contentsname\{\'Indice\}\}
35.57 \else
35.58 \addto\captionsspanish\{\def\contentsname\{\'Indice general\}\}
35.59 \fi

And the date.
$35.60 \backslash$ def $\backslash$ datespanish\{\%
35.61 \def \today\{\the\day~de \ifcase\month\or enero\or febrero\or
35.62 marzo\or abril\or mayo\or junio\or julio\or agosto\or
35.63 septiembre \or octubre\or noviembre\or diciembre\fi
$35.64 \backslash$ de\ifnum \year>1999\es@yearl\fi~\the\year\}\}
35.65 \def $\backslash$ spanishdatedel $\{\backslash$ def $\backslash e s @ y e a r l\{1\}\}$
35.66 \def \spanishdatede\{\let\es@yearl\@empty\}
35.67 \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.
5.68 \def $\backslash$ selectspanish\{\%
35.69 \def $\backslash$ selectspanish $\{\%$
55.70 \def \selectspanish\{\%
35.71 \PackageWarning\{spanish\}\{Extra \string\selectspanish ignored\}\}\%
35.72 \es@select\}\}
35.73
35.74 \@onlypreamble\selectspanish
35.75
35.76 \def $\backslash$ es@select\{\%
55.77 \let\es@select\@undefined
35.78 \selectlanguage\{spanish\}\%
35.79 \catcode' $\backslash$ " $\backslash$ active \catcode ${ }^{\prime}$ ~ $=$ =active\}

Instead of joining all the extras directly in \extrasspanish, we subdivide them in three further groups.
35.80 \def \extrasspanish$\{\%$
35.81 \textspanish
55.82 \mathspanish
35.83 \ifx\shorthandsspanish\@empty
5.84 \spanishdeactivate\{."'~<>\}\%
5.85 \languageshorthands\{none\}\%
35.86 \else
\shorthandsspanish
\fi\}
\def\noextrasspanish\{\%
\ifx\textspanish\@empty\else
\notextspanish
\fi
\ifx\mathspanish\@empty\else
$\backslash$ nomathspanish
\fi
\ifx\shorthandsspanish\@empty\else
\noshorthandsspanish
\fi
\es@reviveshorthands\}
And the first of these sub-groups is defined.
35.100 \addto\textspanish\{\%
35.101 \babel@save\sptext
35.102 \def \sptext\{\protect\es@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.

```
35.103 \ifes@LaTeXe %<<<<<<<
35.104 \newcommand\es@sptext[1]{%
35.105 {.\setbox\z@\hbox{8}\dimen@\ht\z@
35.106 \csname S@\f@size\endcsname
```

```
35.107 \edef\@tempa{\def\noexpand\@tempc{#1}%
    \lowercase{\def\noexpand\@tempb{#1}}}\@tempa
    \ifx\@tempb\@tempc
        \fontsize\sf@size\z@
        \selectfont
        \advance\dimen@-1.15ex
        \else
        \fontsize\ssf@size\z@
        \selectfont
        \advance\dimen@-1.5ex
    \fi
    \math@fontsfalse\raise\dimen@\hbox{#1}}}
lelse %<<<<<<<
\let\sptextfont\rm
\newcommand\es@sptext[1]{%
        {.\setbox\z@\hbox{8}\dimen@\ht\z@
            \edef\@tempa{\def\noexpand\@tempc{#1}%
                \lowercase{\def\noexpand\@tempb{#1}}}\@tempa
            \ifx\@tempb\@tempc
                \advance\dimen@-0.75ex
                \raise\dimen@\hbox{$\scriptstyle\sptextfont#1$}%
            \else
                \advance\dimen@-0.8ex
                \raise\dimen@\hbox{$\scriptscriptstyle\sptextfont#1$}%
        \fi}}
    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.
35.133 \ifes@LaTeXe $\% \lll \lll$
35.134 \addto\textspanish\{\%
$35.135 \backslash$ babel@save\lsc
35.136
$\backslash$ def $\backslash 1$ lsc\{\protect\es@lsc\}\}
35.137
35.138 \def $\backslash e s @ 1 s c \# 1\{\%$
35.139 \leavevmode
$35.140 \backslash$ hbox $\{$ scshape $\backslash$ selectfont
$35.141 \backslash$ expandafter $\backslash i f x \backslash c s n a m e \backslash f @ e n c o d i n g / \backslash f @ f a m i l y / \backslash f @ s e r i e s$
$/ \mathrm{n} / \backslash \mathrm{f}$ @size
\csname\curr@fontshape/\f@size\endcsname
\csname S@\f@size\endcsname
\fontsize\sf@size\z@\selectfont
$\backslash$ PackageInfo\{spanish\}\{Replacing undefined sc font $\backslash$ MessageBreak shape by faked small caps\}\%
$\backslash$ MakeUppercase\{\#1\}\% \else
\MakeLowercase\{\#1\}\% \fi\}\}
$\% \lll \lll<$

The quoting environment. This part is not available in Plain, hence the test. Overriding the default \everypar is a bit tricky.
35.153 \newif \ifes@listquot
35.154
35.155 \ifes@plain\else $\% \lll \lll$
35.156 \csname newtoks \endcsname\es@quottoks
35.157 \csname newcount \endcsname\es@quotdepth
35.158
35.159 \newenvironment\{quoting\}
$35.160 \quad\{\backslash$ leavevmode
35.161 \advance\es@quotdepth1
35.162 \csname lquot\romannumeral\es@quotdepth\endcsname\%
35.163 \ifnum\es@quotdepth=\@ne
35.164 \es@listquotfalse
35.165 \let\es@quotpar\everypar
35.166 \let\everypar\es@quottoks
35.167 \everypar\expandafter\{\the $\backslash e s @ q u o t p a r\} \%$
35.168 \es@quotpar\{\the $\backslash e v e r y p a r$
$35.169 \backslash i f e s @ l i s t q u o t \backslash g l o b a l \backslash e s @ l i s t q u o t f a l s e \backslash e l s e \backslash e s @ q u o t c o n t \backslash f i\} \%$
$35.170 \quad \backslash \mathrm{fi}$
35.171 \toks@\expandafter\{\es@quotcont\}\%
$35.172 \backslash$ edef $\backslash e s @ q u o t c o n t\{\backslash t h e \backslash t o k s @$
$35.173 \quad$ \expandafter $\backslash$ noexpand
35.174 \csname rquot $\backslash$ romannumeral \es@quotdepth\endcsname\}\}
35.175 \{\csname rquot\romannumeral\es@quotdepth\endcsname\}
35.176
$35.177 \backslash$ def $\backslash$ lquoti\{ $\backslash$ guillemotleft $\}\}$
35.178 \def $\backslash$ rquoti\{\guillemotright\{\}\}
35.179 \def\lquotii\{‘‘\}
35.180 \def $\backslash$ rquotii\{','\}
35.181 \def\lquotiii\{‘\}
$35.182 \backslash$ def $\backslash$ rquotiii\{'\}
35.183
35.184 \let\es@quotcont \@empty

If there is a margin par inside quoting, we don't add the quotes. \es@listqout stores the quotes to be used before item labels; otherwise they could appear after the labels.

```
35.185 \addto\@marginparreset{\let\es@quotcont\@empty}
35.186
35.187\def\es@listquot{%
35.188 \csname rquot\romannumeral\es@quotdepth\endcsname
35.189 \global\es@listquottrue}
35.190 \fi %<<<<<<
```

Now, the \frenchspacing, followed by \... and <br>%
35.191 \addto\textspanish $\{\backslash$ bbl@frenchspacing\}
35.192 \addto\notextspanish $\{\backslash$ bbl@nonfrenchspacing\}
35.193
35.194 \addto\textspanish\{\%
35.195 \let\es@save@dot\.\%

```
35.196 \babel@save\.%
35.197 \def\.{\@ifnextchar.{\es@dots}{\es@save@dot}}}
35.198
35.199\def\es@dots..{\leavevmode\hbox{...}\spacefactor\@M}
35.200
35.201 \ifes@LaTeXe %<<<<<<<
35.202 \addto\textspanish{%
35.203 \let\percentsign\%%
35.204 \babel@save\%%
35.205\\def\%{\unskip\,\percentsign{}}}
35.206 \else
35.207 \addto\textspanish{%
35.208 \let\percentsign\%%
35.209 \babel@save\%%
35.210 \def\%{\unskip\ifmmode\,\else$\m@th\,$\fi\percentsign{}}}
35.211 \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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$. There are two versions, depending on the format.
35.212 \addto\mathspanish\{\%
35.213 \babel@save\dotlessi
35.214 \def\dotlessi\{\protect\es@dotlessi\}\}
35.215
35.216 \let $\backslash$ nomathspanish $\backslash r e l a x$ \%\% Unused, but called
35.217
35.218 \ifes@LaTeXe $\% \lll \lll$
35.219 \def $\backslash e s @ t e x t i\{\backslash i\}$
35.220 \addto\@uclclist\{\dotlessi\es@texti\}
35.221 fi $\% \lll \lll$
35.222
35.223 \ifes@LaTeXe $\% \lll \lll$
35.224 \def\es@dotlessi\{\%
35.225 \ifmmode
$35.226 \quad\{\backslash$ ifnum $\backslash$ mathgroup= $\backslash m @$ ne
35.227 \imath
35.228 \else
35.229 \count@\escapechar \escapechar=\m@ne
35.230 \expandafter $\backslash$ expandafter $\backslash$ expandafter
35.231 \split@name\expandafter\string\the\textfont\mathgroup\@nil
35.232 \escapechar=\count@
35.233 \@ifundefined\{\f@encoding\string\i\}\%
$35.234 \quad\{\backslash e d e f \backslash f @ e n c o d i n g\{\backslash$ string? $\}\}\} \%$
$35.235 \backslash$ expandafter $\backslash c o u n t @ \backslash t h e \backslash c s n a m e \backslash f @ e n c o d i n g \backslash s t r i n g \backslash i \backslash e n d c s n a m e$
35.236 \advance\count@"7000
35.237 \mathchar\count@
$35.238 \quad \backslash f i\} \%$
35.239 \else


The switches for accents and spaces in math.
$35.250 \backslash$ def $\backslash$ accentedoperators $\{\%$

```
35.251 \def\es@op@ac##1{\acute{##1}}%
35.252 \def\es@op@i{\acute{\dotlessi}}}
35.253\def\unaccentedoperators{%
35.254\def\es@op@ac##1{##1}%
35.255 \def\es@op@i{i}}
35.256 \accentedoperators
35.257
35.258\def\spacedoperators{\let\es@op@sp\,}
35.259\def\unspacedoperators{\let\es@op@sp\@empty}
35.260 \spacedoperators
```

The operators are stored in \es@operators, which in turn is included in the math group. Since \operator@font is defined in $\mathrm{HT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ only, we defined in the plain variant.
35.261 \addto \mathspanish\{\es@operators\}
35.262
35.263 \ifes@LaTeXe\else $\% \lll \lll$
35.264 \let\operator@font $\backslash$ rm
35.265 \def \@empty\{\}
35.266 fi $\quad \% \lll \lll$
35.267
35.268 \def $\backslash$ es@operators\{\%
35.269 \babel@save\lim
35.270 \def\lim\{\mathop\{\operator@font l\protect\es@op@i m\}\}\%
35.271 \babel@save\limsup
35.272 \def $\backslash$ limsup\{ $\backslash$ mathop\{\operator@font l\es@op@i m\,sup\}\}\%
35.273 \babel@save\liminf
$35.274 \backslash \operatorname{def} \backslash l i m i n f\{\backslash$ mathop $\{$ \operator@font $1 \backslash e s @ o p @ i m$,inf\}\} $\%$
35.275 \babel@save $\backslash \max$
35.276 \def $\backslash \max \{\backslash$ mathop\{\operator@font m\es@op@ac ax\}\}\%
35.277 \babel@save\inf
35.278 \def $\backslash i n f\{\backslash m a t h o p\{\backslash o p e r a t o r @ f o n t ~ \ p r o t e c t \backslash e s @ o p @ i ~ n f\}\} \% ~$
35.279 \babel@save\min
35.280 \def $\backslash \min \{\backslash$ mathop\{\operator@font m\protect\es@op@i n\}\}\%
35.281 \babel@save $\backslash$ bmod
35.282 \def $\backslash$ bmod\{\%
$35.283 \backslash$ nonscript $\backslash m s k i p-\backslash m e d m u s k i p \backslash m k e r n 5 m u \%$
$35.284 \backslash$ mathbin\{\operator@font m\es@op@ac od\}\penalty900\mkern5mu\%

```
    \nonscript\mskip-\medmuskip}%
\babel@save\pmod
\def\pmod##1{%
    \allowbreak\mkern18mu({\operator@font m\es@op@ac od}\,\,##1)}%
    \def\es@a##1 {%
        \es@delayed
        \if^##1^\then % is it empty? do nothing and continue
        \es@a
    \else
        \es@delayed
        \if&##1\then % is it &? do nothing and finish
        \else
            \begingroup
                \let\,\@empty % \, is ignored when def'ing the macro name
                \let\acute\@firstofone % same
                \edef\es@b{\expandafter\noexpand\csname##1\endcsname}%
                \def\,{\noexpand\es@op@sp}%
                \def\acute####1{%
                    \if i####1%
                        \noexpand\es@op@i
                    \else
                        \noexpand\es@op@ac####1%
                        \fi}%
                \edef\es@a{\endgroup
                    \noexpand\babel@save\expandafter\noexpand\es@b
                            \def\expandafter\noexpand\es@b{%
                            \mathop{\noexpand\operator@font##1}\nolimits}}%
                \es@a % It restores itself
                \es@a
        \fi
    \fi}%
    \let\es@b\spanishoperators
    \addto\es@b{ }%
    \expandafter\es@a\es@b sen tg arc\,sen arc\, cos arc\,tg & }
35.319
35.320\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 shortands are called. The goals are to allow espression like $\$ a^{\wedge}\left\{x^{\prime}\right\} \$$ and to deactivate the shorthands making them of category 'other.' After providing a $\backslash$ 'i shorthand, the new macros are defined.

```
35.321 \DeclareTextCompositeCommand{\'}{0T1}{i}{\@tabacckludge'{\i}}
35.322
35.323\def\es@set@shorthand#1{%
35.324 \expandafter\edef\csname es@savecat\string#1\endcsname
        {\the\catcode`#1}%
35.326 \initiate@active@char{#1}%
35.327 \catcode`#1=\csname es@savecat\string#1\endcsname\relax
35.328 \expandafter\let\csname es@math\string#1\expandafter\endcsname
```

```
35.329 \csname normal@char\string#1\endcsname}
35.330
35.331 \def\es@use@shorthand{%
35.332 \es@delayed
35.333 \ifx\thepage\relax\then
35.334 \string
35.335 \else{%
35.336 \es@delayed
35.337 \ifx\protect\@unexpandable@protect\then
3 5 . 3 3 8 ~ \ n o e x p a n d
35.339 \else
35.340 \es@use@sh
35.341 \fi}%
35.342 \fi}
35.343
35.344\def\es@text@sh#1{\csname active@char\string#1\endcsname}
35.345\def\es@math@sh#1{\csname es@math\string#1\endcsname}
35.346
35.347\def\es@use@sh{%
35.348 \es@delayed
35.349 \if@safe@actives\then
35.350 \string
35.351 \else{%
35.352 \es@delayed
35.353 \ifmmode\then
35.354 \es@math@sh
35.355 \else
35.356 \es@text@sh
35.357 \fi}%
35.358 \fi}
35.359
35.360 \gdef\es@activate#1{%
35.361 \begingroup
35.362 \lccode`\ ~='#1
35.363 \lowercase{%
35.364 \endgroup
35.365 \def~{\es@use@shorthand~}}}
35.366
35.367 \def\spanishdeactivate#1{%
35.368 \@tfor\@tempa:=#1\do{\expandafter\es@spdeactivate\@tempa}}
35.369
35.370\def\es@spdeactivate#1{%
35.371 \if.#1%
35.372 \mathcode`\.=\es@period@code
35.373 \else
35.374 \begingroup
35.375 \lccode`\~=`#1
35.376 \lowercase{%
35.377 \endgroup
35.378 \expandafter\let\expandafter ~%
```

```
35.379 \csname normal@char\string#1\endcsname}%
35.380 \catcode`#1\csname es@savecat\string#1\endcsname\relax
35.381 \fi}
35.382
35.383 \def\es@reviveshorthands{%
35.384 \es@restore{"}\es@restore{~}%
35.385 \es@restore{<}\es@restore{>}%
35.386 \es@quoting}
35.387
35.388 \def\es@restore#1{%
35.389 \catcode`#1=\active
3 5 . 3 9 0 ~ \ b e g i n g r o u p ~
35.391 \lccode'\~='#1
35.392 \lowercase{%
35.393 \endgroup
35.394 \bbl@deactivate{~}}}
```

But spanish allows two category codes for ', so both should be taken into account in \bbl@pr@m@s.

```
35.395 \begingroup
```

35.396 \catcode' $\backslash$ ' = 12
35.397 \lccode'~=' ${ }^{\text {( }}$ lccode' $=$ ' '
35.398 \lowercase\{\%
35.399 \gdef $\backslash \mathrm{bbl@pr@m@s} \mathrm{\{ } \mathrm{\%}$
35.400 \es@delayed
35.401 \ifx~\@let@token\then
$35.402 \quad$ \pr@@@s
35.403 \else
$35.404 \quad\{\backslash e s @ d e l a y e d$
35.405 \ifx’\@let@token\then
35.406 \pr@@@s
35.407 \else
$35.408 \quad\{\backslash e s @ d e l a y e d$
35.409 \ifx^\@let@token\then
$35.410 \backslash \mathrm{pr@@@t}$
$35.411 \backslash e l s e$
else
\egroup
\fi\}\%
\fi\}\%
$35.414 \quad$ \f
35.416 \endgroup
35.417 \expandafter $\backslash i f x \backslash c s n a m e ~ @ t a b a c c k l u d g e \backslash e n d c s n a m e \backslash r e l a x$
35.418 \let\es@tak\a
35.419 \else
35.420 \let\es@tak\@tabacckludge
35.421 \fi
35.422
35.423 \ifes@LaTeXe $\% \lll \lll$
$35.424 \backslash$ def \@tabacckludge\#1\{\expandafter\es@tak\string\#1\}
35.425 \let\a\@tabacckludge

```
35.426 \else\ifes@plain %<<<<<<<
35.427 \def\@tabacckludge#1{\csname\string#1\endcsname}
35.428 \else %<<<<<<<
35.429\\def\@tabacckludge#1{\csname a\string#1\endcsname}
35.430 \fi\fi %<<<<<<<
35.431
35.432 \expandafter\ifx\csname add@accent\endcsname\relax
35.433 \def\add@accent#1#2{\accent#1 #2}
35.434\fi
```

Instead of redefining $\backslash$ ', we redefine the internal macro for the OT1 encoding.

```
35.435 \ifes@LaTeXe %<<<<<<<
35.436 \def\es@accent#1#2#3{%
35.437 \expandafter\@text@composite
35.438 \csname OT1\string#1\endcsname#3\@empty\@text@composite
35.439 {\bbl@allowhyphens\add@accent{#2}{#3}\bbl@allowhyphens
35.440 \setbox\@tempboxa\hbox{#3%
35.441 \global\mathchardef\accent@spacefactor\spacefactor}%
35.442 \spacefactor\accent@spacefactor}}
35.443 \else %<<<<<<<
35.444 \def\es@accent#1#2#3{%
35.445 \bbl@allowhyphens\add@accent{#2}{#3}\bbl@allowhyphens
35.446 \spacefactor\sfcode`#3 }
35.447 \fi %<<<<<<<
```

The shorthands are activated in the aux file. Now, we begin the shorthands group.

```
35.448 \addto\shorthandsspanish{\languageshorthands{spanish}}
```

35.449 \let $\backslash n o s h o r t h a n d s s p a n i s h \backslash r e l a x$

First, decimal comma.
$35.450 \backslash$ def $\backslash$ spanishdecimal\#1\{\def $\backslash e s @ d e c i m a l\{\{\# 1\}\}\}$
35.451 \def $\backslash$ decimalcomma\{\spanishdecimal\{,\}\}
35.452 \def $\backslash$ decimalpoint\{\spanishdecimal\{.\}\}
35.453 \decimalcomma
35.454
35.455 \es@set@shorthand\{.\}
35.456
35.457 \@namedef \{es@math\string.\}\{\%
35.458 \@ifnextchar\egroup
35.459 \{\mathchar\es@period@code\relax\}\%
35.460 \{\es@text@sh.\}\}
35.461
35.462 \declare@shorthand\{system\}\{.\}\{\mathchar\es@period@code\relax\}
35.463 \addto \shorthandsspanish\{\%
$35.464 \backslash$ mathchardef $\backslash$ es@period@code\the $\backslash$ mathcode' $\backslash . \%$
35.465 \babel@savevariable\{\mathcode'\.\}\%
35.466 \mathcode‘\.="8000 \%
35.467 \es@activate\{.\}\}
35.468

```
35.469\AtBeginDocument{%
35.470 \catcode`\.=12
35.471 \if@filesw
35.472 \immediate\write\@mainaux{%
35.473 \string\catcode'\string\.=12}%
35.474 \fi}
35.475
35.476 \declare@shorthand{spanish}{.1}{\es@decimal1}
35.477 \declare@shorthand{spanish}{.2}{\es@decimal2}
35.478 \declare@shorthand{spanish}{.3}{\es@decimal3}
35.479 \declare@shorthand{spanish}{.4}{\es@decimal4}
35.480 \declare@shorthand{spanish}{.5}{\es@decimal5}
35.481 \declare@shorthand{spanish}{.6}{\es@decimal6}
35.482 \declare@shorthand{spanish}{.7}{\es@decimal7}
35.483 \declare@shorthand{spanish}{.8}{\es@decimal8}
35.484 \declare@shorthand{spanish}{.9}{\es@decimal9}
35.485\declare@shorthand{spanish}{.0}{\es@decimal0}
```

Now accents and tools
35.486 \es@set@shorthand\{"\}
35.487 \def $\backslash e s @ u m l a u t \# 1\{\%$
35.488 \bbl@allowhyphens \add@accent\{127\}\#1\bbl@allowhyphens
35.489 \spacefactor $\backslash$ sfcode‘\#1 \}

We override the default " of babel, intended for german

```
35.490 \ifes@LaTeXe %<<<<<<<
35.491 \addto\shorthandsspanish{%
35.492 \es@activate{"}%
35.493 \es@activate{~}}
35.494 \babel@save\bbl@umlauta
35.495 \let\bbl@umlauta\es@umlaut
35.496 \expandafter\babel@save\csname OT1\string\~\endcsname
35.497 \expandafter\def\csname OT1\string\~\endcsname{\es@accent\~{126}}%
35.498 \expandafter\babel@save\csname OT1\string\'\endcsname
35.499 \expandafter\def\csname OT1\string\'\endcsname{\es@accent\'{19}}}
35.500 \else %<<<<<<<
35.501 \addto\shorthandsspanish{%
35.502 \es@activate{"}%
35.503 \es@activate{~}%
35.504 \babel@save\bbl@umlauta
35.505 \let\bbl@umlauta\es@umlaut
35.506 \babel@save\~%
35.507 \def\~{\es@accent\~{126}}%
35.508 \babel@save\'%
35.509 \def\'#1{\if#1i\es@accent\'{19}\i\else\es@accent\'{19}{#1}\fi}}
35.510\fi %<<<<<<<
35.511 \declare@shorthand{spanish}{"a}{\protect\es@sptext{a}}
35.512 \declare@shorthand{spanish}{"A}{\protect\es@sptext{A}}
35.513 \declare@shorthand{spanish}{"o}{\protect\es@sptext{o}}
35.514 \declare@shorthand{spanish}{"O}{\protect\es@sptext{0}}
```

```
35.515\declare@shorthand{spanish}{"e}{\protect\es@sptext@r{e}}
35.516 \declare@shorthand{spanish}{"E}{\protect\es@sptext@r{E}}
35.517
35.518\def\es@sptext@r#1#2{\es@sptext{#1#2}}
35.519
35.520 \declare@shorthand{spanish}{"u}{\"u}
35.521 \declare@shorthand{spanish}{"U}{\"U}
35.522
35.523 \declare@shorthand{spanish}{"c}{\c{c}}
35.524 \declare@shorthand{spanish}{"C}{\c{C}}
35.525
35.526 \declare@shorthand{spanish}{"<}{\guillemotleft{}}
35.527 \declare@shorthand{spanish}{">}{\guillemotright{}}
35.528 \declare@shorthand{spanish}{"-}{\bbl@allowhyphens\-\bbl@allowhyphens}
35.529 \declare@shorthand{spanish}{"=}%
35.530 {\bbl@allowhyphens\char\hyphenchar\font\hskip\z@skip}
35.531 \declare@shorthand{spanish}{"~}
35.532 {\bbl@allowhyphens\discretionary{\char\hyphenchar\font}%
35.533 {\char\hyphenchar\font}{\char\hyphenchar\font}\bbl@allowhyphens}
35.534 \declare@shorthand{spanish}{"r}
35.535 {\bbl@allowhyphens\discretionary{\char\hyphenchar\font}%
35.536 {}{r}\bbl@allowhyphens}
35.537\declare@shorthand{spanish}{"R}
35.538 {\bbl@allowhyphens\discretionary{\char\hyphenchar\font}%
35.539 {}{R}\bbl@allowhyphens}
35.540 \declare@shorthand{spanish}{"y}
35.541 {\@ifundefined{scalebox}%
35.542 {\ensuremath{\tau}}%
35.543 {\raisebox{1ex}{\scalebox{-1}{\resizebox{.45em}{1ex}{2}}}}}
35.544 \declare@shorthand{spanish}{""}{\hskip\z@skip}
35.545 \declare@shorthand{spanish}{"/}
35.546 {\setbox\z@\hbox{/}%
35.547 \dimen@\ht\z@
35.548 \advance\dimen@-1ex
35.549 \advance\dimen@\dp\z@
35.550 \dimen@.31\dimen@
35.551 \advance\dimen@-\dp\z@
35.552 \ifdim\dimen@>0pt
35.553 \kern.01em\lower\dimen@\box\z@\kern.03em
35.554 \else
35.555 \box\z@
35.556 \fi}
35.557 \declare@shorthand{spanish}{"?}
35.558 {\setbox\z@\hbox{?`}%
35.559 \leavevmode\raise\dp\z@\box\z@}
35.560 \declare@shorthand{spanish}{"!}
35.561 {\setbox\z@\hbox{!`}%
35.562 \leavevmode\raise\dp\z@\box\z@}
35.563
35.564 \es@set@shorthand{~}
```

```
35.565 \declare@shorthand{spanish}{~n}{\~n}
35.566 \declare@shorthand{spanish}{~N}{\~N}
35.567 \declare@shorthand{spanish}{~ -}{%
35.568 \leavevmode
35.569 \bgroup
35.570 \let\@sptoken\es@dashes % This assignation changes the
35.571 \@ifnextchar-% \@ifnextchar behaviour
35.572 {\es@dashes}%
35.573 {\hbox{\char\hyphenchar\font}\egroup}}
35.574 \def\es@dashes-{%
35.575 \@ifnextchar-%
35.576 {\bbl@allowhyphens\hbox{---}\bbl@allowhyphens\egroup\@gobble}%
35.577 {\bbl@allowhyphens\hbox{--}\bbl@allowhyphens\egroup}}
35.578
35.579 \def\deactivatetilden{%
35.580 \expandafter\let\csname spanish@sh@\string~@n@\endcsname\relax
35.581 \expandafter\let\csname spanish@sh@\string~@N@\endcsname\relax}
    The shorthands for quoting.
35.582 \expandafter\ifx\csname XML@catcodes\endcsname\relax
35.583 \addto\es@select{%
35.584 \catcode`\<\active\catcode`\>=\active
35.585 \es@quoting}
35.586
35.587 \es@set@shorthand{<}
35.588 \es@set@shorthand{>}
35.589
35.590 \declare@shorthand{system}{<}{\csname normal@char\string<\endcsname}
35.591 \declare@shorthand{system}{>}{\csname normal@char\string>\endcsname}
35.592
35.593 \addto\shorthandsspanish{%
35.594 \es@activate{<}%
35.595 \es@activate{>}}
35.596 \ifes@LaTeXe %<<<<<<<
35.597 \AtBeginDocument{%
35.598 \es@quoting
35.599 \if@filesw
35.600 \immediate\write\@mainaux{\string\es@quoting}%
35.601 \fi}%
35.602 \fi %<<<<<<<
35.603
35.604
\catcode'>=\active \catcode'<=\active
35.606 \let\es@quoting\activatequoting}
35.607 \def\deactivatequoting{%
35.608 \catcode'>=12 \catcode'<=12
35.609 \let\es@quoting\deactivatequoting}
35.610
35.611 \declare@shorthand{spanish}{<<}{\begin{quoting}}
35.612 \declare@shorthand{spanish}{>>}{\end{quoting}}
```

```
35.613\fi
35.614
35.615\let\es@quoting\relax
35.616\let\deactivatequoting\relax
35.617\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 $\mathrm{LAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ formats is always executed.
35.618 \def \es@activeacute\{ $\%$
35.619 \es@set@shorthand\{'\}\%
35.620 \addto\shorthandsspanish\{\es@activate\{'\}\}\%
35.621 \addto\es@reviveshorthands\{\es@restore\{'\}\}\%
35.622 \addto\es@select\{\catcode' '=\active\}\%
35.623 \declare@shorthand\{spanish\}\{'a\}\{\@tabacckludge'a\}\%
35.624 \declare@shorthand\{spanish\}\{'A\}\{\@tabacckludge'A\}\%
35.625 \declare@shorthand\{spanish\}\{'e\}\{\@tabacckludge'e\}\%
35.626 \declare@shorthand\{spanish\}\{'E\}\{\@tabacckludge'E\}\%
35.627 \declare@shorthand\{spanish\}\{'i\}\{\@tabacckludge'i\}\%
35.628 \declare@shorthand\{spanish\}\{'I\}\{\@tabacckludge'I\}\%
35.629 \declare@shorthand\{spanish\}\{'o\}\{\@tabacckludge'o\}\%
35.630 \declare@shorthand\{spanish\}\{'0\}\{\@tabacckludge'0\}\%
35.631 \declare@shorthand\{spanish\}\{'u\}\{\@tabacckludge'u\}\%
35.632 \declare@shorthand\{spanish\}\{'U\}\{\@tabacckludge'U\}\%
35.633 \declare@shorthand\{spanish\}\{'n\}\{\~n\}\%
35.634 \declare@shorthand\{spanish\}\{'N\}\{\~N\}\%
35.635 \declare@shorthand\{spanish\}\{', \}\{”\}\%
35.636 \let\es@activeacute\relax\}
35.637
35.638 \ifes@LaTeXe $\% \lll \lll$
35.639 \@ifpackagewith\{babel\}\{activeacute\}\{\es@activeacute\}\{\}
35.640 \else $\% \lll \lll$
35.641 \es@activeacute
35.642 fi $\% \lll \lll \%$

And the customization. By default these macros only store the values and do nothing.
35.643 \def \es@enumerate\#1\#2\#3\#4\{\%
$35.644 \backslash \mathrm{def}$ \es@enum\{\{\#1\}\{\#2\}\{\#3\}\{\#4\}\}\}
35.645
35.646 \def \es@itemize\#1\#2\#3\#4\{\%
35.647 \def\es@item\{\{\#1\}\{\#2\}\{\#3\}\{\#4\}\}\}

The part formerly in the .lld file comes here. It performs layout adaptation of $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ to "orthodox" Spanish rules.

```
35.648 \ifes@LaTeXe %<<<<<<<
35.649
35.650 \es@enumerate{1.}{a)}{1)}{a$'$}
35.651 \def\spanishdashitems{\es@itemize{---}{---}{---}{---}}
35.652\def\spanishsymbitems{%
35.653 \es@itemize
```

```
        {\leavevmode\hbox to 1.2ex
            {\hss\vrule height .9ex width .7ex depth -.2ex\hss}}%
        {\textbullet}%
        {$\m@th\circ$}%
        {$\m@th\diamond$}}
\def\spanishsignitems{%
    \es@itemize
        {\textbullet}%
        {$\m@th\circ$}%
        {$\m@th\diamond$}%
        {$\m@th\triangleright$}}
\spanishsymbitems
\def\es@enumdef#1#2#3\@@{%
    \if#21%
        \@namedef{theenum#1}{\arabic{enum#1}}%
        \else\if#2a%
        \@namedef{theenum#1}{\emph{\alph{enum#1}}}%
    \else\if#2A%
        \@namedef {theenum#1}{\Alph{enum#1}}%
    \else\if#2i%
        \@namedef{theenum#1}{\roman{enum#1}}%
        \else\if#2I%
        \@namedef{theenum#1}{\Roman{enum#1}}%
        \else\if#20%
        \@namedef{theenum#1}{\arabic{enum#1}\protect\es@sptext{o}}%
    \fi\fi\fi\fi\fi\fi
    \toks@\expandafter{\csname theenum#1\endcsname}
    \expandafter\edef\csname labelenum#1\endcsname
            {\noexpand\es@listquot\the\toks@#3}}
\addto\layoutspanish{%
    \def\es@enumerate##1##2##3##4{%
        \es@enumdef{i}##1\@empty\@empty\@@
        \es@enumdef{ii}##2\@empty\@empty\@@
        \es@enumdef{iii}##3\@empty\@empty\@@
        \es@enumdef{iv}##4\@empty\@empty\@@}%
    \def\es@itemize##1##2##3##4{%
        \def\labelitemi{\es@listquot##1}%
        \def\labelitemii{\es@listquot##2}%
        \def\labelitemiii{\es@listquot##3}%
        \def\labelitemiv{\es@listquot##4}}%
        \def\p@enumii{\theenumi}%
        \def\p@enumiii{\theenumi\theenumii}%
        \def\p@enumiv{\p@enumiii\theenumiii}%
        \expandafter\es@enumerate\es@enum
        \expandafter\es@itemize\es@item
        \DeclareTextCommand{\guillemotleft}{OT1}{%
        \ifmmode\ll
        \else
```

```
        \save@sf@q{\penalty\@M
    \leavevmode\hbox{\usefont{U}{lasy}{m}{n}%
                                \char40 \kern-0.19em\char40 }}%
    \fi}%
\DeclareTextCommand{\guillemotright}{OT1}{%
        \ifmmode\gg
        \else
            \save@sf@q{\penalty\@M
                    \leavevmode\hbox{\usefont{U}{lasy}{m}{n}%
                \char41 \kern-0.19em\char41 }}%
    \fi}%
    \def\@fnsymbol##1%
        {\ifcase##1\or*\or**\or***\or****\or
        *****\or******\else\@ctrerr\fi}%
    \def\@alph##1%
        {\ifcase##1\or 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 n\or \~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\@ctrerr\fi}%
    \def\@Alph##1%
        {\ifcase##1\or 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 N\or \~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\@ctrerr\fi}%
    \let\@afterindentfalse\@afterindenttrue
    \@afterindenttrue
    \def\@seccntformat##1{\csname the##1\endcsname.\quad}%
    \def\numberline##1{\hb@xt@\@tempdima{##1\if&##1&\else.\fi\hfil}}%
    \def\@roman##1{\protect\es@roman{\number##1}}%
    \def\es@roman##1{\protect\es@lsc{\romannumeral##1}}%
    \def\esromanindex##1##2{##1{\protect\es@lsc{##2}}}}
```

We need to execute the following code when babel has been run, in order to see if spanish is the main language.

```
35.733 \AtEndOfPackage{%
    \let\es@activeacute\@undefined
    \def\bbl@tempa{spanish}%
    \ifx\bbl@main@language\bbl@tempa
        \AtBeginDocument{\layoutspanish}%
        \addto\es@select{%
            \@ifstar{\let\layoutspanish\relax}%
                            {\layoutspanish\let\layoutspanish\relax}}%
    \fi
    \selectspanish}
35.743
35.744 \fi %<<<<<<<
```

After restoring the catcode of $\sim$ and setting the minimal values for hyphenation, the . ldf is finished.
35.745 \es@savedcatcodes
35.746
35.747 \providehyphenmins\{\CurrentOption\}\{\tw@\tw@\}
35.748

```
35.749 \ifes@LaTeXe %<<<<<<<
35.750 \ldf@finish{spanish}
35.751 \else %<<<<<<<
35.752 \es@select
35.753 \ldf@finish{spanish}
35.754 \csname activatequoting\endcsname
35.755 \fi %<<<<<<<
35.756
35.757 </code>
```

That＇s all in the main file．Now the file with custom－bib macros．
35.758 〈*bblbst〉
35.759 \def $\backslash \mathrm{bbland}\{\mathrm{y}\}$
$35.760 \backslash$ def $\backslash$ bbleditors\{directores \} \def $\backslash$ bbleds\{dirs. \@\}
35.761 \def \bbleditor\{director\} \def\bbled\{dir.\@\}
35.762 \def $\backslash$ bbledby\{dirigido por\}
$35.763 \backslash$ def \bbledition\{edici\’on\} \def\bbledn\{ed.\@\}
35.764 \def \bbletal\{y otros\}
$35.765 \backslash$ def $\backslash \mathrm{bblvolume} \mathrm{\{vol} \mathrm{\} \backslash u m e n\} \backslash$ def $\backslash \mathrm{bblvol}\{\mathrm{vol} . \backslash @\}$
$35.766 \backslash \mathrm{def} \backslash \mathrm{bblof}\{\mathrm{de}\}$
$35.767 \backslash$ def $\backslash \mathrm{bblnumber} \mathrm{\{n} \mathrm{\backslash 'umero} \mathrm{\}} \backslash \mathrm{def} \backslash \mathrm{bblno} \mathrm{\{n} \mathrm{\backslash sptext} \mathrm{\{o} \mathrm{\}} \mathrm{\}}$
35.768 \def $\backslash \mathrm{bblin}\{\mathrm{en}\}$
35.769 \def \bblpages\{p\'aginas\} \def\bblpp\{p\'ags.\@\}
35.770 \def \bblpage\{p\'gina\} \def $\backslash \mathrm{bblp}\{\mathrm{p} \backslash \prime a g . \backslash @\}$
35.771 \def \bblchapter\{cap\’itulo\} \def\bblchap\{cap.\@\}
35.772 \def \bbltechreport\{informe t\'ecnico\}
35.773 \def \bbltechrep\{inf.\@ t\'ec.\@\}
35.774 \def $\backslash \mathrm{bblmthesis}\{$ proyecto de fin de carrera\}
35.775 \def \bblphdthesis\{tesis doctoral\}
$35.776 \backslash d e f \backslash b b l f i r s t$ \{primera\} \def $\backslash$ bblfirsto $\{1 \backslash s p t e x t\{a\}\}$
$35.777 \backslash$ def $\backslash$ bblsecond\{segunda\} $\backslash$ def $\backslash$ bblsecondo\{2\sptext\{a\}\}
35.778 \def \bblthird \{tercera\} \def \bblthirdo \{3\sptext\{a\}\}
35.779 \def \bblfourth\{cuarta\} \def \bblfourtho\{4\sptext\{a\}\}
$35.780 \backslash$ def $\backslash b b l f i f t h ~\{q u i n t a\} ~ \ d e f \backslash b b l f i f t h o ~\{5 \backslash s p t e x t\{a\}\} ~$
35.781 \def \bblth\{\sptext\{a\}\}
$35.782 \backslash$ let $\backslash \mathrm{bbl}$ st $\backslash \mathrm{bblth} \backslash$ let $\backslash \mathrm{bblnd} \backslash \mathrm{bblth} \backslash$ let $\backslash \mathrm{bblrd} \backslash \mathrm{bblth}$
$35.783 \backslash d e f \backslash$ bbljan\{enero\} \def $\backslash$ bblfeb\{febrero\} \def $\backslash$ bblmar\{marzo\}
$35.784 \backslash$ def $\backslash b b l a p r\{a b r i l\} ~ \ d e f \backslash b b l m a y\{m a y o\} ~ \ d e f \backslash b b l j u n\{j u n i o\} ~$
35.785 \def \bbljul\{julio\} \def \bblaug\{agosto\} \def \bblsep\{septiembre\}
$35.786 \backslash \operatorname{def} \backslash \mathrm{bbloct}\{o c t u b r e\} \backslash d e f \backslash b b l n o v\{n o v i e m b r e\} \backslash d e f \backslash b b l d e c\{d i c i e m b r e\}$
35.787 〈/bblbst〉

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．

```
35.788 <*indexes〉
```

35.789 \makeatletter
35.790
35.791 \newcount \es@converted
35.792 \newcount $\backslash$ es@processed
35.793

```
35.794\def\es@encap{`\|}
35.795 \def\es@openrange{`\(}
35.796 \def\es@closerange{`\)}
35.797
35.798\def\es@split@file#1.#2\@@{#1}
35.799 \def\es@split@ext#1.#2\@@{#2}
35.800
35.801 \typein[\answer]{~`JArchivo que convertir^^J%
35.802 (extension por omision .idx):}
35.803
35.804 \@expandtwoargs\in@{.}{\answer}
35.805 \ifin@
35.806 \edef\es@input@file{\expandafter\es@split@file\answer\@@}
35.807 \edef\es@input@ext{\expandafter\es@split@ext\answer\@@}
35.808 \else
35.809 \edef\es@input@file{\answer}
35.810 \def\es@input@ext{idx}
35.811 \fi
35.812
35.813 \typein[\answer]{^^JArchivo de destino^^J%
35.814 (archivo por omision: \es@input@file.eix, ` J%
35.815 extension por omision .eix):}
35.816 \ifx\answer\@empty
35.817 \edef\es@output{\es@input@file.eix}
35.818 \else
35.819 \@expandtwoargs\in@{.}{\answer}
35.820 \ifin@
35.821 \edef\es@output{\answer}
35.822 \else
35.823 \edef\es@output{\answer.eix}
35.824 \fi
35.825 \fi
35.826
35.827 \typein[\answer] {%
35.828 ~~J?Se ha usado algun esquema especial de controles^^J%
3 5 . 8 2 9 ~ d e ~ M a k e I n d e x ~ p a r a ~ e n c a p , ~ o p e n \_ r a n g e ~ o ~ c l o s e \_ r a n g e ? ` ` J \% )
35.830 [s/n] (n por omision)}
35.831
35.832 \if s\answer
35.833 \typein[\answer]{~~JCaracter para 'encap'~~J%
35.834 (\string| por omision)}
35.835 \ifx\answer\@empty\else
35.836 \edef\es@encap{%
35.837 '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
35.838 \fi
35.839 \typein[\answer]{^`JCaracter para 'open_range'^^J%
35.840 (\string( por omision)}
35.841 \ifx\answer\@empty\else
35.842 \edef\es@openrange{%
35.843 '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
```

```
35.844 \fi
35.845 \typein[\answer]{~~JCaracter para 'close_range'~^J%
35.846 (\string) por omision)}
35.847 \ifx\answer\@empty\else
35.848 \edef\es@closerange{%
35.849 '\expandafter\noexpand\csname\expandafter\string\answer\endcsname}
35.850 \fi
35.851 \fi
35.852
35.853 \newwrite\es@indexfile
35.854 \immediate\openout\es@indexfile=\es@output
35.855
35.856\newif\ifes@encapsulated
35.857
35.858 \def\es@roman#1{\romannumeral#1 }
35.859\edef\es@slash{\expandafter\@gobble\string\\}
35.860
35.861 \def\indexentry{%
35.862 \begingroup
35.863 \@sanitize
35.864 \es@indexentry}
35.865
35.866 \begingroup
35.867
35.868 \catcode'\|=12 \lccode'\|=\es@encap\relax
35.869 \catcode'\(=12 \lccode'\(=\es@openrange\relax
35.870 \catcode`\)=12 \lccode`\)=\es@closerange\relax
35.871
35.872 \lowercase{
35.873 \gdef\es@indexentry#1{%
35.874 \endgroup
35.875 \advance\es@processed\@ne
35.876 \es@encapsulatedfalse
35.877 \es@bar@idx#1\\@@
35.878 \es@idxentry}%
35.879 }
35.880
35.881 \lowercase{
35.882 \gdef \es@idxentry#1{%
35.883 \in@{\es@roman}{#1}%
35.884 \ifin@
35.885 \advance\es@converted\@ne
35.886 \immediate\write\es@indexfile{%
35.887 \string\indexentry{\es@b|\ifes@encapsulated\es@p\fi esromanindex%
35.888 {\ifx\es@a\@empty\else\es@slash\es@a\fi}}{#1}}%
35.889 \else
35.890 \immediate\write\es@indexfile{%
35.891 \string\indexentry{\es@b\ifes@encapsulated|\es@p\es@a\fi}{#1}}%
35.892 \fi}
35.893 }
```

```
35.894
35.895 \lowercase{
35.896 \gdef\es@bar@idx#1|#2\@@{%
35.897\\def\es@b{#1}\def\es@a{#2}%
35.898 \ifx\es@a\@empty\else\es@encapsulatedtrue\es@bar@eat#2\fi}
35.899 }
35.900
35.901 \lowercase{
35.902 \gdef\es@bar@eat#1#2|{\def\es@p{#1}\def\es@a{#2}%
35.903\edef\es@t{(}\ifx\es@t\es@p
35.904\else\edef\es@t{)}\ifx\es@t\es@p
35.905 \else
35.906 \edef\es@a{\es@p\es@a}\let\es@p\@empty%
35.907 \fi\fi}
35.908}
35.909
35.910 \endgroup
3 5 . 9 1 1
35.912 \input \es@input@file.\es@input@ext
35.913
35.914 \immediate\closeout\es@indexfile
35.915
35.916 \typeout{******************}
35.917 \typeout{Se ha procesado: \es@input@file.\es@input@ext }
35.918\typeout{Lineas leidas: \the\es@processed}
35.919 \typeout{Lineas convertidas: \the\es@converted}
35.920 \typeout{Resultado en: \es@output}
35.921 \ifnum\es@converted>\z@
35.922 \typeout{Genere el indice a partir de ese archivo}
35.923\else
35.924 \typeout{No se ha realizado ningun tipo de conversion}
35.925 \typeout{Se puede generar el archivo directamente^^J%
35.926 de \es@input@file.\es@input@ext}
35.927 \fi
35.928 \typeout{******************}
35.929 \@@end
35.930 </indexes〉
```


## 36 The Catalan language

The file catalan.dtx ${ }^{32}$ 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 10 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 11 also become available ${ }^{33}$.
\l.l geminated-l digraph (similar to l-l). \L.L produces the uppercase version.
\lgem geminated-l digraph (similar to l-l). \Lgem produces the uppercase version.
\up Macro to help typing raised ordinals, like $1^{\mathrm{er}}$. Takes one argument.
\- like the old $\backslash$-, but allowing hyphenation in the rest of the word.
"i i with diaeresis, allowing hyphenation in the rest of the word. Valid for the following vowels: i, u (both lowercase and uppercase).
"c c-cedilla (ç). Valid for both uppercase and lowercase c.
"l geminated-l digraph (similar to $\mathrm{l} \cdot \mathrm{l}$ ). Valid for both uppercase and lowercase 1 .
" $<\quad$ French left double quotes (similar to $\ll$ ).
"> French right double quotes (similar to $\gg$ ).
"- explicit hyphen sign, allowing hyphenation in the rest of the word.
" $\mid$ disable ligature at this position.
Table 10: Extra definitions made by file catalan.ldf (activated by default)
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.

```
36.1 〈*code\rangle
36.2\LdfInit{catalan}\captionscatalan
```

When this file is read as an option, i.e. by the command,catalancouldbean'unknown'languageinwhichcasewehavetomakeitknown.Soweundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

[^28]'e acute accented a, allowing hyphenation in the rest of the word. Valid for the following vowels: e, i, o, u (both lowercase and uppercase).
'a grave accented a, allowing hyphenation in the rest of the word. Valid for the following vowels: a, e, o (both lowercase and uppercase).

Table 11: Extra definitions made by file catalan.ldf (activated only when using the options activeacute and activegrave)
check for the existence of \1@catalan to see whether we have to do something here.

```
36.3\ifx\l@catalan\@undefined
36.4 \@nopatterns{Catalan}
36.5 \adddialect\l@catalan0
36.6 \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.
36.7 \providehyphenmins\{catalan\}\{\tw@\tw@\}
\captionscatalan The macro \captionscatalan defines all strings used in the four standard documentclasses provided with $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
    36.8 \addto\captionscatalan{%
    36.9\\def\prefacename{Pr\'oleg}%
    36.10\\def\refname{Refer\'encies}%
    36.11\def\abstractname{Resum}%
    36.12 \def\bibname{Bibliografia}%
    36.13\\def\chaptername{Cap\'{\i}tol}%
    36.14\def\appendixname{Ap\'endix}%
    36.15 \def\contentsname{\'Index}%
    36.16 \def\listfigurename{\'Index de figures}%
    36.17 \def\listtablename{\'Index de taules}%
    36.18 \def\indexname{\'Index alfab\'etic}%
    36.19\def\figurename{Figura}%
    36.20 \def\tablename{Taula}%
    36.21 \def\partname{Part}%
    36.22 \def\enclname{Adjunt}%
    36.23 \def\ccname{C\'opies a}%
    36.24 \def\headtoname{A}%
    36.25\def\pagename{P\'agina}%
    36.26\def\seename{Vegeu}%
    36.27 \def\alsoname{Vegeu tamb\'e}%
    36.28\def\proofname{Demostraci\'o}%
    36.29\def\glossaryname{Glossari}%
```

\datecatalan The macro \datecatalan redefines the command \today to produce Catalan dates. Months are written in lowercase ${ }^{34}$.

```
6.31 \def\datecatalan{%
36.32\\def\today{\number\day~\ifcase\month\or
36.33 de gener\or de febrer\or de mar\c{c}\or d'abril\or de maig\or
36.34 de juny\or de juliol\or d'agost\or de setembre\or d'octubre\or
36.35 de novembre\or de desembre\fi
36.36 \space de~\number\year}}
```

\extrascatalan The macro \extrascatalan will perform all the extra definitions needed for the \noextrascatalan 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ will find more breakpoints in words that contain this character in its rôle as apostrophe.

```
36.37 \addto\extrascatalan{%
36.38 \lccode''=''}
36.39 \addto\noextrascatalan{%
36.40 \lccode''=0}
```

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.

```
36.41 \addto\extrascatalan{\languageshorthands{catalan}}
36.42 \initiate@active@char{"}
36.43 \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 actived grave character leads to error messages.

```
36.44 \@ifpackagewith{babel}{activegrave}{%
36.45 \AtBeginDocument{%
            \if@filesw\immediate\write\@auxout{\catcode096=12}\fi}
    \initiate@active@char{`}%
    }{}
\@ifpackagewith{babel}{activegrave}{%
    \addto\extrascatalan{\bbl@activate{`}}%
    }{}
36.52 \@ifpackagewith{babel}{activeacute}{%
36.53 \initiate@active@char{'}%
36.54 }{}
```

[^29]```
36.55\@ifpackagewith{babel}{activeacute}{%
36.56 \addto\extrascatalan{\bbl@activate{'}}%
36.57 }{}
```

Now make sure that the characters that have been turned into shorthanfd characters expand to 'normal' characters outside the catalan environment.

```
36.58 \addto\noextrascatalan{\bbl@deactivate{"}}
36.59 \@ifpackagewith{babel}{activegrave}{%
36.60 \addto\noextrascatalan{\bbl@deactivate{`}}}{}
36.61 \@ifpackagewith{babel}{activeacute}{%
36.62 \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.

```
36.63 \addto\extrascatalan\{\%
36.64 \babel@save\"\%
36.65 \def\"\{\protect\@umlaut\}\}\%
36.66 \@ifpackagewith\{babel\}\{activegrave\}\{\%
36.67 \babel@save\‘\%
36.68 \addto\extrascatalan\{\def \'\{\protect\@grave\}\}
36.69 \}\{\}
36.70 \@ifpackagewith\{babel\}\{activeacute\}\{\%
36.71 \babel@save\’\%
36.72 \addto\extrascatalan\{\def\'\{\protect\@acute\}\}
36.73 \}\{\}
```

All the code above is necessary because we need a few extra active characters. These characters are then used as indicated in tables 10 and 11.
\dieresis The original definition of $\backslash "$ is stored as \dieresis, because the definition of \textacute \" might not be the default plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ one. If the user uses PostScript fonts \textgrave 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 \accent"7F 1, but as \accent' 310 \#1. Something similar happens when using fonts that follow the Cork encoding. For this reason we save the definition of $\backslash "$ and use that in the definition of other macros. We do likewise for $\backslash^{\prime}$, and $\backslash$ '.

```
36.74\let\dieresis\"
36.75 \@ifpackagewith{babel}{activegrave}{\let\textgrave\'}{}
36.76 \@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 will be found in words that contain accents, but this is a limitation in $\mathrm{T}_{\mathrm{E}} \mathrm{X}$. 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ at least finds the breaks that are not too close to the accent. The warning will only be printed to the log file.

```
36.77\ifx\DeclareFontShape\@undefined
36.78 \wlog{Warning: You are using an old LaTeX}
36.79 \wlog{Some word breaks will not be found.}
36.80\def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
36.81 \@ifpackagewith{babel}{activeacute}{%
36.82 \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
36.83\\ifpackagewith{babel}{activegrave}{%
36.84\\def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{}
36.85\else
    \ifx\f@encoding\bbl@t@one
        \let\@umlaut\dieresis
        \@ifpackagewith{babel}{activeacute}{%
            \let\@acute\textacute}{}
        \@ifpackagewith{babel}{activegrave}{%
            \let\@grave\textgrave}{}
        \else
            \wlog{Warning: You are using encoding \f@encoding\space
            instead of T1.}
        \wlog{Some word breaks will not be found.}
        \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
        \@ifpackagewith{babel}{activeacute}{%
            \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
        \@ifpackagewith{babel}{activegrave}{%
            \def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{}
        \fi
36.102 \fi
```

If the user setup has extended fonts, the Ferguson macros are required to be defined. We check for their existance and, if defined, expand to whatever they are defined to. For instance, \'a would check for the existance 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- $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ because the hyphenation algorithm can work on the whole word. The following macros are directly derived from ML-TEX. ${ }^{35}$

Now we can define our shorthands: the diaeresis and "ela geminada" support, $36.103 \backslash$ declare@shorthand\{catalan\}\{"i\}\{\textormath\{\@umlaut $\backslash i\}\{\backslash$ ddot $\backslash$ imath\}\}

[^30]```
36.104 \declare@shorthand{catalan}{"l}{\lgem{}}
36.105 \declare@shorthand{catalan}{"u}{\textormath{\@umlaut u}{\ddot u}}
36.106 \declare@shorthand{catalan}{"I}{\textormath{\@umlaut I}{\ddot I}}
36.107 \declare@shorthand{catalan}{"L}{\Lgem{}}
36.108 \declare@shorthand{catalan}{"U}{\textormath{\@umlaut U}{\ddot U}}
    cedille,
36.109 \declare@shorthand{catalan}{"c}{\textormath{\c c}{~{\prime} c}}
36.110 \declare@shorthand{catalan}{"C}{\textormath{\c C}{^{\prime} C}}
    'french' quote characters,
36.111 \declare@shorthand{catalan}{"<}{%
36.112 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
36.113 \declare@shorthand{catalan}{">}{%
36.114 \textormath{\guillemotright}{\mbox{\guillemotright}}}
    grave accents,
36.115 \@ifpackagewith{babel}{activegrave}{%
36.116 \declare@shorthand{catalan}{'a}{\textormath{\@grave a}{\grave a}}
36.117 \declare@shorthand{catalan}{'e}{\textormath{\@grave e}{\\grave e}}
36.118 \declare@shorthand{catalan}{`o}{\textormath{\@grave o}{\\grave o}}
36.119 \declare@shorthand{catalan}{`A}{\textormath{\@grave A}{\grave A}}
36.120 \declare@shorthand{catalan}{`E}{\textormath{\@grave E}{\\grave E}}
36.121 \declare@shorthand{catalan}{`O}{\textormath{\@grave 0}{\grave 0}}
36.122 \declare@shorthand{catalan}{'`}{\textquotedblleft}%','
36.123 }{}
```

acute accents,
36.124 \@ifpackagewith\{babel\}\{activeacute\}\{\%
36.125 \declare@shorthand\{catalan\}\{'a\}\{\textormath\{\@acute a\}\{~\{\prime\} a\}\}
36.126 \declare@shorthand\{catalan\}\{'e\}\{\textormath\{\@acute e\}\{~\{\prime\} e\}\}
36.127 \declare@shorthand\{catalan\}\{'i\}\{\textormath\{\@acute\i\{\}\}\{~\{\prime\} i\}\}
36.128 \declare@shorthand\{catalan\}\{'o\}\{\textormath\{\@acute o\}\{~\{\prime\} o\}\}
36.129 \declare@shorthand\{catalan\}\{'u\}\{\textormath\{\@acute u\}\{~\{\prime\} u\}\}
$36.130 \backslash$ declare@shorthand\{catalan\}\{'A\}\{\textormath\{\@acute A\}\{~\{\prime\} A\}\}
36.131 \declare@shorthand\{catalan\}\{'E\}\{\textormath\{\@acute E\}\{~\{\prime\} E\}\}
36.132 \declare@shorthand\{catalan\}\{'I\}\{\textormath\{\@acute I\}\{~\{\prime\} I\}\}
36.133 \declare@shorthand\{catalan\}\{'O\}\{\textormath\{\@acute 0\}\{~\{\prime\} 0\}\}
36.134 \declare@shorthand\{catalan\}\{'U\}\{\textormath\{\@acute U\}\{~\{\prime\} U\}\}
36.135 \declare@shorthand\{catalan\}\{'|\}\{\%
36.136 \textormath\{\csname normal@char\string'\endcsname\}\{~\{\prime\}\}\}
the acute accent,
36.137 \declare@shorthand\{catalan\}\{', \}\{\%
36.138 \textormath\{”\}\{\sp\bgroup\prim@s'\}\}
36.139 \}\{\}
and finally, some support definitions
36.140 \declare@shorthand\{catalan\}\{"-\}\{\nobreak-\bbl@allowhyphens\}
36.141 \declare@shorthand\{catalan\}\{"|\}\{\%
36.142 \textormath\{\nobreak\discretionary\{-\}\{\}\{\kern.03em\}\%
36.143 \allowhyphens\}\{\}\}
\- All that is left now is the redefinition of $\backslash$-. The new version of $\backslash$ - should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ can generate from the hyphenation patterns. However, the average length of words in Catalan makes this desirable and so it is kept here.

```
36.144 \addto\extrascatalan{%
36.145 \babel@save{\-}%
36.146 \def\-{\bbl@allowhyphens\discretionary{-}{}{}\bbl@allowhyphens}}
```

\lgem Here we define a macro for typing the catalan "ela geminada" (geminated l). The
\Lgem macros \lgem and \Lgem have been chosen for its lowercase and uppercase representation, respectively ${ }^{36}$.

The code used in the actual macro used is a combination of the one proposed by Feruglio and Fuster ${ }^{37}$ and the proposal ${ }^{38}$ from Valiente presented at the $T_{E} X$ Users Group Annual Meeting in 1995. This last proposal has not been fully implemented due to its limitation to CM fonts.

```
36.147\newdimen\leftllkern \newdimen\rightllkern \newdimen\raiselldim
36.148 \def\lgem{%
36.149 \ifmmode
36.150 \csname normal@char\string"\endcsname 1%
36.151 \else
    \leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
    \setbox0\hbox{l}\setbox1\hbox{l\/}\setbox2\hbox{.}%
    \advance\raiselldim by \the\fontdimen5\the\font
    \advance\raiselldim by -\ht2%
    \leftllkern=-.25\wd0%
    \advance\leftllkern by \wd1%
    \advance\leftllkern by -\wd0%
    \rightllkern=-.25\wd0%
    \advance\rightllkern by -\wd1%
    \advance\rightllkern by \wd0%
    \allowhyphens\discretionary{l-}{l}%
    {\hbox{l}\kern\leftllkern\raise\raiselldim\hbox{.}%
                \kern\rightllkern\hbox{l}}\allowhyphens
    \fi
    }
        \def\Lgem{%
        \ifmmode
            \csname normal@char\string"\endcsname L%
        \else
            \leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
```

[^31]```
36.172\\setbox0\hbox{L}\setbox1\hbox{L\/}\setbox2\hbox{.}%
36.173 \advance\raiselldim by .5\ht0%
36.174 \advance\raiselldim by -.5\ht2%
36.175 \leftllkern=-.125\wd0%
36.176 \advance\leftllkern by \wd1%
36.177 \advance\leftllkern by -\wd0%
36.178 \rightllkern=-\wd0%
36.179 \divide\rightllkern by 6%
36.180 \advance\rightllkern by -\wd1%
36.181 \advance\rightllkern by \wd0%
36.182 \allowhyphens\discretionary{L-}{L}%
36.183 {\hbox{L}\kern\leftllkern\raise\raiselldim\hbox{.}%
36.184\\kern\rightllkern\hbox{L}}\allowhyphens
36.185 \fi
36.186 }
```

\1.1 It seems to be the most natural way of entering the "ela geminda" to use the \L.L sequences \l.l and \L.L. These are not really macro's by themselves but the macros \I 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'.

```
36.187\AtBeginDocument{%
36.188 \let\lslash\l
36.189 \let\Lslash\L
36.190 \DeclareRobustCommand\l{\@ifnextchar.\bbl@l\lslash}
36.191 \DeclareRobustCommand\L{\@ifnextchar.\bbl@L\Lslash}}
36.192\def\bbl@l#1#2{\lgem}
36.193\def\bbl@L#1#2{\Lgem}
```

\up A macro for typesetting things like $1^{\mathrm{er}}$ as proposed by Raymon Seroul ${ }^{39}$.
36.194 \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.
36.195 \ldf@finish\{catalan\}
36.196 〈/code〉

[^32]
## 37 The Galician language

The file galician． $\mathrm{dtx}^{40}$ defines all the language definition macros for the Galician language．

For this language the characters,$\sim$ and＂are made active．In table 12 an overview is given of their purpose．These active accents character behave according
＂｜disable ligature at this position．
＂－an explicit hyphen sign，allowing hyphenation in the rest of the word．
\－like the old $\backslash$－，but allowing hyphenation in the rest of the word．
＇a an accent that allows hyphenation．Valid for all vow－ els uppercase and lowercase．
＇ n a n with a tilde．This is included to improve com－ patibility with FTC．Works for uppercase too．
＂u a u with dieresis allowing hyphenation．
＂a feminine ordinal as in 1a．
＂。 masculine ordinal as in 1ㅇ．
～n a n with tilde．Works for uppercase too．
Table 12：The extra definitions made by galician．ldf
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{galician}\captionsgalician
```

When this file is read as an option，i．e．by the gecommand，galiciancouldbean＇unknown＇languageinwhichcasewehavetomakeitknown．Sowecheckfortheexistenceof\1＠galiciantoseewhetherwehavetodosomethinghere．undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
37.3\ifx\l@galician\@undefined
37.4 \@nopatterns{Galician}
37.5 \adddialect\l@galician0\fi
```

The next step consists of defining commands to switch to（and from）the Gali－ cian language．
\captionsgalician The macro \captionsgalician defines all strings used in the four standard doc－ umentclasses provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ ．

```
37.6 \addto\captionsgalician\{\%
\(37.7 \backslash\) def \(\backslash p r e f a c e n a m e\{P r e f a c i o\} \%\)
37.8 \def \(\backslash\) refname\{Referencias\}\%
```

[^33]```
37.9 \def\abstractname{Resumo}%
37.10 \def\bibname{Bibliograf\'{\i}a}%
37.11 \def\chaptername{Cap\'{\i}tulo}%
37.12 \def\appendixname{Ap\'endice}%
37.13\def\contentsname{\'Indice Xeral}%
37.14\def\listfigurename{\'Indice de Figuras}%
37.15\def\listtablename{\'Indice de T\'aboas}%
37.16 \def\indexname{\'Indice de Materias}%
37.17 \def\figurename{Figura}%
37.18 \def\tablename{T\'aboa}%
37.19 \def\partname{Parte}%
37.20 \def\enclname{Adxunto}%
37.21 \def\ccname{Copia a}%
37.22 \def\headtoname{A}%
37.23 \def\pagename{P\'axina}%
37.24\def\seename{v\'exase}%
37.25 \def\alsoname{v\'exase tam\'en}%
37.26 \def\proofname{Demostraci\'on}%
37.27 \def\glossaryname{Glosario}%
37.28 }
```

\dategalician The macro \dategalician redefines the command \today to produce Galician dates.

```
37.29\def\dategalician{%
37.30 \def\today{\number\day ~de\space\ifcase\month\or
37.31 xaneiro\or febreiro\or marzo\or abril\or maio\or xu\ ~no\or
37.32 xullo\or agosto\or setembro\or outubro\or novembro\or decembro\fi
37.33 \space de~\number\year}}
```

\extrasgalician The macro \extrasgalician will perform all the extra definitions needed for the \noextrasgalician Galician language. The macro \noextrasgalician is used to cancel the actions of \extrasgalician.

For Galician, some characters are made active or are redefined. In particular, the " character and the ~ character receive new meanings this can also happen for the ' character when the option activeacute is specified.

```
37.34 \addto\extrasgalician{\languageshorthands{galician}}
37.35 \initiate@active@char{"}
37.36 \initiate@active@char{~}
37.37 \addto\extrasgalician{%
37.38 \bbl@activate{"}\bbl@activate{~}}
37.39 \@ifpackagewith{babel}{activeacute}{%
37.40 \initiate@active@char{'}}{}
37.41 \@ifpackagewith{babel}{activeacute}{%
37.42 \addto\extrasgalician{\bbl@activate{'}}}{}
```

Now make sure that the characters that have been turned into shorthanfd characters expand to 'normal' characters outside the catalan environment.

```
37.43\addto\noextrasgalician{%
37.44 \bbl@deactivate{"}\bbl@deactivate{~}}
```

```
37.45\@ifpackagewith{babel}{activeacute}{%
37.46 \addto\noextrascatalan{\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.

```
37.47\addto\extrasgalician{%
37.48\babel@save\"\babel@save\~%
37.49 \def\"{\protect\@umlaut}%
37.50\\def\~{\protect\@tilde}}
37.51 \@ifpackagewith{babel}{activeacute}{%
37.52 \babel@save\'%
37.53\addto\extrasgalician{\def\'{\protect\@acute}}
37.54 }{}
```

All the code above is necessary because we need a few extra active characters. These characters are then used as indicated in table 12.

This option includes some support for working with extended, 8 -bit fonts, if available. This assumes that the user has some macros predefined. For instance, if the user has a \@ac@a macro defined, the sequence \'a or 'a will both expand to whatever \@ac@a is defined to expand, presumably á. The names of these macros are the same as those in Ferguson's ML- $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ compatibility package on purpose. Using this method, and provided that adequate hyphenation patterns exist, it is possible to get better hyphenation for Galician than before. If the user has a terminal able to produce these codes directly, it is possible to do so. If the need arises to send the document to someone who does not have such support, it is possible to mechanically translate the document so that the receiver can make use of it.

To be able to define the function of the new accents, we first define a couple of 'support' macros.
\dieresis The original definition of $\backslash "$ is stored as \dieresis, because the definition of \textacute \" might not be the default plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ one. If the user uses PostScript fonts 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 \accent"7F \#1, but as \accent'310 \#1. Something similar happens when using fonts that follow the Cork encoding. For this reason we save the definition of $\backslash "$ and use that in the definition of other macros. We do likewise for \' and <br>~.

```
37.55 \let\dieresis\"
37.56 \let\texttilde\~
37.57\@ifpackagewith{babel}{activeacute}{\let\textacute\'}{}
```

\@umlaut
\@acute
\@tilde

If the user setup has extended fonts, the Ferguson macros are required to be defined. We check for their existance and, if defined, expand to whatever they are defined to. For instance, \'a would check for the existance 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- $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ because the hyphenation algorithm
can work on the whole word. The following macros are directly derived from ML-TEX. ${ }^{41}$
37.58 \def \@umlaut\#1\{\allowhyphens \dieresis\{\#1\}\allowhyphens\}
37.59 \def \@tilde\#1\{\allowhyphens \texttilde\{\#1\}\allowhyphens\}
37.60 \@ifpackagewith\{babel\}\{activeacute\}\{\%
$37.61 \backslash$ def $\backslash @$ @cute\#1\{\allowhyphens $\backslash$ textacute\{\#1\}\allowhyphens $\}\}\}$
Now we can define our shorthands: the umlauts,
37.62 \declare@shorthand\{galician\}\{"-\}\{\nobreak-\bbl@allowhyphens\}
37.63 \declare@shorthand\{galician\}\{"|\}\{\discretionary\{-\}\{\}\{\kern.03em\}\}
37.64 \declare@shorthand\{galician\}\{"u\}\{\@umlaut\{u\}\}
37.65 \declare@shorthand\{galician\}\{"U\}\{\@umlaut\{U\}\}
ordinals ${ }^{42}$,
37.66 \declare@shorthand\{galician\}\{"o\}\{\%
37.67 \leavevmode\raise1ex\hbox\{\underbar\{\scriptsize o\}\}\}
37.68 \declare@shorthand\{galician\}\{"a\}\{\%
37.69 \leavevmode\raise1ex \hbox\{\underbar\{\scriptsize a\}\}\}
acute accents,
37.70 \@ifpackagewith\{babel\}\{activeacute\}\{\%
37.71 \declare@shorthand\{galician\}\{'a\}\{\textormath\{\@acute a\}\{~\{\prime\} a\}\}
37.72 \declare@shorthand\{galician\}\{'e\}\{\textormath\{\@acute e\}\{^\{\prime\} e\}\}
37.73 \declare@shorthand\{galician\}\{'i\}\{\textormath\{\@acute\i\{\}\}\{^\{\prime\}i\}\}
37.74 \declare@shorthand\{galician\}\{'o\}\{\textormath\{\@acute 0$\}\{\wedge\{\backslash$ prime 0 o\}\}
37.75 \declare@shorthand\{galician\}\{'u\}\{\textormath\{\@acute u\}\{^\{\prime\} u\}\}
37.76 \declare@shorthand\{galician\}\{'A\}\{\textormath\{\@acute A\}\{^\{\prime\} A\}\}
37.77 \declare@shorthand\{galician\}\{'E\}\{\textormath\{\@acute E\}\{~\{\prime\} E\}\}
37.78 \declare@shorthand\{galician\}\{'I\}\{\textormath\{\@acute I\}\{^\{\prime\} I\}\}
37.79 \declare@shorthand\{galician\}\{'O\}\{\textormath\{\@acute 0\}\{^\{\prime\} 0\}\}
37.80 \declare@shorthand\{galician\}\{'U\}\{\textormath\{\@acute U\}\{^\{\prime\} U\}\}
tildes,
37.81 \declare@shorthand\{galician\}\{'n\}\{\textormath\{\~n\}\{~\{\prime\} n\}\}
37.82 \declare@shorthand\{galician\}\{’N\}\{\textormath $\{\sim \sim N\}\{\sim\{$ prime $\} \mathrm{N}\}\}$
the acute accent,
37.83 \declare@shorthand\{galician\}\{', \}\{\%
37.84 \textormath\{”\}\{\sp\bgroup\prim@s'\}\}
37.85 \}\{\}
$37.86 \backslash$ declare@shorthand\{galician\}\{~n\}\{\textormath\{\~n\}\{\@@tilde n\}\}
$37.87 \backslash$ declare@shorthand\{galician\}\{~N\}\{\textormath $\{\backslash \sim \mathrm{N}\}\{\backslash @$ ©ilde N$\}\}$

[^34]\- All that is left now is the redefinition of $\backslash$-. The new version of $\backslash$ - should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ in this respect is unfortunate for Galician 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ can generate from the hyphenation patterns. However, the average length of words in Galician makes this desirable and so it is kept here.

```
37.88\addto\extrasgalician{%
37.89 \babel@save{\-}%
37.90 \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.

```
37.91 \ldf@finish{galician}
37.92 \/code〉
```


## 38 The Basque language

The file basque. $\mathrm{dtx}^{43}$ defines all the language definition macro's for the Basque language.

For this language the characters $\sim$ and " are made active. In table 13 an overview is given of their purpose. These active accent characters behave according
" | disable ligature at this position.
" - an explicit hyphen sign, allowing hyphenation in the rest of the word.
\- like the old $\backslash-$, but allowing hyphenation in the rest of the word.
" $<$ for French left double quotes (similar to $\ll$ ).
" $>$ for French right double quotes (similar to $\gg$ ).
${ }^{\sim} \mathrm{n}$ a n with tilde. Works for uppercase too.
Table 13: 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 }2_{\varepsilon}\)or$\operatorname{NFSS}2$.IfT1encodingischosen,andprovidedthatadequatehyphenationpatterns${}^{44}$areavailable.Theeasiestwaytousethenewencodingwith$\mathrm{IAT}_{\mathrm{E}}2_{\varepsilon}$istoloadthepackaget1encwith\usepackage.Thismustbedonebeforeloadingbabel.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

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

```
38.1 〈*code>
38.2 \LdfInit{basque}\captionsbasque
```

When this file is read as an option, i.e. by the command,basquecouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof$\backslash1@basque~to~see~whether~we~have~to~do~something~here.~$undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
38.3\ifx\l@basque\@undefined
38.4 \@nopatterns{Basque}
38.5 \adddialect\l@basque0
38.6 \fi
```

The next step consists of defining commands to switch to (and from) the Basque language.

[^35]\captionsbasque The macro \captionsbasque defines all strings used in the four standard documentclasses provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

| 38. | dto\captionsbasque\{\% |
| :---: | :---: |
| 38.8 | \def \prefacename\{Hitzaurrea\}\% |
| 38.9 | \def \refname\{Erreferentziak\}\% |
| 38.10 | \def \abstractname\{Laburpena\}\% |
| 38.11 | \def \bibname\{Bibliografia\}\% |
| 38.12 | \def \chaptername\{Kapitulua\}\% |
| 38.13 | \def \appendixname\{Eranskina\}\% |
| 38.14 | \def \contentsname\{Gaien Aurkibidea\}\% |
| 38.15 | \def\listfigurename\{Irudien Zerrenda\}\% |
| 38.16 | \def \listtablename\{Taulen Zerrenda\}\% |
| 38.17 | \def \indexname\{Kontzeptuen Aurkibidea\}\% |
| 38.18 | \def $\backslash$ figurename\{Irudia\}\% |
| 38.19 | \def\tablename\{Taula\}\% |
| 38.20 | \def\partname\{Atala\}\% |
| 38.21 | \def \enclname\{Erantsia\}\% |
| 38.22 | \def \ccname\{Kopia\}\% |
| 38.23 | \def \headtoname\{Nori\}\% |
| 38.24 | \def\pagename\{0rria\}\% |
| 38.25 | \def\seename\{Ikusi\}\% |
| 38.26 | \def\alsoname\{Ikusi, halaber\}\% |
| 38.27 | \def \proofname\{Frogapena\}\% |
| 38.28 | \def \glossaryname\{Glosarioa\}\% |
| 38.29 | 3\% |

\datebasque The macro \datebasque redefines the command \today to produce Basque 38.30 \def $\backslash$ datebasque\{\%
38.31 \def \today\{\number $\backslash$ year.eko\space $\backslash i f c a s e \backslash m o n t h \backslash o r$
38.32 urtarrilaren\or otsailaren\or martxoaren\or apirilaren\or
38.33 maiatzaren\or ekainaren\or uztailaren\or abuztuaren\or
38.34 irailaren\or urriaren\or azaroaren\or
38.35 abenduaren $\backslash f i \sim$ ~number $\backslash$ day $\}$ \}
\extrasbasque The macro \extrasbasque will perform all the extra definitions needed for the \noextrasbasque 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.

```
38.36 \addto\extrasbasque{\languageshorthands{basque}}
38.37 \initiate@active@char{"}
38.38 \initiate@active@char{~}
38.39 \addto\extrasbasque{%
38.40 \bbl@activate{"}%
38.41 \bbl@activate{~}}
```

Don't forget to turn the shorthands off again.
38.42 \addto\noextrasbasque\{
38.43 \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.

```
38.44 \addto\extrasbasque{%
38.45 \babel@save\"%
38.46 \babel@save\~%
38.47 \def\"{\protect\@umlaut}%
38.48\\def\~{\protect\@tilde}}
```

\basquehyphenmins Basque hyphenation uses \lefthyphenmin and \righthyphenmin both set to 2 . 38.49 \providehyphenmins\{\CurrentOption\}\{\tw@\tw@\}
\dieresia The original definition of \" is stored as \dieresia, because the we do not know \texttilde 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ \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 $\backslash "$ and use that in the definition of other macros. We do likewise for $\backslash^{\prime}$ and $\backslash^{\sim}$. 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.

```
38.50 \let\dieresia\"
```

38.51 \let $\backslash t e x t t i l d e \backslash ~$
\@umlaut We check the encoding and if not using T1, we make the accents expand but 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$. 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 T 1 is in use. If not, we will give a warning and proceed redefining the accent macros so that $\mathrm{TEX}_{\mathrm{E}}$ at least finds the breaks that are not too close to the accent. The warning will only be printed to the $\log$ file.

```
\ifx\DeclareFontShape\@undefined
\wlog{Warning: You are using an old LaTeX}
    \wlog{Some word breaks will not be found.}
    \def\@umlaut#1{\allowhyphens\dieresia{#1}\allowhyphens}
\def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
\else
    \edef\bbl@next{T1}
    \ifx\f@encoding\bbl@next
            \let\@umlaut\dieresia
            \let\@tilde\texttilde
    \else
            \wlog{Warning: You are using encoding \f@encoding\space
                instead of T1.}
            \wlog{Some word breaks will not be found.}
            \def\@umlaut#1{\allowhyphens\dieresia{#1}\allowhyphens}
```

```
38.67\\def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
38.68 \fi
38.69\fi
```

Now we can define our shorthands: the french quotes,

```
38.70\declare@shorthand{basque}{"<}{%
```

38.71 \textormath\{\guillemotleft\}\{\mbox\{\guillemotleft\}\}\}
38.72 \declare@shorthand\{basque\}\{">\}\{\%
38.73 \textormath\{\guillemotright\}\{\mbox\{\guillemotright\}\}\}
ordinals ${ }^{45}$,
38.74 \declare@shorthand\{basque\}\{' '\}\{\%
38.75 \textormath\{”\}\{\sp\bgroup\prim@s'\}\}
tildes,
$38.76 \backslash$ declare@shorthand\{basque\} $\{\sim n\}\{\backslash$ textormath $\{\backslash \sim n\}\{\backslash @$ tilde $n\}\}$
$38.77 \backslash$ declare@shorthand\{basque\}\{~N\}\{\textormath\{ ${ }^{\sim}$ N $\left.\}\{\backslash @ t i l d e ~ N\}\right\}$
and some additional commands.
The shorthand " - should be used in places where a word contains an explictit 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.

```
38.78 \declare@shorthand{basque}{"-}{%
38.79 \nobreak\discretionary{-}{-}{-}\bbl@allowhyphens}
38.80\declare@shorthand{basque}{"|}{%
38.81 \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
38.82
\allowhyphens}{}}
```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of @ to its original value.
$38.83 \backslash l d f @ f$ inish\{basque \}
38.84 〈/code〉

[^36]
## 39 The Romanian language

The file romanian. $\mathrm{dtx}^{46}$ 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.

```
39.1 〈*code>
39.2\LdfInit{romanian}\captionsromanian
```

When this file is read as an option, i.e. by the command,romanianwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@romaniantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
39.3\ifx\l@romanian\@undefined
39.4 \@nopatterns{Romanian}
39.5 \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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
39.6 ls
39.7 \addto\captionsromanian{%
39.8 \def\prefacename{Prefa\c{t}\u{a}}%
39.9\def\refname{Bibliografie}%
39.10\def\abstractname{Rezumat}%
39.11 \def\bibname{Bibliografie}%
39.12 \def\chaptername{Capitolul}%
9.13\def\appendixname{Anexa}%
39.14 \def\contentsname{Cuprins}%
39.15 \def\listfigurename{List\u{a} de figuri}%
39.16 \def\listtablename{List\u{a} de tabele}%
39.17 \def\indexname{Glosar}%
39.18 \def\figurename{Figura}% % sau Plan\c{s}a
9.19 \def\tablename{Tabela}%
3.20\\def\partname{Partea}%
39.21\def\enclname{Anex\u{a}}% % sau Anexe
39.22\\def\ccname{Copie}%
39.23\def\headtoname{Pentru}%
39.24\def\pagename{Pagina}%
39.25\def\seename{Vezi}%
39.26 \def\alsoname{Vezi de asemenea}%
39.27 \def\proofname{Demonstra\c{t}ie} %
39.28\def\glossaryname{Glosar}%
39.29 }%
```

[^37] A contribution was made by Umstatter Horst (hhu@cernvm. cern.ch).
\dateromanian The macro \dateromanian redefines the command \today to produce Romanian dates.

```
39.30\def\dateromanian{%
39.31 \def\today{\number\day~\ifcase\month\or
39.32 ianuarie\or februarie\or martie\or aprilie\or mai\or
39.33 iunie\or iulie\or august\or septembrie\or octombrie\or
39.34 noiembrie\or decembrie\fi
39.35 \space \number\year}}
```

\extrasromanian The macro \extrasromanian will perform all the extra definitions needed for the \noextrasromanian Romanian language. The macro \noextrasromanian is used to cancel the actions of \extrasromanian For the moment these macros are empty but they are defined for compatibility with the other language definition files.
39.36 \addto \extrasromanian\{\}
39.37 \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.
$39.38 \backslash l d f @ f i n i s h\{r o m a n i a n\}$
39.39 〈/code〉

## 40 The Danish language

The file danish.dtx ${ }^{47}$ defines all the language definition macros for the Danish language.

For this language the character " is made active. In table 14 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 $\ll$ ).
"> for French right double quotes (similar to $\gg$ ).
Table 14: 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.

```
40.1 <*code>
40.2\LdfInit{danish}\captionsdanish
```

When this file is read as an option, i.e. by the command,danishwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@danishtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
40.3\ifx\l@danish\@undefined
40.4 \@nopatterns{Danish}
40.5 \adddialect\l@danish0\fi
```

\englishhyphenmins This macro is used to store the correct values of the hyphenation parameters $\backslash$ lefthyphenmin and \righthyphenmin.
40.6 \providehyphenmins\{\CurrentOption\}\{\@ne\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 $\mathrm{HAT}_{\mathrm{E}} \mathrm{X}$.

```
40.7 \addto\captionsdanish{%
40.8 \def\prefacename{Forord}%
40.9\def\refname{Litteratur}%
40.10\\def\abstractname{Resum\'e}%
40.11\def\bibname{Litteratur}%
```

[^38]```
40.12\def\chaptername{Kapitel}%
40.13\def\appendixname{Bilag}%
40.14 \def\contentsname{Indhold}%
40.15 \def\listfigurename{Figurer}%
40.16 \def\listtablename{Tabeller}%
40.17 \def\indexname{Indeks}%
40.18\def\figurename{Figur}%
40.19 \def\tablename{Tabel}%
40.20 \def\partname{Del}%
40.21 \def\enclname{Vedlagt}%
40.22 \def\ccname{Kopi til}% or Kopi sendt til
40.23 \def\headtoname{Til}% in letter
40.24\def\pagename{Side}%
40.25\def\seename{Se}%
40.26 \def\alsoname{Se ogs{\aa}}%
40.27 \def\proofname{Bevis}%
40.28\\def\glossaryname{Gloseliste}%
40.29 }%
```

\datedanish The macro \datedanish redefines the command \today to produce Danish dates.

```
40.30\def\datedanish{%
40.31 \def\today{\number\day. ~\ifcase\month\or
40.32 januar\or februar\or marts\or april\or maj\or juni\or
40.33 juli\or august\or september\or oktober\or november\or december\fi
40.34 \space\number\year}}
```

\extrasdanish The macro \extrasdanish will perform all the extra definitions needed for the \noextrasdanish Danish language. The macro \noextrasdanish is used to cancel the actions of \extrasdanish.

Danish typesetting requires $\backslash$ frenchspacing to be in effect.
40.35 \addto \extrasdanish\{ \bbl@frenchspacing\}
40.36 \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.

```
40.37 \initiate@active@char{"}
40.38 \addto\extrasdanish{\languageshorthands{danish}}
40.39 \addto\extrasdanish{\bbl@activate{"}}
```

Don't forget to turn the shorthands off again.
40.40 \addto\noextrasdanish\{\bbl@deactivate\{"\}\}

First we define access to the low opening double quote and guillemets for quotations,
40.41 \declare@shorthand\{danish\}\{"'\}\{\%
40.42 \textormath\{\quotedblbase\}\{\mbox\{\quotedblbase\}\}\}
40.43 \declare@shorthand\{danish\}\{"'\}\{\%

```
40.44 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
40.45 \declare@shorthand{danish}{"<}{%
40.46 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
40.47 \declare@shorthand{danish}{">}{%
40.48 \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

then we define to be able to specify hyphenation breakpoints that behave a little different from \-.
40.49 \declare@shorthand\{danish\}\{"-\}\{\nobreak-\bbl@allowhyphens\}
40.50 \declare@shorthand\{danish\}\{""\}\{\hskip\z@skip\}
40.51 \declare@shorthand\{danish\}\{"~\}\{\textormath\{\leavevmode $\backslash$ hbox $\{-\}\}\{-\}\}$
40.52 \declare@shorthand\{danish\}\{"=\}\{\nobreak-\hskip\z@skip\}

And we want to have a shorthand for disabling a ligature.
40.53 \declare@shorthand\{danish\}\{"|\}\{\%
40.54 \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 $\backslash$ begin\{document\} and resetting the category code of © to its original value.
$40.55 \backslash l d f @ f i n i s h\{d a n i s h\}$
40.56 〈/code〉

## 41 The Icelandic language

### 41.1 Overview

The file iceland. $\mathrm{dtx}^{48}$ 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 15 an overview is given of its purpose. The shorthands in table 15 can also be typeset by using the

| " | disable ligature at this pos |
| :---: | :---: |
| " - | an explicit hyphen sign, allowing hyphenation in the rest of the word. |
| " " | like " - , but producing no hyphen sign (for compund 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 $\gg)$. |
| "< | for Icelandic 'french' right double quotes (similar to $\ll$ ). |
| "。 | for old Icelandic 9 |
| "0 | for old Icelandic Q |
| "ó | for old Icelandic ${ }^{\text {Q }}$ |
| "0́ | for old Icelandic Q́ |
| "e | for old Icelandic e |
| "E | for old Icelandic $\mathrm{E}_{d}$ |
| "é | for old Icelandic é |
| "É | for old Icelandic $\underline{E}_{E}$ |
| \tala | for typesetting numbers |
| $\backslash \mathrm{grada}$ | for the 'degree' symbol |
| \gradur | for 'degrees', e.g. $5{ }^{\circ} \mathrm{C}$ |
| \upp | for textsuperscript |

Table 15: The shorthands and extra definitions made by icelandic.ldf
commands in table 16.

[^39]```
\ilqq for Icelandic left double quotes (looks like „).
\irqq for Icelandic right double quotes (looks like").
\ilq for Icelandic left single quotes (looks like,).
\irq for Icelandic right single quotes (looks like ').
\iflqq for Icelandic 'french' left double quotes (similar to
        >>).
\ifrqq for Icelandic 'french' right double quotes (similar to
        <<).
\ifrq for Icelandic 'french' right single quotes (similar to
        <).
\iflq for Icelandic 'french' left single quotes (similar to >).
\dq the original quotes character (").
\oob for old Icelandic Q
\Oob for old Icelandic Q
\ooob for old Icelandic ó
\00ob for old Icelandic Q́
\eob for old Icelandic e&
\Eob for old Icelandic E
\eeob for old Icelandic é
\EEob for old Icelandic É
```

Table 16: Commands which produce quotes and old Icelandic diacritics, defined by icelandic.ldf

## References

[1] Alpingi. Reglur um frágang pingskjala og prentun umræððna, 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.

### 41.2 TEXnical details

When this file was read through the option icelandic we make it behave as if icelandic was specified.

```
41.2 \ifx\CurrentOption\bbl@tempa
41.3 \def\CurrentOption{icelandic}
41.4 \fi
```

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

```
41.5 <*code>
41.6\LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e., by the command,icelandicwillbean'unknown'language,sowehavetomakeitknown.Sowehere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
41.7\ifx\l@icelandic\@undefined
41.8 \@nopatterns{Icelandic}
41.9 \adddialect\l@icelandic0
41.10\fi
```

\if@Two@E We will need a new 'if': \if@Two@E is true if and only if $\mathrm{ET}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ 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 $\backslash$ if in $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$-2.09 based formats.

```
41.11 \newif \if@Two@E \@Two@Etrue
\(41.12 \backslash \operatorname{def} \backslash @ F I @\{\backslash f i\}\)
41.13 \ifx\@compatibilitytrue\@undefined
\(41.14 \backslash @ T w o @ E f a l s e \backslash d e f \backslash @ F I @\{\) \relax\}
41.15 \else
41.16 \if@compatibility \@Two@Efalse \fi
41.17 \@FI@
```

\extrasicelandic The macro \extrasicelandic will perform all the extra definitions needed for the \noextrasicelandic 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.

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

41.19 \@namedef\{extras \CurrentOption\}\{\%
41.20 \languageshorthands\{icelandic\}\}
41.21 \expandafter\addto\csname extras\CurrentOption\endcsname\{\%
41.22 \bbl@activate\{"\}\}

Don't forget to turn the shorthands off again.
41.23 \addto\noextrasicelandic \{\bbl@deactivate\{"\}\}

The icelandic hyphenation patterns can be used with \lefthyphenmin and $\backslash$ righthyphenmin set to 2 .
$41.24 \backslash$ 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 16.

To be able to define the function of ", we first define a couple of 'support' macros.

### 41.3 Captionnames and date

The next step consists of defining the Icelandic equivalents for the $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ captionnames.
\captionsicelandic The macro \captionsicelandic will define all strings used used in the four standard document classes provided with $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
41.25\@namedef{captions\CurrentOption}{%
41.26 \def\prefacename{Form\'{a}li}%
41.27\def\refname{Heimildir}%
41.28\\def\abstractname{\'{U}tdr\'{a}ttur}%
41.29 \def\bibname{Heimildir}%
41.30\\def\chaptername{Kafli}%
41.31 \def\appendixname{Vi{\dh}auki}%
41.32 \def\contentsname{Efnisyfirlit}%
41.33\def\listfigurename{Myndaskr\'{a}}%
41.34 \def\listtablename{T\"{o}fluskr\'{a}}%
41.35 \def\indexname{Atri{\dh}isor{\dh}askr\'{a}}%
41.36\\def\figurename{Mynd}%
41.37\def\tablename{Tafla}%
41.38 \def\partname{Hluti}%
41.39\def\enclname{Hj\'{a}lagt}%
41.40 \def\ccname{Samrit}%
41.41 \def\headtoname{Til:}% in letter
41.42\\ef\pagename{Bla{\dh}s\'{\i}{\dh}a}%
41.43\def\seename{Sj\'{a}}%
41.44\def\alsoname{Sj\'{a} einnig}%
41.45\def\proofname{S\"{o}nnun}%
41.46 \def\glossaryname{Or{\dh}alisti}%
41.47 }
```

\dateicelandic The macro \dateicelandic redefines the command \today to produce Icelandic dates.

```
41.48\def\dateicelandic{%
41.49 \def\today{\number\day. ~\ifcase\month\or
41.50 jan\'{u}ar\or febr\'{u}ar\or mars\or apr\'{\i}l\or ma\'{\i}\or
41.51 j\'{u}n\'{\i}\or j\'{u}l\'{\i}\or \'{a}g\'{u}st\or september\or
41.52 okt\'{o}ber\or n\'{o}vember\or desember\fi
41.53 \space\number\year}}
```


### 41.4 Icelandic quotation marks

$\backslash d q$ We save the original double quote character in \dq to keep it available, the math accent \" can now be typed as ".

```
41.54 \begingroup \catcode`\"12
```

```
41.55\def \x{\endgroup
41.56\def\@SS{\mathchar"7019 }
41.57 \def\dq{"}}
41.58 \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.

```
41.59\let\ilq\grq
41.60\let\irq\grq
41.61 \let\iflq\frq
41.62\let\ifrq\flq
41.63\let\ilqq\glqq
41.64\let\irqq\grqq
41.65\let\iflqq\frqq
41.66\let\ifrqq\flqq
41.67\declare@shorthand{icelandic}{"'}{\glqq}
41.68 \declare@shorthand{icelandic}{"'}{\grqqq}
41.69 \declare@shorthand{icelandic}{">}{\frqq}
41.70 \declare@shorthand{icelandic}{"<}{\flqq}
    and some additional commands:
41.71 \declare@shorthand{icelandic}{"-}{\nobreak\-\bbl@allowhyphens}
41.72 \declare@shorthand{icelandic}{"|}{%
41.73 \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
4 1 . 7 4 ~ \ b b l @ a l l o w h y p h e n s \} \{ \} \}
41.75\declare@shorthand{icelandic}{""}{\hskip\z@skip}
41.76 \declare@shorthand{icelandic}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
41.77 \declare@shorthand{icelandic}{"=}{\nobreak-\hskip\z@skip}
```


### 41.5 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 ' $O$ ', 'e' and ' $E$ ', and 'é' and 'E'. 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.

```
41.78 \newdimen\pl@left
41.79 \newdimen\pl@down
41.80 \newdimen\pl@right
41.81 \newdimen\pl@temp
```

\sob The macro \sob is used to put the 'ogonek' in the right place.

```
41.82 \def\sob#1#2#3#4#5{%parameters: letter and fractions hl,ho,vl,vo
41.83\setbox0\hbox{#1}\setbox1\hbox{\k{}}\setbox2\hbox{p}%
```

```
    41.84 \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
    41.85 \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
    41.86 \pl@left=\pl@right \advance\pl@left by\wd1
    41.87 \pl@temp=-\pl@down \advance\pl@temp by\dp2 \dp1=\pl@temp
    41.88 \leavevmode
    41.89 \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
\oob
\Oob 41.90\DeclareTextCommand{\oob}{T1}{\sob {o}{.85}{0}{.04}{0}}
\ooob41.91 \DeclareTextCommand{\Oob}{T1}{\sob {0}{.7}{0}{0}{0}}
\00ob41.92\DeclareTextCommand{\ooob}{T1}{\sob {ó}{.85}{0}{.04}{0}}
\eob41.93 \DeclareTextCommand{\00ob}{T1}{\sob {D́}{.7}{0}{0}{0}}
\Eob41.94 \DeclareTextCommand{\eob}{T1}{\sob {e}{1}{0}{.04}{0}}
\eeob41.95\DeclareTextCommand{\Eob}{T1}{\sob {E}{1}{0}{.04}{0}}
\EEob 41.96 \DeclareTextCommand{\eeob}{T1}{\sob {é}{1}{0}{.04}{0}}
    41.97\DeclareTextCommand{\EEob}{T1}{\sob {É}{1}{0}{.04}{0}}
    41.98 \declare@shorthand{icelandic}{"o}{\oob}
    41.99 \declare@shorthand{icelandic}{"O}{\Oob}
    41.100 \declare@shorthand{icelandic}{"ó}{\ooob}
    41.101 \declare@shorthand{icelandic}{"O}{\OOob}
    41.102 \declare@shorthand{icelandic}{"e}{\eob}
    41.103 \declare@shorthand{icelandic}{"E}{\Eob}
    41.104 \declare@shorthand{icelandic}{"é}{\eeob}
    41.105\declare@shorthand{icelandic}{"É}{\EEob}
```


### 41.6 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 $1000 \pi$ 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 $1000 \pi$ should be inputed as $\$ 3 \backslash ; 141\{\} ,592 \backslash ; 653 \$$ in math-mode and as something like $3^{\sim} 141,592^{\sim} 653$ in text. Braces are mandatory around the comma in math-mode, the reason is mentioned in the $\mathrm{T}_{\mathrm{E}} \mathrm{Xbook} \mathrm{p}$. 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 $\backslash$ 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 <br>
\tala[,]\{123,4567\} <br>
\tala[,]\{9876,543\}<br>\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 \thousandsep

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 ( $\backslash \mathrm{f}$ @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.

```
41.106\mathchardef\m@comma="013B \def\@comma@{\ifmmode\m@comma\else,\fi}
41.107\def\f@thousandsep{\ifmmode\mskip5.5mu\else\penalty\@M\kern.3em\fi}
41.108 \newcommand{\decimalsep}{.} \newcommand{\thousandsep}{\@comma@}
41.109 \expandafter\addto\csname extras\CurrentOption\endcsname{%
41.110 \def\decimalsep{\@comma@}%
41.111 \def\thousandsep{\f@thousandsep}}
41.112 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
4 1 . 1 1 3 ~ \ d e f \ d e c i m a l s e p \{ . \} \%
41.114 \def\thousandsep{\@comma@}}
```

\tala The decimal separator used when inputing a number with \tala has to be a comma. \tala splits the inputed 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 $\mathrm{LA}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ only) the optional argument of $\backslash$ tala.

```
41.115 \if@Two@E
41.116 \newcommand{\tala}[2] [\decimalsep] {%
41.117 \def\@decimalsep{#1}\@tala#2\@empty,\@empty,\@nil}
41.118\else
41.119 \newcommand{\tala}[1]{%
41.120 \def\@decimalsep{\decimalsep}\@tala#1\@empty,\@empty,\@nil}
41.121 \fi
41.122 \def\@tala#1,#2,#3\@nil{%
41.123 \ifx\@empty#2%
41.124 \@integerpart{#1}%
41.125 \else
41.126 \@integerpart{#1}\@decimalsep\@decimalpart{#2}%
41.127 \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.

```
41.128\def\@decimalpart#1{\@@decimalpart#1\@empty\@empty\@empty}
41.129 \def\@@decimalpart#1#2#3#4{#1#2#3%
41.130 \ifx\@empty#4%
4 1 . 1 3 1 ~ \ e l s e
41.132 \thousandsep\expandafter\@@decimalpart\expandafter#4%
41.133 \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.

```
41.134 \def\@integerpart#1{\@@integerpart{}#1\@empty\@empty\@empty}
```

41.135 \def \@@integerpart\#1\#2\#3\#4\{\%
41.136 \ifx\@empty\#2\%
$41.137 \backslash$ 【addthousandsep\#1\relax
41.138 \else
41.139 \ifx\@empty\#3\%
$41.140 \backslash$ @addthousandsep\@empty\@empty\#1\#2\relax
41.141 \else
41.142 \ifx\@empty\#4\%
$41.143 \backslash$ \addthousandsep\@empty\#1\#2\#3\relax
$41.144 \backslash$ else
41.145 \@@integerpartafterfi\{\#1\#2\#3\#4\}\%
41.146 \fi
41.147 \fi
41.148 \fi\}
41.149 \def \@@integerpartafterfi\#1\fi\fi\fi\{\fi\fi\fi\@@integerpart\{\#1\}\}
41.150 \def \@addthousandsep\#1\#2\#3\#4\{\#1\#2\#3\%
41.151 \if\#4\relax
41.152 \else
41.153 \thousandsep\expandafter\@addthousandsep\expandafter\#4\%
41.154 \fi\}

### 41.7 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.
41.155 \ifx\sevenrm\@undefined
41.156 \ifx\@ptsize\@undefined
41.157 \let\upp@size\small
41.158 \else
41.159 \ifx $\backslash s e l e c t f o n t \backslash @ u n d e f i n e d$

In this case the format is the original $\mathrm{IA}_{\mathrm{E}} \mathrm{X}-2.09$ :

```
41.160 \ifcase\@ptsize
41.161 \let\upp@size\ixpt\or
41.162 \let\upp@size\xpt\or
41.163 \let\upp@size\xipt
41.164 \fi
```

When \selectfont is defined we probably have NFSS available:

```
4 1 . 1 6 5 ~ \ e l s e
41.166 \ifcase\@ptsize
41.167 \def\upp@size{\fontsize\@ixpt{10pt}\selectfont}\or
41.168 \def\upp@size{\fontsize\@xpt{11pt}\selectfont}\or
41.169 \def\upp@size{\fontsize\@xipt{12pt}\selectfont}
41.170 \fi
4 1 . 1 7 1 ~ \ f i
41.172 \fi
41.173\else
If we end up here it must be a plain based TEX format, so:
```

```
41.174 \let\upp@size\sevenrm
```

41.174 \let\upp@size\sevenrm
41.175 \fi

```
41.175 \fi
```

Now we can define \upp. When $\mathrm{EAT}_{\mathrm{E}} 2_{\varepsilon}$ runs in compatibility mode ( $\mathrm{E} \mathrm{T}_{\mathrm{E}} \mathrm{X}-2.09$ emulation), \textsuperscript is also defined, but does no good job, so we give two different definitions for \upp using \if@Two@E.

```
41.176 \if@Two@E
41.177 \DeclareRobustCommand*{\upp}[1]{\textsuperscript{#1}}
41.178 \else
41.179 \DeclareRobustCommand*{\upp}[1] {%
41.180 \leavevmode\raise1ex\hbox{\upp@size#1}}
41.181 \fi
```

Some definitions for special characters. \grada needs a special treatment: it is \char6 in T1-encoding and \char23 in OT1-encoding.
41.182 \ifx $\backslash f m t n a m e \backslash L a T e X e F m t N a m e$
$41.183 \backslash$ DeclareTextSymbol\{\grada\}\{T1\}\{6\}
$41.184 \backslash$ DeclareTextSymbol\{\grada\}\{0T1\}\{23\}
41.185 \else
41.186 \def\T@one\{T1\}
41.187 \ifx\f@encoding\T@one
$41.188 \quad$ newcommand\{\grada\}\{\char6\}
41.189 \else
$41.190 \quad$ newcommand\{\grada\}\{\char23\}
41.191 \fi
41.192 \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 $\backslash$ gradur to 0.3 em ,
this lets the symbol 'degree' stick to the preceding (e.g., $45 \backslash$ gradur) or following character (e.g., 20~\gradur C).
41.193 \DeclareRobustCommand*\{\gradur\}\{\%
41.194 \leavevmode\hbox to $0.3 \mathrm{em}\{\backslash h s s \backslash g r a d a \backslash h s s\}\}$

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.
41.195 \ldf@finish $\backslash$ CurrentOption
41.196 〈/code〉

## 42 The Norwegian language

The file norsk.dtx ${ }^{49}$ 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 17 an overview is given of its purpose.
"ff for ff to be hyphenated as ff-f, this is also implemented for $\mathrm{b}, \mathrm{d}, \mathrm{f}, \mathrm{g}, \mathrm{l}, \mathrm{m}, \mathrm{n}, \mathrm{p}, \mathrm{r}, \mathrm{s}$, and t . (o"ppussing)
"ee Hyphenate "ee as \'e-e. (komit"een)
"- an explicit hyphen sign, allowing hyphenation in the composing words. Use this for compound words when the hyphenation patterns fail to hyphenate properly. (alpin"-anlegg)
" I Like "-, but inserts 0.03em space. Use it if the compound point is spanned by a ligature. (hoff"lintriger)
"" Like "-, but producing no hyphen sign. (i""g\aa\{\}r)
"~ Like -, but allows no hyphenation at all. ( $\mathrm{E}^{\text {~ cup) }}$
" = Like - , but allowing hyphenation in the composing words. (marksistisk"=leninistisk)
"< for French left double quotes (similar to $\ll$ ).
"> for French right double quotes (similar to $\gg$ ).
Table 17: 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 'spissslede'.

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\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e. by the command,norskwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof$\backslash1$@norsktoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
42.3 \ifx \(\backslash 1 @\) norsk \(\ @ u n d e f i n e d\)
42.4 \@nopatterns\{Norsk\}
42.5 \adddialect \(\backslash 1 @\) norsk0 \({ }^{\text {fi }}\)
```

[^40]\norskhyphenmins Some sets of Norwegian hyphenation patterns can be used with \lefthyphenmin
 can't. So we use \lefthyphenmin=2 by default.
$42.6 \backslash$ providehyphenmins $\{\backslash$ 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.

```
42.7\def\bbl@tempa{norsk}
42.8 \ifx\CurrentOption\bbl@tempa
```

The next step consists of defining commands to switch to (and from) the Norwegian language.
\captionsnorsk The macro \captionsnorsk defines all strings used in the four standard documentclasses provided with $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
\def\captionsnorsk{%
    \def\prefacename{Forord}%
    \def\refname{Referanser}%
    \def\abstractname{Sammendrag}%
    \def\bibname{Bibliografi}% or Litteraturoversikt
    % or Litteratur or Referanser
    \def\chaptername{Kapittel}%
    \def\appendixname{Tillegg}% or Appendiks
    \def\contentsname{Innhold}%
    \def\listfigurename{Figurer}% or Figurliste
    \def\listtablename{Tabeller}% or Tabelliste
    \def\indexname{Register}%
    \def\figurename{Figur}%
    \def\tablename{Tabell}%
    \def\partname{Del}%
    \def\enclname{Vedlegg}%
    \def\ccname{Kopi sendt}%
    \def\headtoname{Til}% in letter
    \def\pagename{Side}%
    \def\seename{Se}%
    \def\alsoname{Se ogs\aa{}}%
    \def\proofname{Bevis}%
    \def\glossaryname{Ordliste}%
    }
2.33\else
```

For the 'nynorsk' version of these definitions we just add a "dialect".

```
42.34 \adddialect\l@nynorsk\l@norsk
```

\captionsnynorsk The macro \captionsnynorsk defines all strings used in the four standard documentclasses provided with $\mathrm{LT}_{\mathrm{E}} \mathrm{X}$, but using a different spelling than in the command \captionsnorsk.

```
42.35\def\captionsnynorsk{%
42.36 \def\prefacename{Forord}%
```

```
42.37\\def\refname{Referansar}%
42.38 \def\abstractname{Samandrag}%
42.39\\def\bibname{Litteratur}% or Litteraturoversyn
42.40 % or Referansar
42.41 \def\chaptername{Kapittel}%
42.42 \def\appendixname{Tillegg}% or Appendiks
42.43\\def\contentsname{Innhald}%
42.44 \def\listfigurename{Figurar}% or Figurliste
42.45\\def\listtablename{Tabellar}% or Tabelliste
42.46\\def\indexname{Register}%
42.47 \def\figurename{Figur}%
42.48 \def\tablename{Tabell}%
42.49 \def\partname{Del}%
42.50 \def\enclname{Vedlegg}%
42.51 \def\ccname{Kopi til}%
42.52\\def\headtoname{Til}% in letter
42.53 \def\pagename{Side}%
42.54 \def\seename{Sj\aa{}}%
42.55\\def\alsoname{Sj\aa{} \'{o}g}%
42.56 \def\proofname{Bevis}%
42.57 \def\glossaryname{Ordliste}%
42.58 }
42.59\fi
```

\datenorsk The macro \datenorsk redefines the command \today to produce Norwegian dates.

```
42.60\@namedef{date\CurrentOption}{%
42.61 \def\today{\number\day. ~\ifcase\month\or
42.62 januar\or februar\or mars\or april\or mai\or juni\or
42.63 juli\or august\or september\or oktober\or november\or desember
42.64 \fi
42.65\\space\number\year}}
```

\extrasnorsk The macro \extrasnorsk will perform all the extra definitions needed for the \extrasnynorsk Norwegian language. The macro \noextrasnorsk is used to cancel the actions of \extrasnorsk.

Norwegian typesetting requires \frencspacing to be in effect.
42.66 \@namedef \{extras \CurrentOption\}\{\bbl@frenchspacing\}
42.67 \@namedef \{noextras \CurrentOption\}\{\bbl@nonfrenchspacing\}

For Norsk the " character is made active. This is done once, later on its definition may vary.

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

42.69 \expandafter \addto\csname extras\CurrentOption\endcsname\{\%
42.70 \languageshorthands\{norsk\}\}
42.71 \expandafter $\backslash a d d t o \backslash c s n a m e ~ e x t r a s \backslash C u r r e n t O p t i o n \backslash e n d c s n a m e\{\%$
42.72 \bbl@activate\{"\}\}

Don't forget to turn the shorthands off again.

```
42.73\expandafter\addto\csname noextras\CurrentOption\endcsname{%
42.74 \bbl@deactivate{"}}
```

The code above is necessary because we need to define a number of shorthand commands. These sharthand commands are then used as indicated in table 17.

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 ".

```
42.75 \begingroup \catcode'\"12
42.76\def\x{\endgroup
42.77 \def\@SS{\mathchar"7019 }
42.78 \def\dq{"}}
42.79 \x
```

Now we can define the discretionary shorthand commands. The number of words where such hyphenation is required is for each character
taken from a list of 83000 ispell-roots.

```
42.80\declare@shorthand{norsk}{"b}{\textormath{\bbl@disc b{bb}}{b}}
42.81 \declare@shorthand{norsk}{"B}{\textormath{\bbl@disc B{BB}}{B}}
42.82\declare@shorthand{norsk}{"d}{\textormath{\bbl@disc d{dd}}{d}}
42.83\declare@shorthand{norsk}{"D}{\textormath{\bbl@disc D{DD}}{D}}
42.84 \declare@shorthand{norsk}{"e}{\textormath{\bbl@disc e{\'e}}{}}
42.85\declare@shorthand{norsk}{"E}{\textormath{\bbl@disc E{\'E}}{}}
42.86 \declare@shorthand{norsk}{"F}{\textormath{\bbl@disc F{FF}}{F}}
42.87\declare@shorthand{norsk}{"g}{\textormath{\bbl@disc g{gg}}{g}}
42.88 \declare@shorthand{norsk}{"G}{\textormath{\bbl@disc G{GG}}{G}}
42.89 \declare@shorthand{norsk}{"k}{\textormath{\bbl@disc k{kk}}{k}}
42.90\declare@shorthand{norsk}{"K}{\textormath{\bbl@disc K{KK}}{K}}
42.91 \declare@shorthand{norsk}{"l}{\textormath{\bbl@disc l{ll}}{l}}
42.92 \declare@shorthand{norsk}{"L}{\textormath{\bbl@disc L{LL}}{L}}
42.93\declare@shorthand{norsk}{"n}{\textormath{\bbl@disc n{nn}}{n}}
42.94 \declare@shorthand{norsk}{"N}{\textormath{\bbl@disc N{NN}}{N}}
42.95\declare@shorthand{norsk}{"p}{\textormath{\bbl@disc p{pp}}{p}}
42.96 \declare@shorthand{norsk}{"P}{\textormath{\bbl@disc P{PP}}{P}}
42.97\declare@shorthand{norsk}{"r}{\textormath{\bbl@disc r{rr}}{r}}
42.98 \declare@shorthand{norsk}{"R}{\textormath{\bbl@disc R{RR}}{R}}
42.99 \declare@shorthand{norsk}{"s}{\textormath{\bbl@disc s{ss}}{s}}
42.100 \declare@shorthand{norsk}{"S}{\textormath{\bbl@disc S{SS}}{S}}
42.101 \declare@shorthand{norsk}{"t}{\textormath{\bbl@disc t{tt}}{t}}
42.102 \declare@shorthand{norsk}{"T}{\textormath{\bbl@disc T{TT}}{T}}
```

We need to treat " f a bit differently in order to preserve the ff-ligature. 42.103 \declare@shorthand\{norsk\}\{"f\}\{\textormath\{\bbl@discff\}\{f\}\} 42.104 \def $\backslash b b 1 @ d i s c f f\{\backslash$ penalty $\backslash @ M$

```
42.105 \afterassignment\bbl@insertff \let\bbl@nextff= }
42.106 \def\bbl@insertff{%
42.107 \if f\bbl@nextff
42.108 \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi
42.109 {\relax\discretionary{ff-}{f}{ff}\allowhyphens}{f\bbl@nextff}}
42.110\let\bbl@nextff=f
```

We now define the French double quotes and some commands concerning hyphenation:
42.111 \declare@shorthand\{norsk\}\{"<\}\{\flqq\}
42.112 \declare@shorthand\{norsk\}\{">\}\{\frqq\}
42.113 \declare@shorthand\{norsk\}\{"-\}\{\penalty\@M\-\bbl@allowhyphens\}
42.114 \declare@shorthand\{norsk\}\{"|\}\{\%
42.115 \textormath\{\penalty $\backslash$ @M discretionary\{-\}\{\}\{\kern.03em\}\%
42.116 \allowhyphens\}\{\}\}
42.117 \declare@shorthand\{norsk\}\{""\}\{\hskip\z@skip\}
42.118 \declare@shorthand\{norsk\}\{"~\}\{\textormath\{\leavevmode $\backslash$ hbox $\{-\}\}\{-\}\}$
42.119 \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.
$42.120 \backslash l d f @ f i n i s h \backslash C u r r e n t O p t i o n$
42.121 〈/code〉

## 43 The Swedish language

The file swedish. $\mathrm{dtx}^{50}$ 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 18 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 "0. <br> "w, "W <br> gives à and A. <br> for ff to be hyphenated as ff-f. Used for com- <br> pound words, such as stra"ffåge, which should <br> be hyphenated as straff-fånge. This is also imple- <br> mented for b, d, f, g, l, m, n, p, r, s, and t. <br> disable ligature at this position. This should be used <br> for compound words, such as "stra"ffinrättning", <br> which should not have the ligature "ff". |
| :--- | :--- |
| "। |  |
| an explicit hyphen sign, allowing hyphenation in the |  |
| rest of the word, such as e. g. in "x"-axeln". |  |

Table 18: 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.
43.1 〈* code〉
43.2 \LdfInit\{swedish\}\captionsswedish

When this file is read as an option, i.e. by the command,swedishwillbean'unknown'languageinwhichcasewehavetomakeitknown.Soweundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

[^41]check for the existence of \1@swedish to see whether we have to do something here.

```
43.3 \ifx \(\backslash 1 @ s w e d i s h \backslash @ u n d e f i n e d\)
\(43.4 \backslash @ n o p a t t e r n s\{S w e d i s h\}\)
43.5 \adddialect\1@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 $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
43.6 \addto\captionsswedish{%
43.7 \def\prefacename{F\"orord}%
43.8 \def\refname{Referenser}%
43.9\def\abstractname{Sammanfattning}%
43.10\def\bibname{Litteraturf\"orteckning}%
43.11 \def\chaptername{Kapitel}%
43.12 \def\appendixname{Bilaga}%
43.13\\def\contentsname{Inneh\csname aa\endcsname ll}%
43.14 \def\listfigurename{Figurer}%
43.15\def\listtablename{Tabeller}%
43.16 \def\indexname{Sakregister}%
43.17 \def\figurename{Figur}%
43.18\\def\tablename{Tabell}%
43.19\def\partname{Del}%
43.20\def\enclname{Bil.}%
43.21 \def\ccname{Kopia f\"or k\"annedom}%
43.22 \def\headtoname{Till}% in letter
43.23\def\pagename{Sida}%
43.24\def\seename{se}%
43.25\def\alsoname{se \"aven}%
43.26 \def\proofname{Bevis}%
43.27\\def\glossaryname{Ordlista}%
43.28 }%
```

\dateswedish The macro \dateswedish redefines the command \today to produce Swedish dates.

```
43.29\def\dateswedish{%
43.30 \def\today{%
43.31 \number\day~\ifcase\month\or
43.32 januari\or februari\or mars\or april\or maj\or juni\or
43.33 juli\or augusti\or september\or oktober\or november\or
43.34 december\fi
43.35 \space\number\year}}
```

\datesymd The macro \datesymd redefines the command \today to produce dates in the format YYYY-MM-DD, common in Sweden.

```
43.36 \def\datesymd{%
43.37\def\today{\number\year-\two@digits\month-\two@digits\day}%
43.38 }
```

\datesdmy The macro \datesdmy redefines the command \today to produce Swedish dates in the format DD/MM YYYY, also common in Sweden.

```
43.39 \def\datesdmy{%
43.40\\def\today{\number\day/\number\month\space\number\year}%
43.41 }
```

\swedishhyphenmins The swedish hyphenation patterns can be used with \lefthyphenmin set to 2 and $\backslash$ righthyphenmin set to 2 .
43.42 \providehyphenmins\{swedish\}\{\tw@\tw@\}
\extrasswedish The macro \extrasswedish performs all the extra definitions needed for the \noextrasswedish 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.

```
43.43 \addto\extrasswedish{\bbl@frenchspacing}
43.44 \addto\noextrasswedish{\bbl@nonfrenchspacing}
```

For Swedish the " character is made active. This is done once, later on its definition may vary.

```
43.45\initiate@active@char{"}
43.46 \addto\extrasswedish{\languageshorthands{swedish}}
43.47 \addto\extrasswedish{\bbl@activate{"}}
```

Don't forget to turn the shorthands off again.

```
43.48 \addto\noextrasswedish{\bbl@deactivate{"}}
```

The "umlaut" accent macro $\backslash$ " is changed to lower the "umlaut" dots. The redefinition is done with the help of \umlautlow.
43.49 \addto\extrasswedish\{\babel@save\"\umlautlow\}
43.50 \addto \noextrasswedish\{\umlauthigh\}

The code above is necessary because we need an extra active character. This character is then used as indicated in table 18.

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 ".
43.51 \begingroup \catcode'\"12
$43.52 \backslash \mathrm{def} \backslash \mathrm{x}\{\backslash$ endgroup
43.53 \def \@SS\{\mathchar"7019 \}
43.54 \def $\backslash \mathrm{dq}\{"\}\}$
43.55 \x

Now we can define the doublequote macros: the umlauts and à.
43.56 \declare@shorthand\{swedish\}\{"w\}\{\textormath\{\{\aa\}\allowhyphens\}\{\ddot w\}\}
$43.57 \backslash$ declare@shorthand\{swedish\}\{"a\}\{\textormath\{\"\{a\}\allowhyphens\}\{\ddot a\}\}
43.58 \declare@shorthand\{swedish\}\{"o\}\{\textormath\{\"\{o\}\allowhyphens\}\{\ddot o\}\}
43.59 \declare@shorthand\{swedish\}\{"W\}\{\textormath\{\{\AA\}\allowhyphens\}\{\ddot W\}\}
$43.60 \backslash$ declare@shorthand\{swedish\}\{"A\}\{\textormath\{\"\{A\}\allowhyphens\}\{\ddot A\}\}
43.61 \declare@shorthand\{swedish\}\{"0\}\{\textormath $\{\backslash$ "\{0\}\allowhyphens\}\{\ddot 0\}\}
discretionary commands

```
43.62 \declare@shorthand{swedish}{"b}{\textormath{\bbl@disc b{bb}}{b}}
43.63 \declare@shorthand{swedish}{"B}{\textormath{\bbl@disc B{BB}}{B}}
43.64 \declare@shorthand{swedish}{"d}{\textormath{\bbl@disc d{dd}}{d}}
43.65 \declare@shorthand{swedish}{"D}{\textormath{\bbl@disc D{DD}}{D}}
43.66\declare@shorthand{swedish}{"f}{\textormath{\bbl@disc f{ff}}{f}}
43.67\declare@shorthand{swedish}{"F}{\textormath{\bbl@disc F{FF}}{F}}
43.68\declare@shorthand{swedish}{"g}{\textormath{\bbl@disc g{gg}}{g}}
43.69 \declare@shorthand{swedish}{"G}{\textormath{\bbl@disc G{GG}}{G}}
43.70\declare@shorthand{swedish}{"l}{\textormath{\bbl@disc l{ll}}{l}}
43.71 \declare@shorthand{swedish}{"L}{\textormath{\bbl@disc L{LL}}{L}}
43.72 \declare@shorthand{swedish}{"m}{\textormath{\bbl@disc m{mm}}{m}}
43.73\declare@shorthand{swedish}{"M}{\textormath{\bbl@disc M{MM}}{M}}
43.74\declare@shorthand{swedish}{"n}{\textormath{\bbl@disc n{nn}}{n}}
43.75 \declare@shorthand{swedish}{"N}{\textormath{\bbl@disc N{NN}}{N}}
43.76 \declare@shorthand{swedish}{"p}{\textormath{\bbl@disc p{pp}}{p}}
43.77 \declare@shorthand{swedish}{"P}{\textormath{\bbl@disc P{PP}}{P}}
43.78\declare@shorthand{swedish}{"r}{\textormath{\bbl@disc r{rr}}{r}}
43.79\declare@shorthand{swedish}{"R}{\textormath{\bbl@disc R{RR}}{R}}
43.80\declare@shorthand{swedish}{"s}{\textormath{\bbl@disc s{ss}}{s}}
43.81 \declare@shorthand{swedish}{"S}{\textormath{\bbl@disc S{SS}}{S}}
43.82\declare@shorthand{swedish}{"t}{\textormath{\bbl@disc t{tt}}{t}}
43.83\declare@shorthand{swedish}{"T}{\textormath{\bbl@disc T{TT}}{T}}
```

and some additional commands:

```
43.84\declare@shorthand{swedish}{"-}{\nobreak-\bbl@allowhyphens}
43.85 \declare@shorthand{swedish}{"|}{%
43.86 \textormath{\nobreak\discretionary{-}{}{\\kern.03em}%
43.87 \bbl@allowhyphens}{}}
43.88\declare@shorthand{swedish}{""}{\hskip\z@skip}
43.89 \declare@shorthand{swedish}{"~}{%
43.90 \textormath{\leavevmode\hbox{-}\bbl@allowhyphens}{-}}
43.91 \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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ can generate from the hyphenation patterns.

```
43.92 \addto\extrasswedish{\babel@save\-}
43.93\addto\extrasswedish{\def\-{\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.
43.95 \ldf@finish\{swedish\}
43.96 〈/code〉

## 44 The North Sami language

The file samin. $\mathrm{dtx}^{51}$ 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 $\mathrm{A}_{\mathrm{E}} \mathrm{E}_{\mathrm{E}} \mathrm{f}$ format file. See the documentation for your $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ distribution.

A note on writing North Sami in $\mathrm{LA}_{\mathrm{E}} \mathrm{X}$ : 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
$\backslash$ newcommand $\{\backslash t x\}\{\backslash$ mbox $\{t \backslash h s p a c e\{-.35 \mathrm{em}\}-\}\}$
\newcommand\{\txx\}\{\mbox\{T\hspace\{-. 5em\}-\}\}
in the preamble of your documents. They define the commands
$\backslash t x x\}$ for LATIN CAPITAL LETTER T WITH STROKE and
$\backslash t x\}$ for LATIN SMALL LETTER T WITH STROKE.

### 44.1 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.
44.1 〈*code〉
44.2 \LdfInit\{samin\}\{captionssamin\}

When this file is read as an option, i.e. by the command,samincouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof$\backslash1@$samintoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
44.3\ifx\undefined\l@samin
44.4 \@nopatterns{Samin}
44.5 \adddialect\l@samin0\fi
```

The next step consists of defining commands to switch to (and from) the North Sami language.

This macro is used to store the correct values of the hyphenation parameters \lefthyphenmin and \righthyphenmin.
44.6 \providehyphenmins\{samin\}\{\tw@\tw@\}

[^42]\captionssamin The macro \captionssamin defines all strings used in the four standard documentclasses provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
44.7 \def\captionssamin{%
44.8\\def\prefacename{Ovdas\'atni}%
44.9 \def\refname{\v Cujuhusat}%
44.10 \def\abstractname{\v Coahkk\'aigeassu}%
44.11 \def\bibname{Girjj\'ala\v svuohta}%
44.12 \def\chaptername{Kapihttal}%
44.13\def\appendixname{\v Cuovus}%
44.14 \def\contentsname{Sisdoallu}%
44.15\def\listfigurename{Govvosat}%
44.16 \def\listtablename{Tabeallat}%
44.17 \def\indexname{Registtar}%
44.18\\def\figurename{Govus}%
44.19\def\tablename{Tabealla}%
44.20 \def\partname{Oassi}%
44.21 \def\enclname{Mielddus}%
44.22 \def\ccname{Kopia s\'addejuvvon}%
44.23 \def\headtoname{Vuost\'aiv\'aldi}%
44.24\\def\pagename{Siidu}%
44.25 \def\seename{geah\v ca}%
44.26 \def\alsoname{geah\v ca maidd\'ai}%
44.27 \def\proofname{Duo\dj{}a\v stus}%
44.28\\def\glossaryname{S\'atnelistu}%
44.29 }%
```

\datesamin The macro \datesamin redefines the command \today to produce North Sami dates.

```
44.30\def\datesamin{%
44.31 \def\today{\ifcase\month\or
44.32 o\dj{}\dj{}ajagem\'anu\or
44.33 guovvam\'anu\or
44.34 njuk\v cam\'anu\or
44.35 cuo\ng{}om\'anu\or
44.36 miessem\'anu\or
44.37 geassem\'anu\or
44.38 suoidnem\'anu\or
44.39 borgem\'anu\or
44.40 \v cak\v cam\'anu\or
44.41 golggotm\'anu\or
44.42 sk\'abmam\'anu\or
44.43 juovlam\'anu\fi
44.44 \space\number\day . ~b.\space\number\year}%
44.45 }%
```

\extrassamin The macro \extrassamin will perform all the extra definitions needed for the $\backslash$ noextrassamin 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.

### 44.46 \addto \extrassamin\{\}

44.47 \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.
44.48 \ldf@finish\{samin\}
44.49 〈/code〉

## 45 The Finnish language

The file finnish. $\mathrm{dtx}^{52}$ defines all the language definition macros for the Finnish language.

For this language the character " is made active. In table 19 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 $\ll$ ).
"> for French right double quotes (similar to $\gg$ ).
$\backslash$ - like the old $\backslash$-, but allowing hyphenation in the rest of the word.

Table 19: 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.

```
45.1 〈*code>
45.2 \LdfInit{finnish}\captionsfinnish
```

When this file is read as an option, i.e. by the command,finnishwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowehere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
45.3\ifx\l@finnish\@undefined
45.4 \@nopatterns{Finnish}
45.5\\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.
\captionsfinnish The macro \captionsfinnish defines all strings used in the four standard documentclasses provided with $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$.
45.6 \addto\captionsfinnish\{\%
45.7 \def\prefacename\{Esipuhe\}\%
45.8 \def $\backslash$ refname\{Viitteet $\} \%$

[^43]```
                    45.9\def\abstractname{Tiivistelm\"a}
45.10\def\bibname{Kirjallisuutta}%
45.11 \def\chaptername{Luku}%
45.12\def\appendixname{Liite}%
45.13 \def\contentsname{Sis\"alt\"o}% /* Could be "Sis\"allys" as well */
45.14 \def\listfigurename{Kuvat}%
45.15\def\listtablename{Taulukot}%
45.16\def\indexname{Hakemisto}%
45.17 \def\figurename{Kuva}%
45.18\def\tablename{Taulukko}%
45.19 \def\partname{0sa}%
45.20\\def\enclname{Liitteet}%
45.21\\def\ccname{Jakelu}%
45.22 \def\headtoname{Vastaanottaja}%
45.23\def\pagename{Sivu}%
45.24\def\seename{katso}%
45.25\def\alsoname{katso my\"os}%
45.26\def\proofname{Todistus}%
45.27\def\glossaryname{Sanasto}%
45.28 }%
```

\datefinnish The macro \datefinnish redefines the command \today to produce Finnish dates.

```
45.29\def\datefinnish{%
45.30\\def\today{\number\day. ~\ifcase\month\or
45.31 tammikuuta\or helmikuuta\or maaliskuuta\or huhtikuuta\or
45.32 toukokuuta\or kes\"akuuta\or hein\"akuuta\or elokuuta\or
45.33 syyskuuta\or lokakuuta\or marraskuuta\or joulukuuta\fi
45.34 \space\number\year}}
```

\extrasfinnish Finnish has many long words (some of them compound, some not). For this \noextrasfinnish 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 $T_{E} X$ 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.

```
45.35 \addto\extrasfinnish{%
45.36\babel@savevariable\hyphenpenalty\hyphenpenalty=30%
45.37\\mathrm{ \abel@savevariable\exhyphenpenalty\exhyphenpenalty=30%}
45.38\babel@savevariable\doublehyphendemerits\doublehyphendemerits=5000%
45.39 \babel@savevariable\finalhyphendemerits\finalhyphendemerits=5000%
45.40 }
45.41 \addto\noextrasfinnish{}
```

Another thing \extrasfinnish needs to do is to ensure that $\backslash$ frenchspacing
is in effect. If this is not the case the execution of \noextrasfinnish will switch it of again.

```
45.42 \addto\extrasfinnish{\bbl@frenchspacing}
45.43\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.

```
45.44 \initiate@active@char{"}
45.45 \addto\extrasfinnish{\languageshorthands{finnish}}
```

Don't forget to turn the shorthands off again.

```
45.46 \addto\extrasfinnish{\bbl@activate{"}}
45.47 \addto\noextrasfinnish{\bbl@deactivate{"}}
```

The 'umlaut' character should be positioned lower on all vowels in Finnish texts.

```
45.48 \addto\extrasfinnish{\umlautlow\umlautelow}
```

45.49 \addto \noextrasfinnish\{ $\backslash$ umlauthigh\}

First we define access to the low opening double quote and guillemets for quotations,

```
45.50 \declare@shorthand{finnish}{" '}{%
45.51 \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
45.52 \declare@shorthand{finnish}{"'}{%
45.53 \textormath{\textquotedblright}{\mbox{\textquotedblright}}}
45.54 \declare@shorthand{finnish}{"<}{%
45.55 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
45.56 \declare@shorthand{finnish}{">}{%
45.57 \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```
45.58\declare@shorthand{finnish}{"-}{\nobreak-\bbl@allowhyphens}
45.59 \declare@shorthand{finnish}{""}{\hskip\z@skip}
45.60\declare@shorthand{finnish}{"=}{\hbox{-}\bbl@allowhyphens}
```

And we want to have a shorthand for disabling a ligature.

```
45.61 \declare@shorthand{finnish}{"|}{%
45.62 \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

\- All that is left now is the redefinition of $\backslash$-. The new version of $\backslash$ - should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ can generate from the hyphenation patterns.

```
45.63 \addto\extrasfinnish{\babel@save\-}
45.64\addto\extrasfinnish{\def\-{\bbl@allowhyphens
```

\finishhyphenmins The finnish hyphenation patterns can be used with \lefthyphenmin set to 2 and $\backslash$ righthyphenmin set to 2 .
45.66 \providehyphenmins\{ $\backslash$ CurrentOption\}\{\tw@\tw@\}

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of © to its original value.
$45.67 \backslash l d f @ f i n i s h\{f i n n i s h\}$
45.68 〈/code〉

## 46 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 1.3 i was essentially changing the English document elements to Hungarian ones, but because of the differences between these too 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 $1.3 \mathrm{i}^{53}$ the special typesetting rules of the Hungarian language mentioned above were not taken into consideration. This version (v1.4j) ${ }^{54}$ enables babel to typeset 'good-looking' Hungarian texts.

The \ontoday command works like \today but produces a slightly different date format used in expressions such as 'on February 10th'.
$\backslash A z \quad$ The commands $\backslash$ Az\#1 and $\backslash$ az\#1 write the correct definite article for the argument and the argument itself (separated with a ~). The star-forms ( $\backslash$ Az* and \az*) produce the article only.
\Azr $\backslash$ Azr\#1 and \azr\#1 treat the argument as a label so expand it then write the

[^44]| shortcut | explanation | example |
| :---: | :---: | :---: |
| ، 6 | same as \glqq in babel, or \quotedblbase in T1 (opening quotation mark, like ") | ''id\'ezet'' $\longrightarrow$,idézet" |
| 'c, 'C | ccs is hyphenated as cs-cs | 10'ccsan $\longrightarrow$ locs-csan |
| 'd, 'D | ddz is hyphenated as dz-dz | e'ddz\"unk $\longrightarrow$ edz-dzünk |
| 'g, 'G | ggy is hyphenated as gy-gy | po'ggy ${ }^{\prime}$ 'asz $\longrightarrow$ pogy-gyász |
| ${ }^{\prime} 1$, 'L | lly is hyphenated as ly-ly | Kod\'a'llyal $\longrightarrow$ Kodály-lyal |
| ' $\mathrm{n},{ }^{\prime} \mathrm{N}$ | nny is hyphenated as ny-ny | me'nnyei $\longrightarrow$ meny-nyei |
| 's, 'S | ssz is hyphenated as sz-sz | vi'ssza $\longrightarrow$ visz-sza |
| 't, 'T | tty is hyphenated as ty-ty | po'ttyan $\longrightarrow$ poty-tyan |
| 'z, 'z | zzs is hyphenated as zs-zs | ri'zzsel $\longrightarrow$ rizs-zsel |

Table 20: The shortcuts defined in magyar.ldf
definite article for $\backslash \mathbf{r} @ 1$, a non-breakable space then the label expansion. The star-forms do not print the label expansion. $\backslash \operatorname{Azr}$ (\#1 and $\backslash$ azr (\#1 are used for equation referencing with the syntax $\backslash \operatorname{azr}$ (label).
$\backslash$ Aref
There are two aliases $\backslash$ Aref and \aref for $\backslash$ Azr and \azr, respectively. During the preparation of a document it is not known in general, if the code ' $\mathrm{a}^{\sim} \backslash \mathrm{ref}\{l a b e l\}$ ' or the code ' $\mathrm{az} \sim \backslash$ ref $\{$ label $\}$ ' is the grammatically correct one. Writing ' $\backslash$ aref $\{$ label $\}$ ' instead of the previous ones solves the problem.
$\backslash$ Azp\#1 and \azp\#1 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.

## $\backslash$ Apageref

There are aliases \Apageref and \apageref for $\backslash$ Azp and \azp, respectively. The code \apageref $\{l a b e l\}$ is equivalent either to $\mathrm{a}^{\sim} \backslash$ pageref $\{l a b e l\}$ or to az~ ${ }^{\sim}$ pageref $\{l a b e l\}$.
$\backslash \mathrm{Azc}$
$\backslash A z c$ and $\backslash a z c$ 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.
$\backslash$ Acite There are aliases $\backslash$ Acite and $\backslash$ acite for $\backslash \mathrm{Azc}$ and $\backslash$ azc, respectively.
For this language the character ' is made active. Table 20 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.

```
46.1 〈*code>
46.2\LdfInit{magyar}{caption\CurrentOption}
```

When this file is read as an option, i.e. by the command,magyarwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@magyaror\1@hungariantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
46.3\ifx\l@magyar\@undefined
46.4 \ifx\l@hungarian\@undefined
46.5 \@nopatterns{Magyar}
46.6 \adddialect\l@magyar0
46.7 \else
46.8 \let\l@magyar\l@hungarian
46.9 \fi
46.10 \fi
```

The statement above makes sure that \1@magyar is always defined; if \l@hungarian is still undefined we make it equal to \1@magyar.

```
46.11\ifx\l@hungarian\@undefined
46.12 \let\l@hungarian\l@magyar
46.13\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 ${ }^{\mathrm{A}} \mathrm{T}_{\mathrm{E}} \mathrm{X}$.
46.14 \@namedef \{captions \CurrentOption\}\{\%
46.15 \def\prefacename\{E1\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.
46.16 \def $\backslash r e f n a m e\{H i v a t k o z \backslash ' a s o k\} \%$

If you have a document with a summary instead of an abstract you might want to replace the word 'Kivonat' with 'Összefoglaló'.
46.17 \def\abstractname\{Kivonat\}\%

The Hungarian version of 'Bibliography' is 'Bibliográfia', but a more natural word to use is 'Irodalomjegyzék'.

```
46.18\\def\bibname{Irodalomjegyz\'ek}%
46.19 \def\chaptername{fejezet}%
46.20\def\appendixname{F\"uggel\'ek}%
46.21\\def\contentsname{Tartalomjegyz\'ek}%
46.22 \def\listfigurename{\'Abr\'ak jegyz\'eke}%
46.23\\def\listtablename{T\'abl\'azatok jegyz\'eke}%
46.24\def\indexname{T\'argymutat\'o}%
46.25 \def\figurename{\'abra}%
46.26 \def\tablename{t\'abl\'azat}%
46.27 \def\partname{r\'esz}%
46.28\\def\enclname{Mell\'eklet}%
46.29\def\ccname{K\"orlev\'el--c\'\i mzettek}%
46.30\def\headtoname{C\'\i mzett}%
46.31 \def\pagename{oldal}%
46.32\\def\seename{l\'asd}%
46.33\\def\alsoname{l\'asd m\'eg}%
```

Besides the Hungarian word for Proof, 'Bizonyítás' we can also name Corollary (Következmény), Theorem (Tétel) and Lemma (Lemma).

```
46.34 \def\proofname{Bizony\'\i t\'as}%
46.35\def\glossaryname{Sz\'ojegyz\'ek}%
46.36 }%
```

\datemagyar The macro \datemagyar redefines the command \today to produce Hungarian dates.

```
46.37\@namedef{date\CurrentOption}{%
46.38\\def\today{%
46.39 \number\year.\nobreakspace\ifcase\month\or
46.40 janu\'ar\or febru\'ar\or m\'arcius\or
46.41 \'aprilis\or m\'ajus\or j\'unius\or
46.42 j\'ulius\or augusztus\or szeptember\or
46.43 okt\'ober\or november\or december\fi
46.44 \space\number\day.}}
```

\ondatemagyar The macro \ondatemagyar produces Hungarian dates which have the meaning 'on this day'. It does not redefine the command \today.

| 46.45 \@namedef \{ondate\CurrentOption\} \{\% |  |
| :---: | :---: |
| 46.46 | $\backslash$ number \year.\nobreakspace\ifcase\month \or |
| 46.47 | janu\'ar\or febru\'ar\or m\'arcius \or |
| 46.48 | \'aprilis \or m\'ajus \or j \'unius ${ }^{\text {dor }}$ |
| 46.49 | $j \backslash$ 'ulius \or augusztus \or szeptember\or |
| 46.50 | okt\'ober\or november\or december\fi |
| 46.51 | $\backslash$ space\ifcase\day $\backslash$ or |
| 46.52 | 1-j\'en\or 2-\'an\or 3-\'an\or 4-\'en\or 5-\'en\or |
| 46.53 | 6-\'an\or 7-\'en\or 8-\'an\or 9-\'en\or 10-\'en\or |
| 46.54 | 11-\'en\or 12-\'en\or 13-\'an\or 14-\'en\or 15-\'en\or |
| 46.55 | 16-\'an\or 17-\'en\or 18-\'an\or 19-\'en ${ }^{\text {dor }}$ 20-\'an\or |
| 46.56 | 21-\'en\or 22-\'en\or 23-\'an\or 24-\'en\or 25-\'en\or |
| 46.57 | 26-\'an\or 27-\'en\or 28-\'an\or 29-\'en ${ }^{\text {dor }} 30-\$ \an ${ }^{\text {a }}$ |
| 46.58 | 31-\'en\fi\} |

\extrasmagyar The macro \extrasmagyar will perform all the extra definitions needed for the \noextrasmagyar Hungarian language. The macro \noextrasmagyar is used to cancel the actions of \extrasmagyar.

```
46.59 \@namedef{extras\CurrentOption}{%
46.60 \expandafter\let\expandafter\ontoday
46.61 \csname ondate\CurrentOption\endcsname}
46.62\@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.
$\backslash$ fnum@figure In figure and table captions the order of the figure/table number and $\backslash$ figurename \fnum@table / tablename must be changed. To achieve this \fnum@figure and $\backslash$ fnum@table are redefined and added to \extrasmagyar.

```
46.63\expandafter\addto\csname extras\CurrentOption\endcsname{%
46.64 \babel@save\fnum@figure
46.65\\def\fnum@figure{\thefigure.\nobreakspace\figurename}}
46.66 \expandafter\addto\csname extras\CurrentOption\endcsname{%
46.67 \babel@save\fnum@table
46.68 \def\fnum@table{0ble.\nobreakspace\tablename}}
```

\@makecaption The colon in a figure/table caption must be replaced by a dot by redefining \@makecaption
46.69 \expandafter\addto\csname extras \CurrentOption\endcsname\{\%
46.70 \babel@save\@makecaption
46.71 \def \@makecaption\#1\#2\{\%
46.72 \vskip\abovecaptionskip
46.73 \sbox\@tempboxa\{\#1. \#2\}\%
46.74 \ifdim \wd\@tempboxa > Chsize
46.75 \{\#1. \#2\csname par\endcsname\}
$46.76 \backslash$ else
$46.77 \quad$ \global \@minipagefalse
46.78 \hb@xt@\hsize\{\hfil\box\@tempboxa\hfil\}\%
$46.79 \quad$ fi
46.80 \vskip\belowcaptionskip\}\}
\@caption There should be a dot after the figure/table number in lof/lot, so \@caption is redefined.

```
46.81 \expandafter\addto\csname extras\CurrentOption\endcsname{%
46.82 \babel@save\@caption
46.83 \long\def\@caption#1[#2]#3{%
46.84 \csname par\endcsname
46.85 \addcontentsline{\csname ext@#1\endcsname}{#1}%
46.86 {\protect\numberline{\csname the#1\endcsname.}{\ignorespaces #2}}%
46.87 \begingroup
46.88 \@parboxrestore
46.89 \if@minipage
46.90 \@setminipage
46.91 \fi
46.92 \normalsize
46.93 \@makecaption{\csname fnum@#1\endcsname}%
46.94 {\ignorespaces #3}\csname par\endcsname
46.95 \endgroup}}
```

\@seccntformat In order to have a dot after the section number \@secentformat is redefined.

```
46.96 \expandafter\addto\csname extras\CurrentOption\endcsname{%
46.97\ \babel@save\@seccntformat
46.98\def\@seccntformat#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.
46.99 \expandafter $\backslash$ addto $\backslash$ csname extras $\backslash$ CurrentOption $\backslash$ endcsname $\{\%$
46.100 \babel@save\@sect
46.101 \def \@sect\#1\#2\#3\#4\#5\#6[\#7]\#8\{\%
46.102 \ifnum \#2>\c@secnumdepth
$46.103 \backslash$ let \@svsec \@empty
46.104 \else
$46.105 \backslash$ refstepcounter\{\#1\}\%
46.106 \protected@edef \@svsec\{\@seccntformat\{\#1\}\relax\}\%
$46.107 \quad$ fi
46.108 \@tempskipa \#5\relax
46.109 \ifdim \@tempskipa>\z@
46.110 \begingroup
46.111 \#6\{\%
46.112
46.113 \interlinepenalty \@M \#8\@@par\}\%
46.114 \endgroup
46.115 \csname \#1mark\endcsname\{\#7\}\%
$46.116 \backslash$ addcontentsline\{toc\}\{\#1\}\{\%
46.117 \ifnum \#2>\c@secnumdepth \else
46.118 \protect\numberline\{\csname the\#1\endcsname. $\} \%$
$46.119 \quad$ fi
46.120 \#7\}\%
46.121 \else
46.122 \def \@svsechd\{\%
$46.123 \quad$ \#6\{\hskip \#3\relax
46.124 \@svsec \#8.3\%
46.125 \csname \#1mark\endcsname\{\#7\}\%
46.126 \addcontentsline\{toc\}\{\#1\}\{\%
46.127 \ifnum \#2>\c@secnumdepth \else
46.128 \protect $\backslash$ numberline\{\csname the\#1\endcsname.\}\%
46.129 \fi
46.130 \#7\}\}\%
$46.131 \backslash$ fi
46.132 \@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.
46.133 \expandafter $\backslash$ addto $\backslash$ csname extras $\backslash$ CurrentOption $\backslash e n d c s n a m e\{\%$
46.134 \babel@save\@ssect
46.135 \def \@ssect\#1\#2\#3\#4\#5\{\%
46.136 \@tempskipa \#3\relax
46.137 \ifdim \@tempskipa>\z@
46.138 \begingroup
$46.139 \quad \# 4$ \%
$46.140 \backslash$ Changfrom $\{$ hhskip \#1\}\%
46.141 \interlinepenalty \@M \#5\@@par\}\%
46.142 \endgroup

```
46.143
46.144
4 6 . 1 4 5
46.146
```

```
\else
```

\else
\def\@svsechd{\#4{\hskip \#1\relax \#5.}}%
\def\@svsechd{\#4{\hskip \#1\relax \#5.}}%
\i
\i
\@xsect{\#3}}}

```
\@xsect{#3}}}
```

\@begintheorem Order changing and dot insertion in theorem by redefining \@begintheorem and \@opargbegintheorem \@opargbegintheorem.

```
46.147\expandafter\addto\csname extras\CurrentOption\endcsname{%
46.148 \babel@save\@begintheorem
46.149 \def\@begintheorem#1#2{\trivlist
46.150 \item[\hskip \labelsep{\bfseries #2.\ #1.}]\itshape}%
46.151 \babel@save\@opargbegintheorem
46.152 \def\@opargbegintheorem#1#2#3{\trivlist
46.153 \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.
46.154 \@ifclassloaded\{book\}\{\%
$\backslash p s$ @headings The look of the headings is changed: we have to insert some dots and change the order of chapter number and \chaptername.

```
46.155 \expandafter\addto\csname extras \CurrentOption\endcsname\{\%
    \babel@save\ps@headings\}
\expandafter\addto\csname extras\CurrentOption\endcsname\{\%
    \if@twoside
        \def\ps@headings\{\%
            \let\@oddfoot\@empty\let\@evenfoot\@empty
                    \def \@evenhead\{\thepage\hfil\slshape\leftmark\}\%
                    \(\backslash\) def \@oddhead\{\{\slshape\rightmark\}\hfil\thepage\}\%
                    \let \@mkboth \markboth
            \def \chaptermark\#\#1\{\%
                    \(\backslash\) markboth \{\MakeUppercase\{\%
                            \ifnum \c@secnumdepth >\m@ne
                                    \if@mainmatter
                                    \thechapter. \@chapapp. \ \%
                                    \fi
                    \(\backslash f i\)
                    \#\#1\}\}\{\}\}\%
            \def \sectionmark\#\#1\{\%
                    \(\backslash\) markright \{\MakeUppercase\{\%
                    \ifnum \c@secnumdepth >\z@
                            \thesection. \ \%
                    \fi
                    \#\#1\}\}\}\}\%
    \else
        \(\backslash\) def \(\backslash p s @ h e a d i n g s\{\%\)
            \let\@oddfoot\@empty
```

```
46.181 \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
46.182 \let\@mkboth\markboth
    \def\chaptermark##1{%
46.184
46.185
46.186
46.187
46.188
4 6 . 1 8 9
4 6 . 1 9 0
46.191
        \markright {\MakeUppercase{%
        \ifnum \c@secnumdepth >\m@ne
                            \if@mainmatter
                                \thechapter.\@chapapp.\ %
                \fi
            \i
            ##1}}}}%
\i}
```

\@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.
46.192 \expandafter \addto\csname extras $\backslash$ CurrentOption $\backslash$ endcsname $\{\%$
$46.193 \backslash$ babel@save\@part
$46.194 \backslash \operatorname{def} \backslash @ p a r t[\# 1] \# 2\{\%$
46.195 \ifnum \c@secnumdepth >-2\relax
46.196 \refstepcounter\{part\}\%
\addcontentsline\{toc\}\{part\}\{\thepart. $\backslash$ hspace $\{1 \mathrm{em}\} \# 1\} \%$
\else
\addcontentsline\{toc\}\{part\}\{\#1\}\%
\fi
\markboth\{\}\{\}\%
$\{\backslash$ centering
\interlinepenalty \@M
\normalfont
\ifnum \c@secnumdepth >-2\relax
\huge\bfseries \thepart.\nobreakspace\partname
\csname par \endcsname
\vskip 20\p@
$\backslash f i$
\Huge \bfseries \#2\csname par\endcsname\}\%
\@endpart\}\}
\@chapter The same changes are made to chapter. First the screen typeout and the toc are changed by redefining \@chapter.
46.212 \expandafter\addto\csname extras\CurrentOption\endcsname\{\%
46.213 \babel@save\@chapter
$46.214 \backslash$ def $\backslash$ @chapter[\#1]\#2\{\ifnum \c@secnumdepth $>$ \m@ne
46.215
46.216
\refstepcounter\{chapter\}\%
46.21
46.219
46.22
46.221
46.222
46.223
\if@mainmatter
\typeout\{\thechapter.\space\@chapapp.\}\%
\addcontentsline\{toc\}\{chapter\}\%
$\{\backslash$ protect $\backslash$ numberline $\{\backslash$ thechapter.\}\#1\}\%
\else
\addcontentsline\{toc\}\{chapter\}\{\#1\}\% $\backslash f i$
\else

```
46.224
46.225
46.226
46.227
46.228
46.229
46.230
46.231
46.232
46.233
46.234
```

```
    \addcontentsline{toc}{chapter}{#1}%
```

    \addcontentsline{toc}{chapter}{#1}%
    \i
    \i
    \chaptermark{#1}%
    \chaptermark{#1}%
    \addtocontents{lof}{\protect\addvspace{10\p@}}%
    \addtocontents{lof}{\protect\addvspace{10\p@}}%
    \addtocontents{lot}{\protect\addvspace{10\p@}}%
    \addtocontents{lot}{\protect\addvspace{10\p@}}%
    \if@twocolumn
    \if@twocolumn
    \@topnewpage[\@makechapterhead{#2}]%
    \@topnewpage[\@makechapterhead{#2}]%
    \else
    \else
    \@makechapterhead{#2}%
    \@makechapterhead{#2}%
    \@afterheading
    \@afterheading
    \fi}}

```
\fi}}
```

\@makechapterhead Then the look of the chapter-start is modified by redefining \@makechapterhead.

| 46.235 | \expandafter\addto\csname extras\CurrentOption\endcsname\{\% |
| :---: | :---: |
| 46.236 | \babel@save\@makechapterhead |
| 46.237 | \def \@makechapterhead\#1\{\% |
| 46.238 | \vspace*\{50\p@\}\% |
| 46.239 | $\{\backslash p a r i n d e n t ~ \ z @ ~ \ r a g g e d r i g h t ~ \ n o r m a l f o n t ~$ |
| 46.240 | \ifnum \c@secnumdepth > \m@ne |
| 46.241 | \if@mainmatter |
| 46.242 | $\backslash h u g e \backslash b f s e r i e s ~ \ t h e c h a p t e r . \ n o b r e a k s p a c e \ @ c h a p a p p\{ \} ~$ |
| 46.243 | \csname par\endcsname\nobreak |
| 46.244 | \vskip 20\p@ |
| 46.245 | \fi |
| 46.246 | \fi |
| 46.247 | \interlinepenalty $\ @ M$ |
| 46.248 | $\backslash$ Huge \bfseries \#1\csname par\endcsname\nobreak |
| 46.249 | \vskip 40\p@ |
| 46.250 | \}\}\}\% |

This the end of the book class modification.
46.251 \} \{\}

Now we check if report.cls is loaded.
46.252 \@ifclassloaded\{report\}\{\%
$\backslash p s @ h e a d i n g s$ First the headings are modified just in case of the book class.
46.253 \expandafter\addto\csname extras\CurrentOption\endcsname\{\%
$46.254 \backslash$ babel@save\ps@headings\}
46.255 \expandafter\addto\csname extras \CurrentOption\endcsname\{\%
46.256 \if@twoside
$46.257 \backslash$ def $\backslash p s @ h e a d i n g s\{\%$
$46.258 \backslash$ let \@oddfoot $\backslash$ @empty $\backslash$ let $\backslash$ @evenfoot $\backslash$ @empty
$46.259 \backslash$ def $\backslash @ e v e n h e a d\{\backslash t h e p a g e \backslash h f i l \backslash s l s h a p e \backslash l e f t m a r k\} \%$
$46.260 \backslash$ def \@oddhead\{\{\slshape\rightmark\} $\backslash$ hfil $\backslash$ thepage $\} \%$
$46.261 \backslash$ let $\backslash$ @mbboth $\backslash$ markboth
46.262 \def $\backslash$ chaptermark\#\#1\{\%
46.263 \markboth $\{\backslash$ MakeUppercase\{\%
$46.264 \quad$ \ifnum $\backslash c @ s e c n u m d e p t h>\backslash m @ n e$
46.265 \thechapter. \@chapapp. \ \%

```
46.266
46.267
46.268
46.269
46.270
46.271
46.272
46.273
46.274
46.275
46.276
46.277
46.278
4 6 . 2 7 9
46.280
46.281
46.282
46.283
46.284
46.285
\@chapter Chapter-start modification with \@chapter
    46.286 \expandafter\addto\csname extras\CurrentOption\endcsname{%
    46.287 \babel@save\@chapter
    46.288 \def\@chapter[#1]#2{\ifnum \c@secnumdepth >\m@ne
                                    \refstepcounter{chapter}%
                                    \typeout{\thechapter.\space\@chapapp.}%
                                    \addcontentsline{toc}{chapter}%
                                    {\protect\numberline{\thechapter.}#1}%
                                    \else
    \addcontentsline{toc}{chapter}{#1}%
    \fi
    \chaptermark{#1}%
    \addtocontents{lof}{\protect\addvspace{10\p@}}%
    \addtocontents{lot}{\protect\addvspace{10\p@}}%
    \if@twocolumn
        \@topnewpage[\@makechapterhead{#2}]%
    \else
        \@makechapterhead{#2}%
        \@afterheading
        \fi}}
\@makechapterhead and \@makechapterhead.
    46.305 \expandafter\addto\csname extras\CurrentOption\endcsname{%
    46.306 \babel@save\@makechapterhead
    46.307 \def\@makechapterhead#1{%
    46.308 \vspace*{50\p@}%
    46.309 {\parindent \z@ \raggedright \normalfont
    46.310 \ifnum \c@secnumdepth >\m@ne
    46.311 \huge\bfseries \thechapter.\nobreakspace\@chapapp{}
```

| 46.312 | $\quad$ \csname par\endcsname\nobreak |
| :--- | :--- |
| 46.313 | \vskip 20\p@ |
| 46.314 | \fi |
| 46.315 | \interlinepenalty $\backslash @ M$ |
| 46.316 | \Huge \bfseries \#1\csname par\endcsname\nobreak |
| 46.317 | \vskip 40\p@ |
| 46.318 | \}3\}\% |

End of report class modification.
46.319 \}\{\}

Checking if article.cls is loaded.
46.320 \@ifclassloaded\{article\}\{\%
$\backslash p s @ h e a d i n g s$ Changing headings by redefining \ps@headings.

| 46.321 | \expandafter\addto\csname extras\CurrentOption\endcsname\{\% |
| :---: | :---: |
| 46.322 | \babel@save\ps@headings\} |
| 46.323 | \expandafter\addto\csname extras\CurrentOption\endcsname\{\% |
| 46.324 | \if@twoside |
| 46.325 | $\backslash d e f \backslash p s @ h e a d i n g s\{\%$ |
| 46.326 | \let \@oddfoot\@empty $\backslash$ let \@evenfoot \@empty |
| 46.327 | $\backslash \mathrm{def} \backslash @$ evenhead\{\thepage\hfil\slshape\leftmark\}\% |
| 46.328 | $\backslash d e f \backslash @ o d d h e a d\{\{\backslash$ slshape \rightmark\}\hfil\thepage\}\% |
| 46.329 | \let\@mkboth\markboth |
| 46.330 | \def\sectionmark\#\#1\{\% |
| 46.331 | $\backslash$ markboth \{ ${ }^{\text {MakeUppercase\{\% }}$ |
| 46.332 | \ifnum \c@secnumdepth > ${ }^{\text {c }}$ @ |
| 46.333 | \thesection.\quad |
| 46.334 | \fi |
| 46.335 | \#\#1\}\}\{\}\}\% |
| 46.336 | \def \subsectionmark\#\#1\{\% |
| 46.337 | $\backslash$ markright $\{\%$ |
| 46.338 | \ifnum \c@secnumdepth > \@ne |
| 46.339 | $\backslash$ thesubsection. \quad |
| 46.340 | \fi |
| 46.341 | \#\#1\}\}\}\% |
| 46.342 | $\backslash \mathrm{else}$ |
| 46.343 | \def\ps@headings\{\% |
| 46.344 | \let\@oddfoot\@empty |
| 46.345 | $\backslash d e f \backslash @ o d d h e a d\left\{\left\{\backslash\right.\right.$ slshape \rightmark $^{\text {\hfil }}$ \thepage\}\% |
| 46.346 | \let\@mkboth\markboth |
| 46.347 | \def\sectionmark\#\#1\{\% |
| 46.348 | $\backslash$ markright \{\MakeUppercase\{\% |
| 46.349 | \ifnum \c@secnumdepth > \m@ne |
| 46.350 | \thesection.\quad |
| 46.351 | \fi |
| 46.352 | \#\#1\}\}\}\}\% |
| 46.353 | \fi\}\% |

No more necessary changes specific to the article class.

```
46.354 }{}
```

And now this is the turn of letter.cls.
46.355 \@ifclassloaded\{letter\}\{\%
$\backslash$ ps@headings In the headings the page number must be followed by a dot and then $\backslash$ pagename.

```
46.356 \expandafter\addto\csname extras\CurrentOption\endcsname{%
46.357 \babel@save\ps@headings}
46.358 \expandafter\addto\csname extras\CurrentOption\endcsname{%
4 6 . 3 5 9 ~ \ i f @ t w o s i d e
46.360 \def\ps@headings{%
46.361 \let\@oddfoot\@empty\let\@evenfoot\@empty
46.362 \def\@oddhead{\slshape\headtoname{:} \ignorespaces\toname
46.363 \hfil \@date
46.364 \hfil \thepage.\nobreakspace\pagename}%
46.365 \let\@evenhead\@oddhead}
46.366 \else
46.367 \def\ps@headings{%
4 6 . 3 6 8 ~ \ l e t \ @ o d d f o o t \ @ e m p t y ~
46.369 \def\@oddhead{\slshape\headtoname{:} \ignorespaces\toname
46.370 \hfil \@date
46.371 \hfil \thepage.\nobreakspace\pagename}}
46.372 \fi}%
```

End of letter class.
46.373 \}\{\}

After making the changes to the $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$ macros we define some new ones to handle the problem with definite articles.
$\backslash a z \backslash a z$ is a user-level command which decides if the next character is a star. \@az is called for $\backslash a z *$ and $\backslash a z @$ for $\backslash a z$.
$46.374 \backslash \operatorname{def} \backslash \mathrm{az}\{\mathrm{a} \backslash @ i f s t a r\{\backslash @ a z\}\{$ az@ $\}\}$
$\backslash \mathrm{Az} \backslash \mathrm{Az}$ is used at the beginning of a sentence. Otherwise it behaves the same as $\backslash \mathrm{az}$. 46.375 \def $\backslash$ Az $\{A \backslash$ ©ifstar $\{\backslash @ a z\}\{\backslash a z @\}\}$
$\backslash a z @ \backslash a z @$ is called if there is no star after $\backslash a z$ or $\backslash \mathrm{Az}$. It calls \@az and writes \#1 separating with a non-breakable space.
46.376 \def \az@\#1\{\@az\{\#1\} \nobreakspace\#1\}
\@az This macro calls \hun@tempadef to remove the accents from the argument then calls \@@az that determines if a ' $z$ ' should be written after $a / A$ (written by $\backslash a z / \backslash A z)$.
46.377 \def \@az\#1\{\%
46.378 \hun@tempadef\{relax\}\{relax\}\{\#1\}\%
46.379 \edef \@tempb\{\noexpand \@@az\@tempa\hbox!\}\%
46.380 \@tempb\}
\hun@tempadef The macro \hun@tempadef 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. œ $\rightarrow$ 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 $\mathrm{AAT}_{\mathrm{E}} \mathrm{X}$ 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.
```
46.381 \def\hun@tempadef#1#2#3{%
46.382 \begingroup
46.383\\def\@safe@activesfalse{}%
46.384 \def\setbox ##1{}% to get rid of accents and special letters
46.385 \def \hbox ##1{}%
46.386 \def\accent ##1 ##2{##2}%
46.387 \def\add@accent ##1##2{##2}%
46.388 \def\@text@composite@x ##1##2{##2}%
46.389 \def\i{i}\def\j{j}%
46.390 \def\ae{a}\def\AE{A}\def\oe{o}\def\OE{0}%
46.391 \def\ss{s}\def\L{L}%
46.392 \def\d{}\def\b{}\def\c{}\def\t{{}%
46.393 \expandafter\ifx\csname #1\endcsname\relax
46.394 \expandafter\ifx\csname #2\endcsname\relax
46.395 \xdef\@tempa{#3}%
46.396 \else
46.397 \xdef\@tempa{\csname #2\endcsname}%
4 6 . 3 9 8 ~ \ f i
46.399 \else
46.400 \xdef\@tempa{\csname #1\endcsname}%
46.401 \fi
46.402 \endgroup}
```

The following macros are used to determine the definite article for a label's expansion.
\aref \aref is an alias for \azr.
46.403 \def $\backslash$ aref $\{\backslash$ azr $\}$
$\backslash$ Aref $\backslash$ Aref is an alias for $\backslash$ Azr. 46.404 \def $\backslash$ Aref $\{\backslash$ Azr $\}$
\azr \azr calls \@azr if the next character is a star, otherwise it calls \azr@. 46.405 \def $\backslash$ azr $\{a \backslash @ i f s t a r\{\backslash @ a z r\}\{\backslash a z r @\}\}$
$\backslash$ Azr $\backslash$ Azr is the same as $\backslash$ azr except that it writes ' $A$ ' instead of ' $a$ '. $46.406 \backslash \operatorname{def} \backslash \operatorname{Azr}\{A \backslash @ i f s t a r\{\backslash @ a z r\}\{\backslash a z r @\}\}$
\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. 46.407 \def\azr@\{\@ifnextchar ( $\{\backslash$ azr@@@\}\{\azr@@\}\}
\azr@@ Calls \@azr then writes the label's expansion preceded by a non-breakable space. 46.408 \def \azr@@\#1\{\@azr\{\#1\}\nobreakspace $\backslash$ ref $\{\# 1\}\}$
\azr@@@ Same as \azr@@ but inserts a (between the non-breakable space and the label expansion.
46.409 \def $\backslash$ azr@@@(\#1\{\@azr\{\#1\}\nobreakspace( $\backslash$ ref $\{\# 1\}\}$
\@azr Calls \hun@tempadef to choose between the label's \hun@ or original IATEX command and to expand it with accent removal and special letter substitution. Then calls \@@az, the core macro of definite article handling.

```
46.410 \def\@azr#1{%
46.411 \hun@tempadef{hun@r@#1}{r@#1}{}%
46.412 \ifx\@tempa\empty
46.413 \else
46.414 \edef\@tempb{\noexpand\@@az\expandafter\@firstoftwo\@tempa\hbox!}%
46.415 \@tempb
46.416 \fi
46.417 }
```

The following commands are used to generate the definite article for the page number of a label.
\apageref \apageref is an alias for \azp. $46.418 \backslash$ def $\backslash$ apageref $\{\backslash$ azp $\}$
$\backslash$ Apageref $\backslash$ Apageref is an alias for $\backslash$ Azp. $46.419 \backslash$ def $\backslash$ Apageref $\{\backslash$ Azp $\}$
\azp Checks if the next character is * and calls \@azp or \azp@. 46.420 \def $\backslash$ azp $\{a \backslash @ i f s t a r\{\backslash @ a z p\}\{\backslash a z p @\}\}$
$\backslash$ Azp Same as \azp except that it writes 'A' instead of 'a'. 46.421 \def $\backslash \operatorname{Azp}\{A \backslash @ i f s t a r\{\backslash @ a z p\}\{\backslash a z p @\}\}$
\azp@ Calls \@azp then writes the label's page preceded by a non-breakable space. 46.422 \def $\backslash$ azp@\#1\{\@azp\{\#1\}\nobreakspace\pageref\{\#1\}\}
\@azp Calls \hun@tempadef then takes the label's page and passes it to \@@az.

```
46.423\def\@azp#1{%
46.424 \hun@tempadef{hun@r@#1}{r@#1}{}%
46.425 \ifx\@tempa\empty
46.426 \else
46.427 \edef\@tempb{\noexpand\@@az\expandafter\@secondoftwo\@tempa\hbox!}%
46.428 \@tempb
46.429 \fi
46.430 }
```

The following macros are used to give the definite article to citations.
\acite This is an alias for \azc. 46.431 \def \acite\{\azc\}
$\backslash$ Acite This is an alias for $\backslash \mathrm{Azc}$.
46.432 \def $\backslash$ Acite $\{\backslash \mathrm{Azc}\}$
\azc Checks if the next character is a star and calls \@azc or \azc@. 46.433 \def $\backslash \mathrm{azc}\{a \backslash$ @ifstar\{ $\backslash$ @azc $\}\{$ azc@ $\}\}$
$\backslash$ Azc Same as $\backslash$ azc but used at the beginning of sentences.
$46.434 \backslash \operatorname{def} \backslash \mathrm{Azc}\{\mathrm{A} \backslash @ i f s t a r\{\backslash @ a z c\}\{\backslash \mathrm{azc}$ @ $\}\}$
\azc@ If there is no star we accept an optional argument, just like the \cite command. 46.435 \def $\backslash$ azc@\{\@ifnextchar [\{\azc@@\}\{\azc@@[]\}\}
\azc@@ First calls \@azc then \cite.

```
    46.436 \def\azc@@[#1]#2{%
```

    46.437 \@azc\{\#2\}\nobreakspace\def \@tempa\{\#1\}\%
    \(46.438 \backslash\) ifx \(\backslash\) @tempa\@empty \(\backslash\) cite \(\{\# 2\} \backslash\) else\cite[\#1] \{\#2\} \(\backslash\) fi\}
    \@azc This is an auxiliary macro to get the first cite label from a comma-separated list.

```
    46.439\def\@azc#1{\@@azc#1,\hbox!}
```

\@@azc This one uses only the first argument, that is the first element of the commaseparated 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.
46.440 \def \@@azc\#1, \#2 ${ }^{\text {hbox\#3 } \text { ! }\{\% ~}$
46.441 \hun@tempadef\{hun@b@\#1\}\{b@\#1\}\{\}\%
46.442 \ifx\@tempa\empty
46.443 \else
$46.444 \backslash$ edef \@tempb\{\noexpand \@@az\@tempa\hbox! \}\%
46.445 \@tempb
46.446 \fi\}
\hun@number@lehgth 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@lehgtharg $\backslash$ hbox $\backslash$ hbox! and \count@ must be zeroed. It is called by \@@az.
46.447 \def \hun@number@lehgth\#1\#2\hbox\#3! \{\%
46.448 \ifcat $\backslash$ noexpand\#11\%
46.449 \ifnum\expandafter'\csname\#1\endcsname>47
46.450 \ifnum \expandafter' $\backslash$ csname\#1 $\backslash$ endcsname<58
46.451 \advance $\backslash$ count@ by \@ne
$46.452 \backslash$ hun@number@lehgth\#2 $\backslash$ hbox $\backslash$ hbox! $\backslash f i \backslash f i \backslash f i\}$
\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@lehgth $a r g \backslash h b o x \backslash h b o x!$ and \count@ must be set to zero. It is called by \@@az@string.

```
46.453 \def \hun@alph@lehgth#1#2\hbox#3! {%
46.454 \ifcat\noexpand#1A%
46.455 \advance\count@ by \@ne
46.456 \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.

```
46.457 \def\@@az@string#1#2{%
46.458 \ifx#1A%
46.459 \let\@tempa\@tempb
46.460 \else\ifx#1E%
46.461 \let\@tempa\@tempb
46.462 \else\ifx#1I%
46.463 \let\@tempa\@tempb
46.464 \else\ifx#10%
46.465 \let\@tempa\@tempb
46.466 \else\ifx#1U%
46.467 \let\@tempa\@tempb
46.468 \fi\fi\fi\fi\fi
46.469 \ifx\@tempa\@tempb
46.470 \else
46.471 \count@\z@
46.472 \hun@alph@lehgth#1#2\hbox\hbox!%
46.473 \ifnum\count@=\@ne
46.474 \ifx#1F%
46.475 \let\@tempa\@tempb
46.476 \else\ifx#1L%
46.477 \let\@tempa\@tempb
46.478 \else\ifx#1M%
```

```
46.479 \let\@tempa\@tempb
46.480 \else\ifx#1N%
46.481 \let\@tempa\@tempb
46.482 \else\ifx#1R%
46.483 \let\@tempa\@tempb
46.484 \else\ifx#1S%
46.485 \let\@tempa\@tempb
46.486 \else\ifx#1X%
46.487 \let\@tempa\@tempb
46.488 \else\ifx#1Y%
46.489 \let\@tempa\@tempb
46.490 \fi\fi\fi\fi\fi\fi\fi\fi
46.491 \fi
46.492 \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 $(\bmod 3)=1($ 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 $1, L, m, M, n, N, r, R, s, S, x, X, y$, 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.

```
46.493 \def\@@az#1#2\hbox#3! {%
46.494 \let\@tempa\@empty
46.495 \def\@tempb{z}%
46.496 \uppercase{%
46.497 \ifx5#1%
46.498 \let\@tempa\@tempb
46.499 \else\ifx1#1%
46.500 \count@\@ne
46.501 \hun@number@lehgth#2\hbox\hbox!%
46.502 \loop
46.503 \ifnum\count@>\thr@@
4 6 . 5 0 4 ~ \ a d v a n c e \ c o u n t @ - \ t h r @ @ ~
46.505 \repeat
46.506 \ifnum\count@=\@ne
46.507 \let\@tempa\@tempb
46.508 \fi
46.509 \else
46.510 \@@az@string{#1}{#2}%
46.511 \fi\fi
46.512 }%
46.513 \@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.

```
46.514 \expandafter\addto\csname extras\CurrentOption\endcsname{%
46.515 \babel@save\refstepcounter
46.516 \def\refstepcounter#1{\stepcounter{#1}%
46.517 \def\@currentlabel{\csname p@#1\endcsname\csname the#1\endcsname}}%
46.518 }
```

\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.

```
46.519 \expandafter\addto\csname extras\CurrentOption\endcsname{%
46.520 \let\old@label\label
46.521 \def\label#1{\@bsphack
46.522 \old@label{#1}%
46.523 \begingroup
46.524 \let\romannumeral\number
46.525 \def\@roman##1{\number ##1}%
46.526 \def\@Roman##1{\number ##1}%
46.527 {\toks0={\noexpand\noexpand\noexpand\number}%
\def\number##1{\the\toks0 ##1}\xdef\tempb@{\thepage}}%
\edef\@tempa##1{\noexpand\protected@write\@auxout{}%
                                    {\noexpand\string\noexpand\hunnewlabel
                                    {##1}{{\@currentlabel}{\tempb@}}}}%
            \@tempa{#1}%
        \endgroup
    \@esphack}%
46.535 }
46.536 \expandafter\addto\csname noextras\CurrentOption\endcsname{%
46.537 \let\label\old@label
46.538 }
```

\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.
46.539 \def $\backslash$ hunnewlabel\#1\#2\{\%
46.540 \def\@tempa\{\#2\}\%
46.541 \expandafter\ifx\csname r@\#1\endcsname\@tempa
$46.542 \backslash r e l a x \%$ \message\{No need for def: \#1\}\%
46.543 \else
$46.544 \backslash$ lobal \expandafter\let\csname hun@r@\#1\endcsname\@tempa\%
46.545 \fi
46.546 \}

For Hungarian the ' character is made active.

```
46.547\AtBeginDocument{%
46.548 \if@filesw\immediate\write\@auxout{\catcode096=12}\fi}
46.549 \initiate@active@char{`}
46.550 \expandafter\addto\csname extras\CurrentOption\endcsname{%
46.551 \languageshorthands{magyar}%
46.552 \bbl@activate{`}}
46.553\expandafter\addto\csname noextras\CurrentOption\endcsname{%
46.554 \bbl@deactivate{`}}
```

The character sequence ' ' is declared as a shorthand in order to produce the opening quotation sign appropriate for Hungarian.
46.555 \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.
46.556 \declare@shorthand\{magyar\}\{‘c\}\{\textormath\{\bbl@disc\{c\}\{cs\}\}\{c\}\}
46.557 \declare@shorthand\{magyar\}\{‘C\}\{\textormath\{\bbl@disc\{C\}\{CS\}\}\{C\}\}
46.558 \declare@shorthand\{magyar\}\{'d\}\{\textormath\{\bbl@disc\{d\}\{dz\}\}\{d\}\}
46.559 \declare@shorthand\{magyar\}\{‘D\}\{\textormath\{\bbl@disc\{D\}\{DZ\}\}\{D\}\}
46.560 \declare@shorthand\{magyar\}\{'g\}\{\textormath\{ \bbl@disc\{g\}\{gy\}\}\{g\}\}
46.561 \declare@shorthand\{magyar\}\{‘G\}\{\textormath\{\bbl@disc\{G\}\{GY\}\}\{G\}\}

46.563 \declare@shorthand\{magyar\}\{'L\}\{\textormath\{ ${ }^{\text {b bbl@disc\{L\} }}$ \{LY\}\}\{L\}\}
46.564 \declare@shorthand\{magyar\}\{'n\}\{\textormath\{\bbl@disc\{n\}\{ny\}\}\{n\}\}
46.565 \declare@shorthand\{magyar\}\{'N\}\{\textormath\{\bbl@disc\{N\}\{NY\}\}\{N\}\}
46.566 \declare@shorthand\{magyar\}\{'s\}\{\textormath\{\bbl@disc\{s\}\{sz\}\}\{s\}\}
46.567 \declare@shorthand\{magyar\}\{'S\}\{\textormath\{\bbl@disc\{S\}\{SZ\}\}\{S\}\}
46.568 \declare@shorthand\{magyar\}\{'t\}\{\textormath\{\bbl@disc\{t\}\{ty\}\}\{t\}\}
46.569 \declare@shorthand\{magyar\}\{'T\}\{\textormath\{\bbl@disc\{T\}\{TY\}\}\{T\}\}
46.570 \declare@shorthand\{magyar\}\{'z\}\{\textormath\{\bbl@disc\{z\}\{zs\}\}\{z\}\}
46.571 \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 $\backslash$ begin\{document\} and resetting the category code of © to its original value.
$46.572 \backslash l d f @ f i n i s h \backslash C u r r e n t O p t i o n$
46.573 〈/code〉

## 47 The Estonian language

The file estonian. $\mathrm{dtx}^{55}$ 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 ( $\ddot{a}, \ddot{0}, \ddot{u}$ ), 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 21. 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;
"- an explicit hyphen sign, allowing hyphenation in the
    rest of the word;
\- like the old \-, but allowing hyphenation in the rest
    of the word;
"، for Estonian low left double quotes (same as Ger-
    man);
" ' for Estonian right double quotes;
"< for French left double quotes (also rather popular)
"> for French right double quotes.
```

Table 21: 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 tenc\}.Thispackagemustbeloadedbeforebabel.AsthestandardEstonianhyphenationfileeehyph.texisintheCorkencoding,choosingthisencodingwillgiveyoubetterhyphenation.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

As mentioned in the Spanish style file, it may happen that some packages fail (usually in a aseyoushouldchangetheorderofthe\usepackagedeclarationsortheorderofthestyleoptionsin\documentclass.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

[^45]
### 47.1 Implementation

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

```
47.1 〈*code〉
47.2\LdfInit{estonian}\captionsestonian
```

If Estonian is not included in the format file (does not have hyphenation patterns), we shall use English hyphenation.

```
47.3\ifx\l@estonian\@undefined
47.4 \@nopatterns{Estonian}
47.5 \adddialect\l@estonian0
47.6\fi
```

Now come the commands to switch to (and from) Estonian.
\captionsestonian The macro \captionsestonian defines all strings used in the four standard documentclasses provided with $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
47.7 \addto\captionsestonian{%
47.8\\def\prefacename{Sissejuhatus}%
47.9\def\refname{Viited}%
47.10 \def\bibname{Kirjandus}%
47.11 \def\appendixname{Lisa}%
47.12\def\contentsname{Sisukord}%
47.13\\def\listfigurename{Joonised}%
47.14 \def\listtablename{Tabelid}%
47.15\def\indexname{Indeks}%
47.16 \def\figurename{Joonis}%
47.17 \def\tablename{Tabel}%
47.18\\def\partname{0sa}%
47.19 \def\enclname{Lisa(d)}%
47.20 \def\ccname{Koopia(d)}%
47.21 \def\headtoname{}%
47.22\\def\pagename{Lk.}%
47.23\def\seename{vt.}%
47.24\def\alsoname{vt. ka}%
47.25\def\proofname{Korrektuur}%
47.26 \def\glossaryname{Glossary}% <-- Needs translation
47.27 }
```

These captions contain accented characters.

```
47.28\begingroup \catcode'\\"\active
47.29 \def\x{\endgroup
47.30 \addto\captionsestonian{%
47.31 \def\abstractname{Kokkuv~ote}%
47.32 \def\chaptername{Peat"ukk}}}
47.33\x
```

\dateestonian The macro \dateestonian redefines the command \today to produce Estonian dates.

```
47.34 \begingroup \catcode"\"\active
47.35\def\x{\endgroup
47.36 \def\month@estonian{\ifcase\month\or
47.37 jaanuar\or veebruar\or m"arts\or aprill\or mai\or juuni\or
47.38 juuli\or august\or september\or oktoober\or november\or
47.39 detsember\fi}}
47.40\x
47.41 \def\dateestonian{%
47.42 \def\today{\number\day.\space\month@estonian
47.43 \space\number\year.\space a.}}
```

\extrasestonian The macro \extrasestonian will perform all the extra definitions needed for \noextrasestonian Estonian. The macro \noextrasestonian is used to cancel the actions of \extrasestonian. For Estonian, " is made active and has to be treated as 'special' ( $\sim$ is active already).

```
47.44 \initiate@active@char{"}
47.45 \initiate@active@char{~}
47.46 \addto\extrasestonian{\languageshorthands{estonian}}
47.47\addto\extrasestonian{\bbl@activate{"}\bbl@activate{~
```

Store the original macros, and redefine accents.
47.48 \addto \extrasestonian\{ \babel@save\"\umlautlow $\backslash$ babel@save \~\tildelow\}

Estonian does not use extra spaces after sentences.
47.49 \addto $\backslash$ extrasestonian\{ $\backslash$ bbl@frenchspacing $\}$
47.50 \addto\noextrasestonian\{\bbl@nonfrenchspacing\}
\estonianhyphenmins For Estonian, \lefthyphenmin and \righthyphenmin are both 2.
47.51 \providehyphenmins\{\CurrentOption\}\{\tw@\tw@\}
\tildelow The standard TEX accents are too high for Estonian typography, we have to lower \gentilde them (following the babel German style). For a detailed explanation see the file $\backslash$ newtilde glyphs.dtx.
\newcheck $47.52 \backslash$ def $\backslash$ tildelow\{ ${ }^{\text {def } \backslash \sim\{\text { protect } \backslash \text { gentilde }\}\}}$
$47.53 \backslash \operatorname{def} \backslash g e n t i l d e \# 1\{\backslash i f \# 10 \backslash$ newtilde\{\#1\}\else\if\#10\newtilde\{\#1\}\%
$47.54 \backslash e l s e \backslash n e w c h e c k\{\# 1\} \%$
47.55 \fi\fi\}
$47.56 \backslash$ def $\backslash$ newtilde\#1\{\leavevmode $\backslash$ allowhyphens
47.57 \{ $\backslash \mathrm{UQD} 1 \mathrm{ex} \%$
$47.58\{\backslash$ setbox $\backslash z @ \backslash h b o x\{\backslash \operatorname{char} 126\} \backslash d i m e n @-.45 e x \backslash a d v a n c e \backslash d i m e n @ \backslash h t \backslash z @$
47.59 \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi\}\%
47.60 \accent126\fontdimen5\font\U@D \#1\}\allowhyphens\}
47.61 \def $\backslash$ newcheck\#1\{\leavevmode $\backslash$ allowhyphens
47.62 \{ $\backslash U @ D ~ 1 e x \%$
$47.63\{\backslash$ setbox $\backslash \mathbf{z @} \backslash h b o x\{\backslash$ char20\}\dimen@ -.45ex\advance\dimen@\ht $\backslash z @$
47.64 \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi\}\%
47.65 \accent20\fontdimen5\font\U@D \#1\}\allowhyphens\}

We save the double quote character in \dq, and tilde in $\backslash t i l$, and store the original definitions of $\backslash "$ and $\backslash^{\sim}$ as $\backslash$ dieresis and \texttilde.

```
47.66 \begingroup \catcode`\"12
47.67\edef\x{\endgroup
47.68 \def\noexpand\dq{"}
47.69 \def\noexpand\til{~
47.70 \x
47.71 \let\dieresis\"
47.72\let\texttilde\~
```

This part follows closely spanish.ldf. We check the encoding and if it is T1, we have to tell $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ about our redefined accents.

```
47.73 \ifx\f@encoding \(\backslash\) bbl@t@one
47.74 \let\@umlaut\dieresis
47.75 \let\@tilde\texttilde
47.76 \DeclareTextComposite\{\~\}\{T1\}\{s\}\{178\}
47.77 \DeclareTextComposite\{\~\}\{T1\}\{S\}\{146\}
47.78 \DeclareTextComposite \(\{\backslash \sim\{T 1\}\{z\}\{186\}\)
47.79 \DeclareTextComposite\{\~\}\{T1\}\{Z\}\{154\}
47.80 \DeclareTextComposite\{\"\}\{T1\}\{'\}\{17\}
47.81 \DeclareTextComposite\{\"\}\{T1\}\{‘\}\{18\}
47.82 \DeclareTextComposite\{\"\}\{T1\}\{<\}\{19\}
47.83 \DeclareTextComposite\{\"\}\{T1\}\{>\}\{20\}
```

If the encoding differs from T1, we expand the accents, enabling hyphenation beyond the accent. In this case $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ will not find all possible breaks, and we have to warn people.

```
47.84 \else
47.85 \wlog{Warning: Hyphenation would work better for the T1 encoding.}
47.86 \let\@umlaut\newumlaut
47.87 \let\@tilde\gentilde
47.88 \fi
```

Now we define the shorthands.

```
47.89\declare@shorthand{estonian}{"a}{\textormath{\"{a}}{\ddot a}}
47.90\declare@shorthand{estonian}{"A}{\textormath{\"{A}}{\ddot A}}
47.91 \declare@shorthand{estonian}{"o}{\textormath{\"{o}}{\ddot o}}
47.92\declare@shorthand{estonian}{"O}{\textormath{\"{0}}{\ddot 0}}
47.93\declare@shorthand{estonian}{"u}{\textormath{\"{u}}{\ddot u}}
47.94\declare@shorthand{estonian}{"U}{\textormath{\"{U}}{\ddot U}}
```

german and french quotes,

```
47.95\declare@shorthand{estonian}{"'}{%
47.96 \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
47.97 \declare@shorthand{estonian}{"'}{%
47.98 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
47.99 \declare@shorthand{estonian}{"<}{%
47.100 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
47.101 \declare@shorthand{estonian}{">}{%
47.102 \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

```
47.103\declare@shorthand{estonian}{~o}{\textormath{\@tilde o}{\tilde o}}
47.104 \declare@shorthand{estonian}{~0}{\textormath{\@tilde 0}{\tilde 0}}
47.105\declare@shorthand{estonian}{~ s}{\textormath{\@tilde s}{\check s}}
47.106 \declare@shorthand{estonian}{~S}{\textormath{\@tilde S}{\check S}}
47.107\declare@shorthand{estonian}{~z}{\textormath{\@tilde z}{\check z}}
47.108 \declare@shorthand{estonian}{~Z}{\textormath{\@tilde Z}{\check Z}}
    and some additional commands:
47.109 \declare@shorthand{estonian}{"-}{\nobreak\-\bbl@allowhyphens}
47.110 \declare@shorthand{estonian}{"|}{%
47.111 \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
47.112 \allowhyphens}{}}
47.113 \declare@shorthand{estonian}{""}{\dq}
47.114 \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.

```
47.115\ldf@finish{estonian}
47.116 </code\rangle
```


## 48 The Croatian language

The file croatian. $\mathrm{dtx}^{56}$ 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.

```
48.1 <*code\rangle
48.2 \LdfInit{croatian}\captionscroatian
```

When this file is read as an option, i.e. by the command,croatianwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\l@croatiantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
48.3\ifx\l@croatian\@undefined
48.4 \@nopatterns{Croatian}
48.5 \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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
48.6 \addto\captionscroatian\{\%
48.7 \def\prefacename\{Predgovor\}\%
48.8 \def \(\backslash r e f n a m e\{L i t e r a t u r a\} \%\)
48.9 \def \(\backslash\) abstractname\{Sa\v\{z\}etak\}\%
\(48.10 \backslash d e f \backslash\) bibname\{Bibliografija\}\%
48.11 \def\chaptername\{Poglavlje\}\%
48.12 \def \appendixname\{Dodatak\}\%
\(48.13 \backslash\) def \(\backslash c o n t e n t s n a m e\{S a d r \backslash v\{z\} a j\} \%\)
\(48.14 \backslash d e f \backslash l i s t f i g u r e n a m e\{P o p i s ~ s l i k a\} \%\)
48.15 \def \listtablename\{Popis tablica\}\%
48.16 \def \indexname\{Indeks\}\%
48.17 \def \(\backslash f i g u r e n a m e\{S l i k a\} \%\)
48.18 \def\tablename\{Tablica\}\%
48.19 \def\partname\{Dio\}\%
48.20 \def \enclname\{Prilozi\}\%
48.21 \def \ccname\{Kopije\}\%
48.22 \def \(\backslash\) headtoname\{Prima\}\%
\(48.23 \backslash\) def \(\backslash\) pagename\{Stranica\}\%
48.24 \def \seename\{Vidjeti\}\%
48.25 \def \(\backslash\) alsoname\{Vidjeti i\}\%
48.26 \def\proofname\{Dokaz\}\%
48.27 \def \(\backslash g l o s s a r y n a m e\{K a z a l o\} \%\)
48.28 \}\%
```

[^46]\datecroatian The macro \datecroatian redefines the command \today to produce Croatian dates.

```
48.29 \def\datecroatian{%
48.30 \def\today{\number\day.~\ifcase\month\or
48.31 sije\v{c}nja\or velja\v{c}e\or o\v{z}ujka\or travnja\or svibnja\or
48.32 lipnja\or srpnja\or kolovoza\or rujna\or listopada\or studenog\or
48.33 prosinca\fi \space \number\year.}}
```

\extrascroatian The macro \extrascroatian will perform all the extra definitions needed for the \noextrascroatian 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.

```
48.34 \addto\extrascroatian{}
48.35 \addto\noextrascroatian{}
```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of @ to its original value.
48.36 \ldf@finish\{croatian\}
48.37 〈/code〉

## 49 The Czech language

The file czech.dtx ${ }^{57}$ defines all the language definition macros for the Czech language.
 access to two accents are defined.

The command $\backslash q$ is used with the letters ( $\mathrm{t}, \mathrm{d}, \mathrm{l}$, and L ) and adds a' to them to simulate a 'hook' that should be there. The result looks like t'. The command \w is used to put the ring-accent which appears in ångstrøm over the letters $u$ and U.

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

```
49.1 〈*code〉
49.2\LdfInit{czech}\captionsczech
```

When this file is read as an option, i.e. by the command,czechwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@czechtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
49.3\ifx\l@czech\@undefined
49.4 \@nopatterns{Czech}
49.5 \adddialect\l@czech0\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 $\mathrm{EAT}_{\mathrm{E}}$.

```
.6\addto\captionsczech{%
49.7 \def\prefacename{P\v redmluva}%
49.8\def\refname{Reference}%
49.9 \def\abstractname{Abstrakt}%
49.10 \def\bibname{Literatura}%
49.11 \def\chaptername{Kapitola}%
49.12\def\appendixname{Dodatek}%
49.13\def\contentsname{Obsah}%
49.14\def\listfigurename{Seznam obr\'azk\r{u}}%
49.15\def\listtablename{Seznam tabulek}%
49.16 \def\indexname{Index}%
49.17\def\figurename{Obr\'azek}%
49.18\def\tablename{Tabulka}%
49.19 \def\partname{\v{C}\'ast}%
49.20\def\enclname{P\v{r}\'{\i}loha}%
49.21 \def\ccname{Na v\v{e}dom\'{\i}:}%
49.22 \def\headtoname{Komu}%
49.23\def\pagename{Strana}%
49.24\def\seename{viz}%
```

[^47]```
49.25 \def\alsoname{viz tak\'e}%
49.26 \def\proofname{D\r{u}kaz}%
49.27 \def\glossaryname{Glos\'a\v r}%
49.28 }%
```

\dateczech The macro \dateczech redefines the command \today to produce Czech dates.

```
49.29\def\dateczech{%
49.30\\def\today{\number\day. \\ifcase\month\or
49.31 ledna\or \'unora\or b\v{r}ezna\or dubna\or kv\v{e}tna\or
49.32 \v{c}ervna\or \v{c}ervence\or srpna\or z\'a\v{r}\'{\i}\or
49.33\v{r}\'{\i}jna\or listopadu\or prosince\fi
49.34 \space \number\year}}
```

\extrasczech The macro \extrasczech will perform all the extra definitions needed for the
\noextrasczech 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.

```
49.35\addto\extrasczech{\babel@save\q\let\q\v}
49.36 \addto\extrasczech{\babel@save\w\let\w\r}
```

For Czech texts \frenchspacing should be in effect. We make sure this is the case and reset it if necessary.

```
49.37 \addto\extrasczech{\bbl@frenchspacing}
49.38 \addto\noextrasczech{\bbl@nonfrenchspacing}
```

\v $\mathrm{LAT}_{\mathrm{E}}$ 's normal \v accent places a caron over the letter that follows it (ǒ). This is not what we want for the letters $d, t, l$ and $L$; for those the accent should change shape. This is acheived by the following.

```
49.39 \AtBeginDocument\{\%
49.40 \DeclareTextCompositeCommand\{\v\}\{0T1\}\{t\}\{\%
\(49.41 \quad \mathrm{t} \backslash \mathrm{kern}\)-. 23em \(\backslash\) raise. 24ex \(\backslash\) hbox\{'\}\}
49.42 \DeclareTextCompositeCommand\{\v\}\{0T1\}\{d\}\{\%
\(49.43 \quad \mathrm{~d} \backslash\) kern-. 13em \(\backslash\) raise. \(24 \mathrm{ex} \backslash\) hbox \(\{\) ' \(\}\}\)
49.44 \DeclareTextCompositeCommand\{\v\}\{0T1\}\{l\}\{\lcaron\{\}\}
\(49.45 \backslash\) DeclareTextCompositeCommand\{\v\}\{0T1\}\{L\}\{\Lcaron\{\}\}\}
```

\lcaron Fot the letters 1 and $L$ we want to disinguish between normal fonts and monospaced \Lcaron fonts.

| 49.46 \def $\backslash 1$ caron\{\% |  |
| :---: | :---: |
| 49.47 | \setbox0\hbox\{M\} ${ }^{\text {setbox }}$ \tw@ ${ }^{\text {a }}$ hbox\{i\}\% |
| 49.48 | \ifdim\wd0>\wd \tw@\relax |
| 49.49 | $1 \backslash$ kern-.13em\raise. 24 ex \hbox\{'\}\kern-.11em\% |
| 49.50 | $\backslash \mathrm{else}$ |
| 49.51 |  |
| 49.52 | \fi\} |
| 49.53 \def $\backslash$ Lcaron $\{\%$ |  |
| 49.54 | \setbox0\hbox\{M\} ${ }^{\text {setbox }}$ \tw@ ${ }^{\text {a }}$ hbox\{i\}\% |
| 49.55 | \ifdim\wd0>\wd\tw@\relax |
| 49.56 | L\raise. 24 ex \hbox to $\backslash \mathrm{z@}$ \} \backslash \mathrm { kern }  -.  2 8 \mathrm { em }  '  \backslash \mathrm { hss } \} \% |

```
49.57 \else
49.58 L\raise.45ex\hbox to\z@{\kern-.40em '\hss}%
49.59 \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.
49.60 \ldf@finish\{czech\}
49.61 〈/code〉

## 50 The Polish language

The file polish. $\mathrm{dtx}^{58}$ defines all the language-specific macros for the Polish language.

For this language the character " is made active. In table 22 an overview is given of its purpose.

```
"a or \aob, for tailed-a (like a̧)
"A or \Aob, for tailed-A (like A̧)
"e or \eob, for tailed-e (like ȩ)
"E or \Eob, for tailed-E (like Ȩ)
"c or \'c, for accented c (like ć), same with uppercase
    letters and \(\mathrm{n}, \mathrm{o}, \mathrm{s}\)
" 1 or \(\backslash \mathrm{lpb}\}\), for 1 with stroke (like \(ł\) )
"L or \(\backslash \mathrm{Lpb}\}\), for L with stroke (like E )
" r or \(\backslash \mathrm{zkb}\}\), for pointed z (like \(\dot{\mathrm{z}}\) ), cf. pronounciation
"R or \(\backslash \mathrm{Zkb}\}\), for pointed Z (like Ż)
" z or \'z, for accented z
" \(Z\) or \(\backslash\) ' \(Z\), for accented \(Z\)
"| disable ligature at this position.
" - an explicit hyphen sign, allowing hyphenation in the
    rest of the word.
"" like "-, but producing no hyphen sign (for compund
    words with hyphen, e.g. \(x-"\) "y).
"‘ for German left double quotes (looks like „).
", for German right double quotes.
" \(<\) for French left double quotes (similar to \(\ll\) ).
"> for French right double quotes (similar to \(\gg\) ).
```

Table 22: 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.

```
50.1 <*code〉
```

50.2 \LdfInit\{polish\}\captionspolish

When this file is read as an option, i.e. by the command,polishcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof$\backslash1@polish~to~see~whether~we~have~to~do~something~here.~$undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
50.3\ifx\l@polish\@undefined
50.4 \@nopatterns{Polish}
50.5 \adddialect\l@polish0\fi
```

The next step consists of defining commands to switch to (and from) the Polish language.

[^48]\captionspolish The macro \captionspolish defines all strings used in the four standard documentclasses provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
50.6 \addto\captionspolish{%
50.7\def\prefacename{Przedmowa}%
50.8\def\refname{Literatura}%
50.9\def\abstractname{Streszczenie}%
50.10 \def\bibname{Bibliografia}%
50.11 \def\chaptername{Rozdzia\l}%
50.12 \def \appendixname{Dodatek}%
50.13 \def\contentsname{Spis tre\'sci}%
50.14 \def\listfigurename{Spis rysunk\'ow}%
50.15 \def\listtablename{Spis tablic}%
50.16\def\indexname{Indeks}%
50.17 \def\figurename{Rysunek}%
50.18\def\tablename{Tablica}%
50.19 \def\partname{Cz\eob{}\'s\'c}%
50.20 \def\enclname{Za\l\aob{}cznik}%
50.21 \def\ccname{Kopie:}%
50.22\def\headtoname{Do}%
50.23 \def\pagename{Strona}%
50.24\def\seename{Por\'ownaj}%
50.25\def\alsoname{Por\'ownaj tak\.ze}%
50.26 \def\proofname{Dow\'od}%
50.27 \def\glossaryname{Glossary}% <-- Needs translation
50.28}
```

\datepolish The macro \datepolish redefines the command \today to produce Polish dates.

```
50.29\def\datepolish{%
50.30 \def\today{\number\day~\ifcase\month\or
50.31 stycznia\or lutego\or marca\or kwietnia\or maja\or czerwca\or lipca\or
50.32 sierpnia\or wrze\'snia\or pa\'zdziernika\or listopada\or grudnia\fi
50.33 \space\number\year}%
50.34}
```

\extraspolish The macro \extraspolish will perform all the extra definitions needed for the \noextraspolish Polish language. The macro \noextraspolish is used to cancel the actions of \extraspolish.

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.

```
50.35 \initiate@active@char{"}
50.36 \addto\extraspolish{\languageshorthands{polish}}
50.37\addto\extraspolish{\bbl@activate{"}}
```

Don't forget to turn the shorthands off again.
50.38 \addto\noextraspolish\{\bbl@deactivate\{"\}\}

The code above is necessary because we need an extra active character. This character is then used as indicated in table 22.

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 forcasting 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.

```
50.39 \newdimen\pl@left
50.40 \newdimen\pl@down
50.41 \newdimen\pl@right
50.42 \newdimen\pl@temp
```

\sob The macro \sob is used to put the 'ogonek' in the right place.

```
50.43 \def\sob#1#2#3#4#5{%parameters: letter and fractions hl,ho,vl,vo
50.44 \setbox0\hbox{#1}\setbox1\hbox{$_\mathchar'454$}\setbox2\hbox{p}%
50.45 \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
50.46 \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
50.47 \pl@left=\pl@right \advance\pl@left by\wd1
50.48 \pl@temp=-\pl@down \advance\pl@temp by\dp2 \dp1=\pl@temp
50.49 \leavevmode
50.50 \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
```

\aob The ogonek is placed with the letters ' a ', ' A ', 'e', and ' E '.
$\backslash$ Aob $_{50.51} \backslash$ DeclareTextCommand $\{$ aob $\}\{0 T 1\}\{\backslash$ sob a\{.66\}\{.20\}\{0\}\{.90\}\}
\eob50.52 \DeclareTextCommand $\{\backslash$ Aob $\}\{0 T 1\}\{\backslash$ sob A\{.80\}\{.50\}\{0\}\{.90\}\}
\Eob50.53 \DeclareTextCommand\{\eob\}\{0T1\}\{\sob e\{.50\}\{.35\}\{0\}\{.93\}\}
50.54 \DeclareTextCommand\{\EOb\}\{0T1\}\{\sob E\{.60\}\{.35\}\{0\}\{.90\}\}

For the 'new' T1 encoding we can provide simpler definitions.
$50.55 \backslash$ DeclareTextCommand $\{\backslash \mathrm{aob}\}\{T 1\}\{\backslash \mathrm{k}$ a\}
$50.56 \backslash$ DeclareTextCommand $\{\backslash$ Aob $\}\{T 1\}\{\backslash \mathrm{k} \mathrm{A}\}$
50.57 \DeclareTextCommand $\{\backslash$ eob $\}\{T 1\}\{\backslash \mathrm{k}$ e\}
50.58 \DeclareTextCommand\{<br>Eob\}\{T1\}\{\k E\}

Construct the characters by default from the OT1 encoding.
50.59 \ProvideTextCommandDefault\{ $\operatorname{\text {aob}\} \{ \backslash UseTextSymbol\{ 0T1\} \{ \backslash aob\} \} }$
$50.60 \backslash$ ProvideTextCommandDefault $\{\backslash$ Aob $\}\{\backslash$ UseTextSymbol $\{0 \mathrm{~T} 1\}\{\backslash$ Aob $\}\}$
50.61 \ProvideTextCommandDefault\{ $\backslash e o b\}\{\backslash$ UseTextSymbol $\{0 T 1\}\{\backslash e o b\}\}$
$50.62 \backslash$ ProvideTextCommandDefault $\{\backslash$ Eob $\}\{\backslash$ UseTextSymbol $\{0$ T1\} $\{\backslash$ Eob $\}$
$\backslash \mathrm{spb}$ The macro \spb is used to put the 'poprzeczka' in the right place.
50.63 \def \spb\#1\#2\#3\#4\#5\{\%
50.64 \setbox0\hbox\{\#1\}\setbox1 \hbox\{\char'023\}\%
50.65 \pl@right=\#2\wd0 \advance\pl@right by-\#3\wd1
50.66 \pl@down=\#5\ht1 \advance\pl@down by-\#4\ht0
50.67 \pl@left=\pl@right \advance\pl@left by\wd1
50.68 \ht1=\pl@down \dp1=-\pl@down

```
50.69 \leavevmode
50.70 \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.

```
50.71 \def\skb#1#2#3#4#5{%
50.72 \setbox0\hbox{#1}\setbox1\hbox{\char'056}%
50.73 \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
50.74 \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
50.75 \pl@left=\pl@right \advance\pl@left by\wd1
50.76 \leavevmode
50.77 \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.

```
50.78 \def\textpl{%
50.79 \def\lpb{\plll}%
50.80 \def\Lpb{\pLLL}%
50.81 \def\zkb{\.z}%
50.82 \def\Zkb{\.Z}}
```

Initially we assume that typesetting is done with text fonts.

```
50.83 \textpl
50.84 \let\lll=\l \let\LLL=\L
50.85\def\plll{\lll}
50.86\def\pLLL{\LLL}
```

\telepl But for the 'teletype' font in 'OT1' encoding we have to take some special actions, involving the macros defined above.

```
50.87\def\telepl{%
50.88\\def\lpb{\spb l{.45}{.5}{.4}{.8}}%
50.89 \def\Lpb{\spb L{.23}{.5}{.4}{.8}}%
50.90\def\zkb{\skb z{.5}{.5}{1.2}{0}}%
50.91 \def\Zkb{\skb Z{.5}{.5}{1.1}{0}}}
```

To activate these codes the font changing commands as they are defined in ${ }^{\mathrm{LA}} \mathrm{T}_{\mathrm{E}} \mathrm{X}$ are modified. The same is done for plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 's font changing commands.

When \selectfont is undefined the current format is spposed to be either plain (based) or $\mathrm{L}^{\mathrm{A}} \mathrm{T}_{\mathrm{E}}$ 2.09.

```
50.92\ifx\selectfont\@undefined
50.93 \ifx\prm\@undefined \addto\rm{\textpl}\else \addto\prm{\textpl}\fi
50.94 \ifx\pit\@undefined \addto\it{\textpl}\else \addto\pit{\textpl}\fi
50.95\ifx\pbf\@undefined \addto\bf{\textpl}\else \addto\pbf{\textpl}\fi
50.96 \ifx\psl\@undefined \addto\sl{\textpl}\else \addto\psl{\textpl}\fi
50.97 \ifx\psf\@undefined \else \addto\psf{\textpl}\fi
50.98 \ifx\psc\@undefined \else \addto\psc{\textpl}\fi
50.99 \ifx\ptt\@undefined \addto\tt{\telepl}\else \addto\ptt{\telepl}\fi
50.100\else
```

When \selectfont exists we assume $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$.
50.101 \expandafter $\backslash$ addto $\backslash c s n a m e ~ s e l e c t f o n t ~ \ e n d c s n a m e\{\% ~$
50.102 \csname\f@encoding @pl\endcsname\}
50.103 \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.

```
50.104 \expandafter\let\csname T1@pl\endcsname\textpl
    \@family is not.
50.105 \expandafter\def\csname OT1@pl\endcsname{%
50.106 \long\edef\curr@family{\f@family}%
50.107 \ifx\curr@family\ttdefault
50.108 \telepl
50.109 \else
50.110 \textpl
50.111 \fi}
```

    For OT1 we need to check the current font family, stored in \(\backslash f @ f a m i l y\). Un-
    fortunately we need a hack as \ttdefault is defined as a \long macro, while
    \dq We save the original double quote character in \dq to keep it available, the math accent \" can now be typed as ".

```
50.112 \begingroup \catcode'\"12
50.113 \def\x{\endgroup
50.114 \def\dq{"}}
50.115 \x
```

Now we can define the doublequote macros for diacritics,

```
50.116 \declare@shorthand{polish}{"a}{\textormath{\aob}{\ddot a}}
50.117\declare@shorthand{polish}{"A}{\textormath{\Aob}{\ddot A}}
50.118 \declare@shorthand{polish}{"c}{\textormath{\'c}{\acute c}}
50.119\declare@shorthand{polish}{"C}{\textormath{\'C}{\acute C}}
50.120 \declare@shorthand{polish}{"e}{\textormath{\eob}{\ddot e}}
50.121 \declare@shorthand{polish}{"E}{\textormath{\Eob}{\ddot E}}
50.122 \declare@shorthand{polish}{"l}{\textormath{\lpb}{\ddot l}}
50.123\declare@shorthand{polish}{"L}{\textormath{\Lpb}{\ddot L}}
50.124 \declare@shorthand{polish}{"n}{\textormath{\'n}{\acute n}}
50.125 \declare@shorthand{polish}{"N}{\textormath{\'N}{\acute N}}
50.126 \declare@shorthand{polish}{"o}{\textormath{\'o}{\acute o}}
50.127 \declare@shorthand{polish}{"O}{\textormath{\'O}{\acute 0}}
50.128\declare@shorthand{polish}{"s}{\textormath{\'s}{\acute s}}
50.129 \declare@shorthand{polish}{"S}{\textormath{\'S}{\acute S}}
```

$\backslash$ polishrz The command $\backslash$ polishrz defines the shorthands "r, "z and "x to produce pointed $\backslash$ polishzx z , accented z and "x. This is the default as these shorthands were defined by this language definition file for quite some time.

```
50.130 \newcommand*{\polishrz}{%
50.131 \declare@shorthand{polish}{"r}{\textormath{\zkb}{\ddot r}}%
50.132 \declare@shorthand{polish}{"R}{\textormath{\Zkb}{\ddot R}}%
```

```
50.133 \declare@shorthand{polish}{"z}{\textormath{\'z}{\acute z}}%
50.134 \declare@shorthand{polish}{"Z}{\textormath{\'Z}{\acute Z}}%
50.135 \declare@shorthand{polish}{"x}{\dq x}%
50.136 \declare@shorthand{polish}{"X}{\dq X}%
50.137 }
50.138 \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.

```
50.139 \newcommand*{\polishzx}{%
50.140 \declare@shorthand{polish}{"z}{\textormath{\zkb}{\ddot z}}%
50.141 \declare@shorthand{polish}{"Z}{\textormath{\Zkb}{\ddot Z}}%
50.142 \declare@shorthand{polish}{"x}{\textormath{\'z}{\acute x}}%
50.143 \declare@shorthand{polish}{"X}{\textormath{\'Z}{\acute X}}%
50.144 \declare@shorthand{polish}{"r}{\dq r}%
50.145 \declare@shorthand{polish}{"R}{\dq R}%
50.146 }
```

Then we define access to two forms of quotation marks, similar to the german and french quotation marks.

```
50.147 \declare@shorthand{polish}{"‘}{%
50.148 \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
50.149 \declare@shorthand{polish}{"'}{%
50.150 \textormath{\textquotedblright}{\mbox{\textquotedblright}}}
50.151 \declare@shorthand{polish}{"<}{%
50.152 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
50.153 \declare@shorthand{polish}{">}{%
50.154 \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behavew a little different from $\backslash$-.

```
50.155\declare@shorthand{polish}{"-}{\nobreak-\bbl@allowhyphens}
```

50.156 \declare@shorthand\{polish\}\{""\}\{\hskip\z@skip\}

And we want to have a shorthand for disabling a ligature.

```
50.157 \declare@shorthand{polish}{"|}{%
```

50.158 \textormath\{\discretionary\{-\}\{\}\{\kern.03em\}\}\{\}\}
\mdqon All that's left to do now is to define a couple of commands for reasons of compat\mdqoff ibility with polish.tex.
50.159 \def $\backslash$ mdqon $\{\backslash$ shorthandon\{" $\}$ \}
$50.160 \backslash$ def $\backslash$ mdqoff $\{\backslash$ 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.

```
50.161 \ldf@finish{polish}
```

50.162 〈/code〉

## 51 The Serbocroatian language

The file serbian. $\mathrm{dtx}^{59}$ 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 23 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 compund 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 $\ll$ ).
" $>$ for French right double quotes (similar to $\gg$ ).
Table 23: 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.

```
51.1 <*code\rangle
51.2\LdfInit{serbian}\captionsserbian
```

When this file is read as an option, i.e. by the command,serbianwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@serbiantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
51.3\ifx\l@serbian\@undefined
51.4 \@nopatterns{Serbian}
51.5 \adddialect\l@serbian0\fi
```

The next step consists of defining commands to switch to (and from) the Serbocroatian language.

[^49]\captionsserbian The macro \captionsserbian defines all strings used in the four standard documentclasses provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
51.6 \addto\captionsserbian{%
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{Glava}%
51.12\def\appendixname{Dodatak}%
51.13 \def\contentsname{Sadr\v{z}aj}%
51.14 \def\listfigurename{Slike}%
51.15 \def\listtablename{Tabele}%
51.16\def\indexname{Indeks}%
51.17 \def\figurename{Slika}%
51.18\def\tablename{Tabela}%
51.19 \def\partname{Deo}%
51.20 \def\enclname{Prilozi}%
51.21 \def\ccname{Kopije}%
51.22 \def\headtoname{Prima}%
51.23 \def\pagename{Strana}%
51.24\def\seename{Vidi}%
51.25 \def\alsoname{Vidi tako\dj e}%
51.26 \def\proofname{Dokaz}%
51.27 \def\glossaryname{Glossary}% <-- Needs translation
51.28 }%
```

\dateserbian The macro \dateserbian redefines the command \today to produce Serbocroatian dates.

```
51.29\def\dateserbian{%
51.30 \def\today{\number\day . \\ifcase\month\or
51.31 januar\or februar\or mart\or april\or maj\or
51.32 juni\or juli\or avgust\or septembar\or oktobar\or novembar\or
51.33 decembar\fi \space \number\year}}
```

\extrasserbian The macro \extrasserbian will perform all the extra definitions needed for the \noextrasserbian 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.

```
51.34 \initiate@active@char{"}
51.35\addto\extrasserbian{\languageshorthands{serbian}}
51.36 \addto\extrasserbian{\bbl@activate{"}}
```

Don't forget to turn the shorthands off again.

### 51.37 \addto\noextrasserbian\{\bbl@deactivate\{"\}\}

First we define shorthands to facilitate the occurence of letters such as č.

```
51.38\declare@shorthand{serbian}{"c}{\\textormath{\v c}{\check c}}
51.39\declare@shorthand{serbian}{"d}{\textormath{\dj}{\dj}}%%
51.40\declare@shorthand{serbian}{"s}{\textormath{\v s}{\check s}}
51.41 \declare@shorthand{serbian}{"z}{\textormath{\v z}{\check z}}
51.42 \declare@shorthand{serbian}{"C}{\textormath{\v C}{\check C}}
51.43\declare@shorthand{serbian}{"D}{\textormath{\DJ}{\DJ}}%%
51.44 \declare@shorthand{serbian}{"S}{\textormath{\v S}{\check S}}
51.45\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.

```
51.46 \declare@shorthand{serbian}{"‘}{%
51.47 \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
51.48 \declare@shorthand{serbian}{"'}{%
51.49 \textormath{\textquotedblleft{}}{\mbox{\textquotedblleft}}}
51.50\declare@shorthand{serbian}{"<}{%
51.51 \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
51.52\declare@shorthand{serbian}{">}{%
51.53 \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from $\backslash$-.

```
51.54 \declare@shorthand{serbian}{"-}{\nobreak-\bbl@allowhyphens}
```

51.55 \declare@shorthand\{serbian\}\{""\}\{\hskip\z@skip\}

And we want to have a shorthand for disabling a ligature.

```
51.56 \declare@shorthand{serbian}{"|}{%
51.57 \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

\bbl@frenchindent In Serbian the first paragraph of each section should be indented. Add this code $\backslash \mathrm{bbl@nonfrenchindent}$ only in $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$.

```
51.58\ifx\fmtname plain \else
51.59 \let\@aifORI\@afterindentfalse
51.60\\def\bbl@frenchindent{\let\@afterindentfalse\@afterindenttrue
                                    \@afterindenttrue}
    \def\bbl@nonfrenchindent{\let\@afterindentfalse\@aifORI
    \@afterindentfalse}
    \addto\extrasserbian{\bbl@frenchindent}
    \addto\noextrasserbian{\bbl@nonfrenchindent}
51.66 \fi
```

$\backslash$ 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.

```
51.67\def\sh{\mathop{\operator@font sh}\nolimits} % same as \sinh
51.68\def\ch{\mathop{\operator@font ch}\nolimits} % same as \cosh
51.69\def\th{\mathop{\operator@font th}\nolimits} % same as \tanh
51.70\def\cth{\mathop{\operator@font cth}\nolimits} % same as \coth
51.71 \def\arsh{\mathop{\operator@font arsh}\nolimits}
51.72\def\arch{\mathop{\operator@font arch}\nolimits}
51.73\def\arth{\mathop{\operator@font arth}\nolimits}
```

$51.74 \backslash$ def $\backslash \operatorname{arcth}\{\backslash$ mathop $\{$ \operator@font arcth\} $\operatorname{nolimits\} }$
$51.75 \backslash \operatorname{def} \backslash \mathrm{tg}\{\backslash \mathrm{mathop}\{\backslash o p e r a t o r @ f o n t \operatorname{tg}\} \backslash$ nolimits\} $\%$ same as \tan
$51.76 \backslash \operatorname{def} \backslash c t g\{\backslash m a t h o p\{\backslash o p e r a t o r @ f o n t \operatorname{ctg}\} \backslash n o l i m i t s\} \%$ same as $\backslash \cot$
$51.77 \backslash \operatorname{def} \backslash \operatorname{arctg}\{\backslash$ mathop\{\operator@font arctg\}\nolimits\} \% same as \arctan
$51.78 \backslash \operatorname{def} \backslash \operatorname{arcctg}\{\backslash$ mathop\{\operator@font arcctg\}\nolimits\}
51.79 \def $\backslash \operatorname{Prob}\{\backslash$ mathop $\{\backslash$ mathsf $\mathrm{P} \backslash$ hskip0pt $\} \backslash$ nolimits $\}$
$51.80 \backslash \operatorname{def} \backslash$ Expect $\{\backslash$ mathop $\{\backslash$ mathsf E $\backslash$ hskip0pt $\}$ nolimits $\}$
$51.81 \backslash \operatorname{def} \backslash$ Variance $\{\backslash$ mathop\{\mathsf $D \backslash$ hskip0pt $\} \backslash$ nolimits $\}$
The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of @ to its original value.

```
51.82\ldf@finish{serbian}
51.83 </code>
```


## 52 The Slovak language

The file slovak.dtx ${ }^{60}$ defines all the language-specific macros for the Slovak language.

For this language the macro $\backslash \mathrm{q}$ is defined. It was used with the letters ( $\mathrm{t}, \mathrm{d}, \mathrm{l}$, and L) and adds a ' to them to simulate a 'hook' that should be there. The result looks like t. Since the the T1 font encoding has the corresponding characters it is mapped to $\backslash \mathrm{v}$. Therefore we recommend using T1 font encoding. If you don't want to use this encoding, please, feel free to redefine $\backslash q$ in your file. I think babel will honour this ;-).

For this language the characters ", ' and ~ are ade active. In table 24 an overview is given of its purpose. Also the vertical placement of the umlaut can be controlled this way

> "a \"a, also implemented for the other lowercase and uppercase vowels.
> -d \q d, also implemented for l, t and L.
> ${ }^{\text {c }} \mathrm{c}$ \v c, also implemented for $\mathrm{C}, \mathrm{D}, \mathrm{N}, \mathrm{n}, \mathrm{T}, \mathrm{Z}$ and z .
> -o <br>~o, also implemented for O.
> 'a \'a, also implemented for the other lowercase and uppercase l, r, y and vowels.
> " 1 disable ligature at this position.
> " - an explicit hyphen sign, allowing hyphenation in the rest of the word.
> "" like "-, but producing no hyphen sign (for compund 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 $\ll$ ).
> "> for French right double quotes (similar to $\gg$ ).

Table 24: The extra definitions made by slovak.ldf

The quotes in table 24 can also be typeset by using the commands in table 25 .
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\{slovak\}\captionsslovak

[^50]Table 25: More commands which produce quotes, defined by slovak.ldf

When this file is read as an option, i.e. by the command,slovakwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@slovaktoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
52.3\ifx\l@slovak\@undefined
52.4 \@nopatterns{Slovak}
52.5 \adddialect\l@slovak0\fi
```

The next step consists of defining commands to switch to (and from) the Slovak language.
\captionsslovak The macro \captionsslovak defines all strings used in the four standard documentclasses provided with $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
2.6 \addto\captionsslovak{%
    \def\prefacename{\'Uvod}%
52.8\\def\refname{Referencie}%
52.9\def\abstractname{Abstrakt}%
52.10\\def\bibname{Literat\'ura}%
52.11 \def\chaptername{Kapitola}%
52.12 \def\appendixname{Dodatok}%
52.13\def\contentsname{Obsah}%
52.14 \def\listfigurename{Zoznam obr\'azkov}%
52.15 \def\listtablename{Zoznam tabuliek}%
52.16\def\indexname{Index}%
52.17\def\figurename{Obr\'azok}%
52.18\def\tablename{Tabu\q lka}%%% special letter l with hook
52.19 \def\partname{\v{C}as\q t}%%% special letter t with hook
52.20\\def\enclname{Pr\'{\i}lohy}%
52.21 \def\ccname{CC}%
52.22\def\headtoname{Komu}%
52.23\def\pagename{Strana}%
52.24 \def\seename{vi\q d}%%% Special letter d with hook
52.25\def\alsoname{vi\q d tie\v z}%%% Special letter d with hook
52.26\\def\proofname{D\~okaz}%
52.27 \def\glossaryname{Glossary}% <-- Needs translation
```

\dateslovak The macro \dateslovak redefines the command \today to produce Slovak dates.

```
52.29\def\dateslovak{%
52.30\\def\today{\number\day. ~\ifcase\month\or
52.31 janu\'ara\or febru\'ara\or marca\or apr\'{\i}la\or m\'aja\or
52.32 j\'una\or j\'ula\or augusta\or septembra\or okt\'obra\or
52.33 novembra\or decembra\fi
52.34 \space \number\year}}
```

\extrasslovak The macro \extrasslovak will perform all the extra definitions needed for the \noextrasslovak Slovak language. The macro \noextrasslovak is used to cancel the actions of \extrasslovak. For Slovak three characters are used to define shorthands, they need to be made active.

```
52.35\addto\extrasslovak{\languageshorthands{slovak}}
52.36 \initiate@active@char{^}
52.37\addto\extrasslovak{\bbl@activate{^}}
52.38\addto\noextrasslovak{\bbl@deactivate{^}}
52.39 \initiate@active@char{"}
52.40 \addto\extrasslovak{\bbl@activate{"}\umlautlow}
52.41 \addto\noextrasslovak{\bbl@deactivate{"}\umlauthigh}
52.42 \initiate@active@char{'}
52.43\@ifpackagewith{babel}{activeacute}{%
52.44 \addto\extrasslovak{\bbl@activate{'}}
52.45 \addto\noextrasslovak{\bbl@deactivate{'}}%
52.46 }{}
52.47\addto\extrasslovak{\babel@save\q\let\q\v}
```

The slovak hyphenation patterns should be used with \lefthyphenmin set to 2 and \righthyphenmin set to 2.

```
52.48\providehyphenmins{\CurrentOption}{\tw@\tw@}
```

$\backslash \mathrm{dq}$ We save the original double quote character in \dq to keep it available, the math accent \" can now be typed as ".

```
52.49 \begingroup \catcode'\"12
52.50\def\x{\endgroup
52.51 \def\sq{'}
52.52 \def\dq{"}}
52.53\x
```

In order to prevent problems with the active ~ we add a shorthand on system level which expands to a 'normal ~.

```
52.54 \declare@shorthand{system}{^}{\csname normal@char\string`\endcsname}
```

Now we can define the doublequote macros: the umlauts,
$52.55 \backslash$ declare@shorthand\{slovak\}\{"a\}\{\textormath\{\"\{a\}\allowhyphens\}\{\ddot a\}\} 52.56 \declare@shorthand\{slovak\}\{"o\}\{\textormath\{\"\{o\}\allowhyphens\}\{\ddot o\}\}
52.57 \declare@shorthand\{slovak\}\{"u\}\{\textormath\{\"\{u\}\allowhyphens\}\{\ddot u\}\} 52.58 \declare@shorthand\{slovak\}\{"A\}\{\textormath\{\"\{A\}\allowhyphens\}\{\ddot A\}\} 52.59 \declare@shorthand\{slovak\}\{"0\}\{\textormath\{\"\{0\}\allowhyphens\}\{\ddot 0\}\} $52.60 \backslash$ declare@shorthand\{slovak\}\{"U\}\{\textormath\{\"\{U\}\allowhyphens\}\{\ddot $U\}\}$
tremas,

other slovak characters
$52.66 \backslash$ declare@shorthand\{slovak\}\{^c\}\{\textormath\{\v\{c\} ${ }^{\text {allowhyphens }\}\{\backslash c h e c k\{c\}\}\}}$
$52.67 \backslash$ declare@shorthand\{slovak\}\{^d\}\{\textormath $\{\backslash q\{d\} \backslash$ allowhyphens $\}\{\backslash$ check $\{d\}\}\}$
52.68 \declare@shorthand\{slovak\}\{^l\}\{\textormath\{\q\{l\}\allowhyphens\}\{\check\{l\}\}\}
52.69 \declare@shorthand\{slovak\}\{^n\}\{\textormath\{\v\{n\}\allowhyphens\}\{\check\{n\}\}\}
$52.70 \backslash$ declare@shorthand\{slovak\}\{~o\}\{\textormath\{\~\{o\}\allowhyphens\}\{\hat\{o\}\}\}
52.71 \declare@shorthand\{slovak\}\{^s\}\{\textormath\{\v\{s\}\allowhyphens\}\{\check\{s\}\}\}
52.72 \declare@shorthand\{slovak\}\{^t\}\{\textormath\{\q\{t\}\allowhyphens\}\{\check\{t\}\}\}
$52.73 \backslash$ declare@shorthand\{slovak\}\{^z\}\{\textormath $\{\backslash v\{z\} \backslash$ allowhyphens $\}\{\backslash$ check $\{z\}\}\}$
52.74 \declare@shorthand\{slovak\}\{~C\}\{\textormath\{\v\{C\}\allowhyphens\}\{\check\{C\}\}\}
52.75 \declare@shorthand\{slovak\}\{^D\}\{\textormath\{\v\{D\}\allowhyphens\}\{\check\{D\}\}\}
52.76 \declare@shorthand\{slovak\}\{ $\sim$ L\}\{\textormath\{\q\{L\}\allowhyphens\}\{\check\{L\}\}\}
52.77 \declare@shorthand\{slovak\}\{~N\}\{\textormath\{\v\{N\}\allowhyphens\}\{\check\{N\}\}\}
52.78 \declare@shorthand\{slovak\}\{~0\}\{\textormath\{\~\{0\}\allowhyphens\}\{\hat\{0\}\}\}
52.79 \declare@shorthand\{slovak\}\{ ${ }^{\wedge}$ S\}\{\textormath\{\v\{S\}\allowhyphens\}\{\check\{S\}\}\}
$52.80 \backslash$ declare@shorthand\{slovak\}\{^T\}\{\textormath\{\v\{T\}\allowhyphens\}\{\check\{T\}\}\}
52.81 \declare@shorthand\{slovak\}\{^Z\}\{\textormath\{\v\{Z\}\allowhyphens\}\{\check\{Z\}\}\}
acute accents,
52.82 \@ifpackagewith\{babel\}\{activeacute\}\{\%
\declare@shorthand\{slovak\}\{'a\}\{\textormath\{\'a\allowhyphens\}\{^\{\prime\}a\}\} \declare@shorthand\{slovak\}\{'e\}\{\textormath $\{\backslash$ 'e $\backslash$ allowhyphens\}\{~\{\prime\}e\}\} $\backslash$ declare@shorthand\{slovak\}\{'i\}\{\textormath $\{\backslash ’ \backslash i\} \backslash a l l o w h y p h e n s\}\{\wedge\{\backslash$ prime $\} i\}\}$ \declare@shorthand\{slovak\}\{'l\}\{\textormath\{ $\backslash$ ' $1 \backslash$ allowhyphens $\}\{\wedge\{\backslash$ prime $\} 1\}\}$ \declare@shorthand\{slovak\}\{'०\}\{\textormath $\left\{\backslash\right.$ ' ${ }^{\prime}$ \allowhyphens\} $\{\wedge\{\backslash$ prime\}o\}\} \declare@shorthand\{slovak\}\{'r\}\{\textormath\{\'r\allowhyphens\}\{^\{\prime\}r\}\} \declare@shorthand\{slovak\}\{'u\}\{\textormath\{\'u\allowhyphens\}\{^\{\prime\}u\}\} \declare@shorthand\{slovak\}\{'y\}\{\textormath $\{\backslash$ 'y $\backslash$ allowhyphens $\}\{\wedge\{\backslash$ prime $\}$ y\}\} \declare@shorthand\{slovak\}\{'A\}\{\textormath\{\'A\allowhyphens\}\{^\{\prime\}A\}\} \declare@shorthand\{slovak\}\{'E\}\{\textormath\{\'E\allowhyphens\}\{^\{\prime\}E\}\} \declare@shorthand\{slovak\}\{'I\}\{\textormath $\backslash \backslash$ 'I $\backslash$ allowhyphens $\}\{\wedge\{\backslash$ prime $\}$ I $\}\}$ \declare@shorthand\{slovak\}\{'L\}\{\textormath\{\’L\allowhyphens\}\{^\{\prime\}l\}\} \declare@shorthand\{slovak\}\{'0\}\{\textormath\{\'O\allowhyphens\}\{^\{\prime\}0\}\} \declare@shorthand\{slovak\}\{'R\}\{\textormath $\{\backslash$ ' $\backslash \backslash$ allowhyphens $\}\{\wedge\{\backslash$ prime $\}$ R\}\} \declare@shorthand\{slovak\}\{'U\}\{\textormath $\{\backslash$ 'U\allowhyphens\}\{^\{\prime\}U\}\} \declare@shorthand\{slovak\}\{'Y\}\{\textormath $\ \backslash$ 'Y\allowhyphens\}\{~\{\prime\}Y\}\} \declare@shorthand\{slovak\}\{', \}\{\%
\textormath\{”\}\{\sp\bgroup\prim@s'\}\}
\}\{\}

```
    german and french quotes,
52.102 \declare@shorthand{slovak}{"'}{\glqq}
52.103 \declare@shorthand{slovak}{"'}{\grqq}
52.104 \declare@shorthand{slovak}{"<}{\flqq}
52.105\declare@shorthand{slovak}{">}{\frqq}
    and some additional commands:
52.106 \declare@shorthand{slovak}{"-}{\nobreak\-\bbl@allowhyphens}
52.107 \declare@shorthand{slovak}{"|}{%
52.108 \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
52.109 \bbl@allowhyphens}{}}
52.110 \declare@shorthand{slovak}{""}{\hskip\z@skip}
52.111 \declare@shorthand{slovak}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
52.112 \declare@shorthand{slovak}{"=}{\nobreak-\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.

```
52.113\ldf@finish{slovak}
52.114 </code\
```


## 53 The Slovenian language

The file slovene. $\mathrm{dtx}^{61}$ defines all the language-specific macros for the Slovenian language.

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 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 compund 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 $\ll$ ).
" $>$ for French right double quotes (similar to $\gg$ ).
Table 26: 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.
53.1 〈*code〉
53.2 \LdfInit\{slovene\}\captionsslovene

When this file is read as an option, i.e. by the command,slovenewillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@slovenetoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
53.3\ifx\l@slovene\@undefined
53.4 \@nopatterns{Slovene}
53.5 \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 $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$.

```
53.6 \addto\captionsslovene{%
53.7 \def\prefacename{Predgovor}%
53.8 \def\refname{Literatura}%
53.9\def\abstractname{Povzetek}%
```

${ }^{61}$ The file described in this section has version number v 1.2 m 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).

```
53.10\\def\bibname{Literatura}%
53.11 \def\chaptername{Poglavje}%
53.12 \def\appendixname{Dodatek}%
53.13\def\contentsname{Kazalo}%
53.14 \def\listfigurename{Slike}%
53.15\def\listtablename{Tabele}%
53.16 \def\indexname{Stvarno kazalo}% used to be Indeks
53.17 \def\figurename{Slika}%
53.18\\def\tablename{Tabela}%
53.19\def\partname{Del}%
53.20\\def\enclname{Priloge}%
53.21 \def\ccname{Kopije}%
53.22 \def\headtoname{Prejme}%
53.23\def\pagename{Stran}%
53.24\def\seename{glej}%
53.25\def\alsoname{glej tudi}%
53.26 \def\proofname{Dokaz}%
53.27 \def\glossaryname{Glossary}% <-- Needs translation
53.28 }%
```

\dateslovene The macro \dateslovene redefines the command \today to produce Slovenian dates.

```
53.29\def\dateslovene{%
53.30\\def\today{\number\day. ~\ifcase\month\or
53.31 januar\or februar\or marec\or april\or maj\or junij\or
53.32 julij\or avgust\or september\or oktober\or november\or december\fi
53.33 \space \number\year}}
```

\extrasslovene The macro \extrasslovene performs all the extra definitions needed for the Slove\noextrasslovene nian 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.

```
53.34 \initiate@active@char{"}
53.35\addto\extrasslovene{\languageshorthands{slovene}}
53.36 \addto\extrasslovene{\bbl@activate{"}}
```

Don't forget to turn the shorthands off again.

```
53.37\addto\noextrasslovene{\bbl@deactivate{"}}
```

First we define shorthands to facilitate the occurence of letters such as č.

```
53.38 \declare@shorthand{slovene}{"c}{\\textormath{\v c}{\check c}}
```

53.39 \declare@shorthand\{slovene\}\{"s\}\{\textormath\{\v s\}\{\check s\}\}
53.40 \declare@shorthand\{slovene\}\{"z\}\{\textormath\{\v z\}\{\check z\}\}
53.41 \declare@shorthand\{slovene\}\{"C\}\{\textormath\{\v C\}\{\check C\}\}
53.42 \declare@shorthand\{slovene\}\{"S\}\{\textormath\{\v S\}\{\check S\}\}
53.43 \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.

```
53.44 \declare@shorthand{slovene}{"`}{%
53.45\textormath{\quotedblbase}{\mbox{\quotedblbase}}}
53.46 \declare@shorthand{slovene}{"'}{%
53.47 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
53.48 \declare@shorthand{slovene}{"<}{%
53.49 \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
53.50\declare@shorthand{slovene}{">}{%
53.51 \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behavew a little different from \-.

```
53.52\declare@shorthand{slovene}{"-}{\nobreak-\bbl@allowhyphens}
```

53.53 \declare@shorthand\{slovene\}\{""\}\{\hskip\z@skip\}

And we want to have a shorthand for disabling a ligature.
53.54 \declare@shorthand\{slovene\}\{"|\}\{\%
53.55 \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.

```
53.56\ldf@finish{slovene}
53.57 </code\rangle
```


## 54 The Russian language

The file russianb. $\mathrm{dtx}^{62}$ defines all the language-specific macros for the Russian language. It needs the file cyrcod for success documentation with Russian encodings (see below).

For this language the character " is made active. In table 27 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 compund
    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, allow-
        ing 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 27: The extra definitions made by russianb

The quotes in table 27 can also be typeset by using the commands in table 28.

```
\cdash--- Cyrillic emdash in plain text.
\cdash--~ Cyrillic emdash in compound names (surnames).
\cdash--* Cyrillic emdash for denoting direct speech.
\glqq for German left double quotes (looks like „).
\grqq for German right double quotes (looks like ").
"\flqq for French left double quotes (looks like <<).
\frqq for French right double quotes (looks like >>).
\dq the original quotes character (").
```

Table 28: More commands which produce quotes, defined by babel

[^51]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.

```
54.1 〈*code〉
54.2\LdfInit{russian}{captionsrussian}
```

When this file is read as an option, i.e., by the command,russianbwillbean'unknown'language,inwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@russiantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
54.3\ifx\l@russian\@undefined
54.4 \@nopatterns{Russian}
54.5 \adddialect\l@russian0
54.6 \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!

```
54.7\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,0T1] {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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ 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.

```
54.8 \def\reserved@a#1#2{%
54.9\\edef\reserved@b{#1}%
54.10 \edef\reserved@c{#2}%
54.11 \ifx\reserved@b\reserved@c
54.12 \let\cyrillicencoding\reserved@c
54.13 \fi}
```

```
54.14 \def\cdp@elt#1#2#3#4{%
54.15 \reserved@a{#1}{0T2}%
54.16 \reserved@a{#1}{LWN}%
54.17 \reserved@a{#1}{LCY}%
54.18 \reserved@a{#1}{X2}%
54.19 \reserved@a{#1}{T2C}%
54.20 \reserved@a{#1}{T2B}%
54.21 \reserved@a{#1}{T2A}}
54.22 \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).

```
54.23\ifx\cyrillicencoding\undefined
54.24\IfFileExists{ot2enc.def}{\def\cyrillicencoding{OT2}}\relax
54.25\IfFileExists{lwnenc.def}{\def\cyrillicencoding{LWN}}\relax
54.26 \IfFileExists{lcyenc.def}{\def\cyrillicencoding{LCY}}\relax
54.27\IfFileExists{x2enc.def}{\def\cyrillicencoding{X2}}\relax
54.28\IfFileExists{t2cenc.def}{\def\cyrillicencoding{T2C}}\relax
54.29 \IfFileExists{t2benc.def}{\def\cyrillicencoding{T2B}}\relax
54.30\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.

```
54.31 \ifx\cyrillicencoding\undefined
54.32 \PackageError{babel}%
54.33 {No Cyrillic encoding definition files were found}%
54.34 {Your installation is incomplete.\MessageBreak
54.35 You need at least one of the following files:\MessageBreak
54.36 \space\space
    x2enc.def, t2aenc.def, t2benc.def, t2cenc.def,\MessageBreak
        \space\space
        lcyenc.def, lwnenc.def, ot2enc.def.}%
    \else
```

We avoid sepackage[\cyrillicencoding]\{fontenc\}becausewedon'twanttoforcetheswitchof\encodingdefault.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
54.41 \lowercase
54.42 \expandafter{\expandafter\input\cyrillicencoding enc.def\relax}%
54.43 \fi
54.44\fi
    \PackageInfo{babel}
    {Using '\cyrillicencoding' as a default Cyrillic encoding}%
```

54.45 \DeclareRobustCommand\{\Russian\}\{\%
54.46 \fontencoding\cyrillicencoding\selectfont

```
54.47 \let\encodingdefault\cyrillicencoding
54.48 \expandafter\set@hyphenmins\russianhyphenmins
54.49 \language\l@russian}%
54.50 \DeclareRobustCommand{\English}{%
54.51 \fontencoding\latinencoding\selectfont
54.52 \let\encodingdefault\latinencoding
54.53 \expandafter\set@hyphenmins\englishhyphenmins
54.54 \language\l@english}%
54.55\let\Rus\Russian
54.56\let\Eng\English
54.57\let\cyrillictext\Russian
54.58\let\cyr\Russian
```

Since the X2 encoding does not contain Latin letters, we should make some redefinitions of $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ macros which implicitly produce Latin letters.

```
54.59\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 ' $\backslash$ fontencoding OT1' ( $\backslash$ fontencoding is robust).

```
54.60\\def\@alph#1{{\fontencoding{\latinencoding}\selectfont
54.61 \ifcase#1\or
54.62 a\or b\or c\or d\or e\or f\or g\or h\or
54.63 i\or j\or k\or l\or m\or n\or o\or p\or
54.64 q\or r\or s\or t\or u\or v\or w\or x\or
        y\or z\else\@ctrerr\fi}}%
\def\@Alph#1{{\fontencoding{\latinencoding}\selectfont
    \ifcase#1\or
        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 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\@ctrerr\fi}}%
```

Unfortunately, the commands $\backslash \mathrm{AA}$ and $\backslash$ aa are not encoding dependent in $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ (unlike e.g., \oe or $\backslash \mathrm{DH}$ ). They are defined as $\backslash \mathrm{r}\{\mathrm{A}\}$ and $\backslash \mathrm{r}\{\mathrm{a}\}$. This leads to unpredictable results when the font encoding does not contain the Latin letters ' $A$ ' and ' $a$ ' (like X2).

```
54.72 \DeclareTextSymbolDefault{\AA}{OT1}
54.73 \DeclareTextSymbolDefault{\aa}{OT1}
54.74 \DeclareTextCommand{\aa}{OT1}{\r a}
54.75 \DeclareTextCommand{\AA}{OT1}{\r A}
54.76 \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.

```
54.77 % \begingroup\catcode`\"=12
```

```
54.78% % uppercase greek letters:
54.79 % \def\@tempa#1{\expandafter\@tempb\meaning#1\relax\relax\relax\relax
54.80 % "0000\@nil#1}
54.81% \def\@tempb#1"#2#3#4#5#6\@nil#7{%
54.82% \ifnum"#2=7 \count@"1#3#4#5\relax
54.83% \ifnum\count@<"1000 \else \global\mathchardef#7="0#3#4#5\relax \fi
54.84% \fi}
54.85% \@tempa\Gamma\@tempa\Delta\@tempa\Theta\@tempa\Lambda\@tempa\Xi
54.86% \@tempa\Pi\@tempa\Sigma\@tempa\Upsilon\@tempa\Phi\@tempa\Psi
54.87% \@tempa\Omega
54.88% % some accents:
54.89% \def\@tempa#1#2\@nil{\def\@tempc{#1}}\def\@tempb{\mathaccent}
54.90% \expandafter\@tempa\hat\relax\relax\@nil
54.91 % \ifx\@tempb\@tempc
54.92% \def\@tempa#1\@nil{#1}%
54.93% \def\@tempb#1{\afterassignment\@tempa\mathchardef\@tempc=}%
54.94% \def\do#1"#2{}
54.95% \def\@tempd#1{\expandafter\@tempb#1\@nil
54.96% \ifnum\@tempc>"FFF
54.97% \xdef#1{\mathaccent"\expandafter\do\meaning\@tempc\space}%
54.98% \fi}
54.99% \@tempd\hat\@tempd\check\@tempd\tilde\@tempd\acute\@tempd\grave
54.100% \@tempd\dot\@tempd\ddot\@tempd\breve\@tempd\bar
54.101 % \fi
54.102% \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 backslash\)AtBeginDocument,soinputencshouldbeusedbeforebabel.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
54.103\@ifpackageloaded{inputenc}{}{%
54.104\def\reserved@a{LWN}%
54.105 \ifx\reserved@a\cyrillicencoding\else
54.106 \def\reserved@a{0T2}%
54.107 \ifx\reserved@a\cyrillicencoding\else
54.108 \PackageWarning{babel}%
    {No input encoding specified for Russian language}
    \i\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 \latintext 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.

```
54.112% \fontencoding{\latinencoding}\selectfont
54.113 % \def\encodingdefault{\latinencoding}}
54.114 \let\lat\latintext
```

\textcyrillic These commands take an argument which is then typeset using the requested font \textlatin encoding.

```
54.115\DeclareTextFontCommand{\textcyrillic}{\cyrillictext}
54.116 %\DeclareTextFontCommand{\textlatin}{\latintext}
```

We make the $\mathrm{T}_{\mathrm{E}} \mathrm{X}$
$54.117 \%$ ifx $\backslash$ ltxTeX $\backslash$ undefined $\backslash l e t \backslash l t x T e X \backslash T e X \backslash f i$ 54.118 \% $\backslash$ ProvideTextCommandDefault $\{\backslash \mathrm{TeX}\}\{\backslash$ textlatin\{ $\backslash$ ltxTeX\}\}
and $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ logos encoding independent. 54.119 \% \ifx $\backslash$ ltxLaTeX $\backslash u n d e f i n e d \backslash l e t \backslash l t x L a T e X \backslash L a T e X \backslash f i$ 54.120 \% \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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$. The two commands $\backslash \mathrm{cyr}$ and $\backslash$ lat activate Cyrillic resp. Latin encoding.

```
54.121 \addto\captionsrussian{%
54.122% FIXME: Where is the \prefacename used?
54.123\\def\prefacename{%
54.124 {\cyr\CYRP\cyrr\cyre\cyrd\cyri\cyrs\cyrl\cyro\cyrv\cyri\cyre}}%
54.125% {\cyr\CYRV\cyrv\cyre\cyrd\cyre\cyrn\cyri\cyre}}%
54.126 \def\refname{%
54.127 {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
54.128 \ \cyrl\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyrery}}%
54.129% \def\refname{%
54.130% {\cyr\CYRL\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
54.131 \def\abstractname{%
54.132 {\cyr\CYRA\cyrn\cyrn\cyro\cyrt\cyra\cyrc\cyri\cyrya}}%
54.133 \def\bibname{%
54.134 {\cyr\CYRL\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
54.135% \def\bibname{%
54.136% {\cyr\CYRB\cyri\cyrb\cyrl\cyri\cyro
54.137% \cyrg\cyrr\cyra\cyrf\cyri\cyrya}}%
54.138% for reports according to GOST:
54.139 % \def\bibname{%
54.140 % {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
54.141% \ \cyri\cyrs\cyrp\cyro\cyrl\cyrsftsn\cyrz\cyro\cyrv\cyra\cyrn
54.142% \cyrn\cyrery\cyrh\ \cyri\cyrs\cyrt\cyro\cyrch\cyrn\cyri
54.143 % \cyrk\cyro\cyrv}}%
54.144 \def\chaptername{{\cyr\CYRG\cyrl\cyra\cyrv\cyra}}%
54.145% \@ifundefined{chapter}{}{%
54.146 % \def\chaptername{{\cyr\CYRG\cyrl\cyra\cyrv\cyra}}}%
54.147 \def\appendixname{%
54.148 {\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:

```
54.149 \@ifundefined{thechapter}%
    {\def\contentsname{%
        {\cyr\CYRS\cyro\cyrd\cyre\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}}%
    {\def\contentsname{%
        {\cyr\CYRO\cyrg\cyrl\cyra\cyrv\cyrl\cyre\cyrn\cyri\cyre}}}%
    \def\listfigurename{%
    {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
        \ \cyri\cyrl\cyrl\cyryu\cyrs\cyrt\cyrr\cyra\cyrc\cyri\cyrishrt}}%
\def\listfigurename{%
    {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
        \ \cyrr\cyri\cyrs\cyru\cyrn\cyrk\cyro\cyrv}}%
    \def\listtablename{%
        {\cyr\CYRS\cyrp\cyri\cyrs\cyro\cyrk
            \ \cyrt\cyra\cyrb\cyrl\cyri\cyrc}}%
    \def\indexname{%
        {\cyr\CYRP\cyrr\cyre\cyrd\cyrm\cyre\cyrt\cyrn\cyrery\cyrishrt
            \ \cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl\cyrsftsn}}%
    \def\authorname{%
        {\cyr\CYRI\cyrm\cyre\cyrn\cyrn\cyro\cyrishrt
            \ \cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl\cyrsftsn}}%
    \def\figurename{{\cyr\CYRR\cyri\cyrs.}}%
    \def\tablename{{\cyr\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyra}}%
    \def\partname{{\cyr\CYRCH\cyra\cyrs\cyrt\cyrsftsn}}%
    \def\enclname{{\cyr\cyrv\cyrk\cyrl.}}%
    \def\ccname{{\cyr\cyri\cyrs\cyrh.}}%
    \def\ccname{{\cyr\cyri\cyrz}}%
    \def\headtoname{{\cyr\cyrv\cyrh.}}%
54.176% \def\headtoname{{\cyr\cyrv}}%
54.177 \def\pagename{{\cyr\cyrs.}}%
54.178% \def\pagename{{\cyr\cyrs\cyrt\cyrr.}}%
54.179 \def\seename{{\cyr\cyrs\cyrm.}}%
54.180\\def\alsoname{{\cyr\cyrs\cyrm.\ \cyrt\cyra\cyrk\cyrzh\cyre}}%
54.181 \def\proofname{{\cyr\CYRD\cyro\cyrk\cyra\cyrz\cyra\cyrt
    \cyre\cyrl\cyrsftsn\cyrs\cyrt\cyrv\cyro}}%
    \def\glossaryname{Glossary}% <-- Needs translation
    }
```

\daterussian The macro \daterussian redefines the command \today to produce Russian dates.

```
54.185\def\daterussian{%
54.186 \def\today{\number\day~\ifcase\month\or
54.187 \cyrya\cyrn\cyrv\cyra\cyrr\cyrya\or
54.188 \cyrf\cyre\cyrv\cyrr\cyra\cyrl\cyrya\or
54.189 \cyrm\cyra\cyrr\cyrt\cyra\or
54.190 \cyra\cyrp\cyrr\cyre\cyrl\cyrya\or
```

```
54.191 \cyrm\cyra\cyrya\or
54.192 \cyri\cyryu\cyrn\cyrya\or
54.193 \cyri\cyryu\cyrl\cyrya\or
54.194 \cyra\cyrv\cyrg\cyru\cyrs\cyrt\cyra\or
54.195 \cyrs\cyre\cyrn\cyrt\cyrya\cyrb\cyrr\cyrya\or
54.196 \cyro\cyrk\cyrt\cyrya\cyrb\cyrr\cyrya\or
54.197 \cyrn\cyro\cyrya\cyrb\cyrr\cyrya\or
54.198 \cyrd\cyre\cyrk\cyra\cyrb\cyrr\cyrya\fi
54.199 \ \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'.

```
54.200 \addto\extrasrussian{\cyrillictext}
```

When the encoding definition file was processed by $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ the current font encoding is stored in \latinencoding, assuming that $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ uses T 1 or OT1 as default. Therefore we switch back to \latinencoding whenever the Russian language is no longer 'active'.

```
54.201 \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 $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ command somewhat:

```
54.202 %\def\verbatim@font{%
54.203% \let\encodingdefault\latinencoding
54.204% \normalfont\ttfamily
54.205% \expandafter\def\csname\cyrillicencoding-cmd\endcsname##1##2{%
54.206 % \ifx\protect\@typeset@protect
54.207% \begingroup\UseTextSymbol\cyrillicencoding##1\endgroup
54.208% \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 $\backslash$ AutoSpaceBeforeFDP switches back to the default behavior of russianb.

```
54.209 <*frenchpunct\rangle
54.210 \initiate@active@char{:}
54.211 \initiate@active@char{;}
54.212 〈/frenchpunct\rangle
54.213 〈*frenchpunct | spanishligs\rangle
54.214 \initiate@active@char{!}
54.215 \initiate@active@char{?}
54.216 </frenchpunct | spanishligs\rangle
54.217 \initiate@active@char{"}
```

The code above is necessary because we need extra active characters．The character＂is used as indicated in table 27.

We specify that the Russian group of shorthands should be used．

```
54.218\addto\extrasrussian{\languageshorthands{russian}}
```

These characters are＇turned on＇once，later their definition may vary．

```
54.219 \addto\extrasrussian{%
54.220 <frenchpunct\rangle \bbl@activate{:}\bbl@activate{;}%
54.221 <frenchpunct|spanishligs\rangle \bbl@activate{!}\bbl@activate{?}%
54.222 \bbl@activate{"}}
54.223 \addto\noextrasrussian{%
54.224 <frenchpunct\rangle \bbl@deactivate{:}\bbl@deactivate{;}%
54.225 <frenchpunct|spanishligs\rangle \bbl@deactivate{!}\bbl@deactivate{?}%
54.226 \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．

```
54.227〈*spanishligs\rangle
54.228 \declare@shorthand{russian}{?`}{\UseTextSymbol{OT1}\textquestiondown}
54.229 \declare@shorthand{russian}{!`}{\UseTextSymbol{OT1}\textexclamdown}
54.230 </spanishligs〉
```

\russian＠sh＠；＠We have to reduce the amount of white space before ；，：and ！．This should only \russian＠sh＠：＠happen in horizontal mode，hence the test with \ifhmode．

```
\russian@sh@!@4.231 <*frenchpunct>
\russian@sh@?@4.232 \declare@shorthand{russian}{;}{%
54.233 \ifhmode
```

In horizontal mode we check for the presence of a＇space＇，＇unskip＇if it exists and place a 0.1 em kerning．
\unskip\nobreak\kern.1em
54.236
\else

If no space has been typed, we add $\backslash$ FDP@thinspace which will be defined, up to the user's wishes, as an automatic added thinspace, or as \@empty.

```
54.237 \FDP@thinspace
54.238 \fi
54.239 \fi
```

Now we can insert a ';' character.

```
54.240 \string;}
```

The other definitions are very similar.

```
54.241 \declare@shorthand{russian}{:}{%
54.242 \ifhmode
54.243 \ifdim\lastskip>\z@
54.244 \unskip\nobreak\kern.1em
54.245 \else
54.246 \FDP@thinspace
54.247 \fi
54.248 \fi
54.249 \string:}
54.250\declare@shorthand{russian}{!}{%
54.251 \ifhmode
54.252 \ifdim\lastskip>\z@
            \unskip\nobreak\kern.1em
        \else
            \FDP@thinspace
        \fi
        \fi
54.258 \string!}
54.259\declare@shorthand{russian}{?}{%
54.260 \ifhmode
54.261 \ifdim\lastskip>\z@
54.262 \unskip\nobreak\kern.1em
54.263 \else
54.264 \FDP@thinspace
54.265 \fi
54.266 \fi
54.267 \string?}
```

$\backslash$ AutoSpaceBeforeFDP \FDP@thinspace is defined as unbreakable spaces if $\backslash$ AutoSpaceBeforeFDP is $\backslash$ NoAutoSpaceBeforeFDP activated or as \@empty if \NoAutoSpaceBeforeFDP is in use. The default is $\backslash$ FDP@thinspace \AutoSpaceBeforeFDP.

```
54.268 \def \AutoSpaceBeforeFDP{%
54.269 \def\FDP@thinspace{\nobreak\kern.1em}}
54.270\def\NoAutoSpaceBeforeFDP{\let\FDP@thinspace\@empty}
54.271 \AutoSpaceBeforeFDP
```

$\backslash$ FDPon The next macros allow to switch on/off activeness of double punctuation signs. $\backslash$ FDPoff

```
54.272\def\FDPon{\bbl@activate{:}%
54.273 \bbl@activate{;}%
54.274 \bbl@activate{?}%
54.275 \bbl@activate{!}}
54.276\def\FDPoff{\bbl@deactivate{:}%
54.277 \bbl@deactivate{;}%
54.278 \bbl@deactivate{?}%
54.279 \bbl@deactivate{!}}
```

\system@sh@:@ When the active characters appear in an environment where their Russian be\system@sh@!@ haviour is not wanted they should give an 'expected' result. Therefore we define \system@sh@?@ shorthands at system level as well.

```
\system@sh@;@4.280\declare@shorthand{system}{:}{\string:}
54.281 \declare@shorthand{system}{;}{\string;}
54.282 </frenchpunct\rangle
54.283 <*frenchpunct&!spanishligs\rangle
54.284 \declare@shorthand{system}{!}{\string!}
54.285 \declare@shorthand{system}{?}{\string?}
54.286 〈/frenchpunct&!spanishligs〉
```

To be able to define the function of '"', we first define a couple of 'support' macros.
$\backslash d q$ We save the original double quote character in $\backslash d q$ to keep it available, the math accent \" can now be typed as '"'.

```
54.287 \begingroup \catcode'\"12
54.288\def\reserved@a{\endgroup
54.289 \def\@SS{\mathchar"7019 }
54.290 \def\dq{"}}
54.291 \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.

```
54.292 \declare@shorthand{russian}{"‘}{\glqq}
54.293 \declare@shorthand{russian}{"'}{\grqq}
54.294 \declare@shorthand{russian}{"<}{\flqq}
54.295 \declare@shorthand{russian}{">}{\frqq}
Some additional commands:
54.296 \declare@shorthand{russian}{""}{\hskip\z@skip}
54.297\declare@shorthand{russian}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
54.298 \declare@shorthand{russian}{"=}{\nobreak-\hskip\z@skip}
54.299 \declare@shorthand{russian}{"|}{%
54.300 \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
54.301
    \allowhyphens}{}}
```

The next two macros for "- and "--- are somewhat different. We must check whether the second token is a hyphen character:

[^52]If the next token is '-', we typeset an emdash, otherwise a hyphen sign:

```
54.303 \def\russian@sh@tmp{%
54.304 \if\russian@sh@next-\expandafter\russian@sh@emdash
54.305 \else\expandafter\russian@sh@hyphen\fi
54.306 }%
```

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.

```
54.307 \futurelet\russian@sh@next\russian@sh@tmp}
```

Here are the definitions of hyphen and emdash. First the hyphen:

```
54.308\def\russian@sh@hyphen{%
```

54.309 \nobreak\-\bbl@allowhyphens\}

For the emdash definition, there are the two parameters: we must 'eat' two last hyphen signs of our emdash...:
$54.310 \backslash$ def \russian@sh@emdash\#1\#2\{\cdash-\#1\#2\}
\cdash ... these two parameters are useful for another macro: \cdash:

```
54.311%\ifx\cdash\undefined % should be defined earlier
54.312 \def\cdash#1#2#3{\def\tempx@{#3}%
54.313\def\tempa@{-}\def\tempb@{~}}\def\tempc@{*}
5 4 . 3 1 4 \ i f x \ t e m p x @ \ t e m p a @ \ @ A c d a s h \ e l s e
54.315 \ifx\tempx@\tempb@\@Bcdash\else
54.316 \ifx\tempx@\tempc@\@Ccdash\else
54.317 \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 $\mathrm{T}_{E} \mathrm{X}$ uses horizontal mode. Maybe there is a misunderstanding? Maybe there is another way?) After a dash a breakable thinspace is always placed;

```
54.318% What is more grammatically: . 2em or . 2\fontdimen6\font ?
54.319\def\@Acdash{\ifdim\lastskip>\z@\unskip\nobreak\hskip.2em\fi
54.320 \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
$54.321 \backslash d e f \backslash @ B c d a s h\{\backslash l e a v e v m o d e \backslash i f d i m \backslash l a s t s k i p>\backslash z @ \backslash u n s k i p \backslash f i$
54.322 \nobreak\cyrdash \penalty ${ }^{2}$ exhyphenpenalty $\backslash$ hskip $\left.\backslash z @ s k i p \backslash i g n o r e s p a c e s\right\} \%$
"--* for denoting direct speech (a space like \enskip must follow the emdash);

```
54.323\def\@Ccdash{\leavevmode
54.324 \nobreak\cyrdash\nobreak\hskip.35em\ignorespaces}%
54.325 %\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 ---.

```
54.326 % Is there an IF necessary?
54.327\ifx\cyrdash\undefined
54.328\\def\cyrdash{\hbox to.8em{--\hss--}}
54.329 \fi
```

Here a really new macro - to place thinspace between initials. This macro used instead of $\backslash$, allows hyphenation in the following surname.
$54.330 \backslash$ declare@shorthand\{russian\}\{",\}\{\nobreak\hskip.2em\ignorespaces\}
$\backslash$ mdqon All that's left to do now is to define a couple of commands for ".
\mdqof § $_{4.331 \text { \def } \backslash \text { mdqon\{ } \backslash \text { bbl@activate\{" }\}\}}$
54.332 \def $\backslash$ mdqoff $\{\backslash$ bbl@deactivate\{"\}\}

The Russian hyphenation patterns can be used with \lefthyphenmin and \righthyphenmin set to 2 .
$54.333 \backslash$ providehyphenmins $\{\backslash$ CurrentOption $\}\{$ tw@ $\backslash$ tw@ $\}$ 54.334 \% temporary hack: 54.335 \ifx\englishhyphenmins \undefined $54.336 \backslash$ def $\backslash e n g l i s h h y p h e n m i n s\{\backslash t w @ \backslash t h r @ @\}$ 54.337 \fi

Now the action \extrasrussian has to execute is to make sure that the command \frenchspacing is in effect. If this is not the case the execution of \noextrasrussian will switch it off again.
54.338 \addto \extrasrussian\{ $\backslash$ bbl@frenchspacing\}
54.339 \addto \noextrasrussian\{ \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.
$\backslash$ Asbuk We begin by defining \Asbuk which works like $\backslash$ Alph, but produces (uppercase) Cyrillic letters intead of Latin ones. The letters YO, ISHRT, HRDSN, ERY, and SFTSN are skipped, as usual for such enumeration.

```
54.340\def\Asbuk#1{\expandafter\@Asbuk\csname c@#1\endcsname}
54.341 \def\@Asbuk#1{\ifcase#1\or
54.342 \CYRA\or\CYRB\or\CYRV\or\CYRG\or\CYRD\or\CYRE\or\CYRZH\or
54.343\CYRZ\or\CYRI\or\CYRK\or\CYRL\or\CYRM\or\CYRN\or\CYRO\or
54.344 \CYRP\or\CYRR\or\CYRS\or\CYRT\or\CYRU\or\CYRF\or\CYRH\or
54.345 \CYRC\or\CYRCH\or\CYRSH\or\CYRSHCH\or\CYREREV\or\CYRYU\or
54.346 \CYRYA\else\@ctrerr\fi}
```

\asbuk The macro \asbuk is similar to \alph; it produces lowercase Russian letters.

```
54.347\def\asbuk#1{\expandafter\@asbuk\csname c@#1\endcsname}
54.348\def\@asbuk#1{\ifcase#1\or
54.349 \cyra\or\cyrb\or\cyrv\or\cyrg\or\cyrd\or\cyre\or\cyrzh\or
54.350 \cyrz\or\cyri\or\cyrk\or\cyrl\or\cyrm\or\cyrn\or\cyro\or
54.351 \cyrp\or\cyrr\or\cyrs\or\cyrt\or\cyru\or\cyrf\or\cyrh\or
54.352 \cyrc\or\cyrch\or\cyrsh\or\cyrshch\or\cyrerev\or\cyryu\or
54.353 \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).

```
54.354 %\RequirePackage{textmath}
```

54.355 \@ifundefined\{sym\cyrillicencoding letters\}\{\}\{\%
54.356 \SetSymbolFont\{\cyrillicencoding letters\}\{bold\}\cyrillicencoding
54.357 \rmdefault\bfdefault\updefault
54.358 \DeclareSymbolFontAlphabet\cyrmathrm\{\cyrillicencoding letters\}

And we need a few commands to be able to switch to different variants.

```
54.359\DeclareMathAlphabet\cyrmathbf\cyrillicencoding
54.360 \rmdefault\bfdefault\updefault
54.361 \DeclareMathAlphabet\cyrmathsf\cyrillicencoding
54.362 \sfdefault\mddefault\updefault
54.363\DeclareMathAlphabet\cyrmathit\cyrillicencoding
54.364 \rmdefault\mddefault\itdefault
54.365\DeclareMathAlphabet\cyrmathtt\cyrillicencoding
54.366 \ttdefault\mddefault\updefault
54.367 %
54.368 \SetMathAlphabet\cyrmathsf{bold}\cyrillicencoding
54.369 \sfdefault\bfdefault\updefault
54.370 \SetMathAlphabet\cyrmathit{bold}\cyrillicencoding
54.371 \rmdefault\bfdefault\itdefault
54.372}
```

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:
$54.373 \backslash \operatorname{def} \backslash \mathrm{sh}\{\backslash$ mathop $\{\backslash$ operator@font sh\}\nolimits $\}$
$\backslash \cosh$ :
$54.374 \backslash \operatorname{def} \backslash c h\{\backslash$ mathop $\{\backslash$ operator@font ch\} $\operatorname{\text {nolimits}\} }$
\tan:
$54.375 \backslash \operatorname{def} \backslash \operatorname{tg}\{\backslash$ mathop $\{\backslash$ operator@font $\operatorname{tg}\} \backslash$ nolimits $\}$
\arctan:
$54.376 \backslash \operatorname{def} \backslash \operatorname{arctg}\{\backslash$ mathop\{\operator@font arctg\}$\backslash$ nolimits $\}$
arcctg:
$54.377 \backslash$ def $\backslash \operatorname{arcctg}\{\backslash$ mathop $\{\backslash$ operator@font arcctg\} $\operatorname{anolimits\} }$

The following macro conflicts with \th defined in Latin 1 encoding:
\tanh:
54.378 \addto\extrasrussian\{\%
54.379 \babel@save\{\th\}\%
54.380 \let\ltx@th\th
$54.381 \backslash$ def $\backslash$ th $\{\backslash$ textormath $\{\backslash$ ltx@th $\} \%$
$54.382 \quad\{\backslash$ mathop\{\operator@font th\} $\backslash$ nolimits\}\}\%
$54.383\}$
\cot:
$54.384 \backslash \operatorname{def} \backslash c t g\{\backslash m a t h o p\{\backslash o p e r a t o r @ f o n t \operatorname{ctg}\} \backslash$ nolimits $\}$
\coth:
$54.385 \backslash \operatorname{def} \backslash c t h\{\backslash$ mathop $\{\backslash$ operator@font $c t h\} \backslash$ nolimits $\}$
\csc:
$54.386 \backslash$ def \cosec\{\mathop\{\operator@font cosec\}\nolimits\}
And finally some other Russian mathematical symbols:
$54.387 \backslash$ def $\backslash \operatorname{Prob}\{\backslash$ mathop $\{\backslash$ kern $\backslash$ z@ $\backslash$ mathsf $\{$ P\} $\} \backslash$ nolimits $\}$ $54.388 \backslash$ def $\backslash$ Variance $\{\backslash$ mathop $\{\backslash$ kern $\backslash z @ \backslash$ mathsf $\{D\}\} \backslash$ nolimits $\}$
$54.389 \backslash \operatorname{def} \backslash$ nod $\{\backslash \operatorname{mathop}\{\backslash c y r m a t h r m\{\backslash c y r n . \backslash c y r o . \backslash c y r d\}.\} \backslash n o l i m i t s\}$
$54.390 \backslash$ def $\backslash$ nok $\{\backslash$ mathop $\{$ \cyrmathrm\{\cyrn. \cyro. \cyrk.\}\}
$54.391 \backslash$ def $\backslash$ NOD $\{\backslash$ mathop $\{\backslash$ cyrmathrm\{ $\backslash$ CYRN $\backslash C Y R O \backslash C Y R D\}\} \backslash n o l i m i t s\} ~$
54.392 \def $\backslash N O K\{\backslash m a t h o p\{\backslash c y r m a t h r m\{\backslash C Y R N \backslash C Y R O \backslash C Y R K\}\} \backslash n o l i m i t s\}$
$54.393 \backslash \operatorname{def} \backslash \operatorname{Proj}\{\backslash$ mathop $\{\backslash$ cyrmathrm\{ $\backslash$ CYRP $\backslash c y r r\}\} \backslash$ nolimits $\}$
This is for compatibility with older Russian packages
54.394 \DeclareRobustCommand\{ \No\}\{\%
$54.395 \backslash i f m m o d e\{\backslash n f s s @ t e x t\{\backslash t e x t n u m e r o\}\} \backslash e l s e \backslash t e x t n u m e r o \backslash f i\}$
The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of @ to its original value.
54.396 \ldf@finish\{russian\}
54.397 〈/code〉

## 55 The Bulgarian language

The file bulgarian.dtx ${ }^{63}$ provides the language-specific macros for the Bulgarian language.

Users should take note of the vaious "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 29 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, allow-
    ing 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 29: The extra definitions made bybulgarian

The quotes in table 29 can also be typeset by using the commands in table 30.
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 o they keep their original names. French quotation marks may be seen as well in older books.

[^53]```
\cdash--- Cyrillic emdash in plain text.
\cdash--~ Cyrillic emdash in compound names (surnames).
\cdash--* Cyrillic emdash for denoting direct speech.
\glqq for German left double quotes (looks like,).
\grqq for German right double quotes (looks like").
"\flqq for French left double quotes (looks like <<).
\frqq for French right double quotes (looks like >>).
\dq the original quotes character (").
```

Table 30: More commands which produce quotes, defined by babel

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{bulgarian}{captionsbulgarian}
```

When this file is read as an option, i.e., by the command,bulgarianwillbean'unknown'language,inwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\l@bulgariantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
55.3\ifx\l@bulgarian\@undefined
55.4 \@nopatterns{Bulgarian}
55.5 \adddialect\l@bulgarian0
55.6 \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!

## 55.7 \let $\backslash$ latinencoding $\backslash c f @ e n c o d i n g$

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 $\mathrm{IA}_{\mathrm{E}} \mathrm{X}$ 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.

```
\def\reserved@a#1#2{%
    \edef\reserved@b{#1}%
    \edef\reserved@c{#2}%
    \ifx\reserved@b\reserved@c
        \let\cyrillicencoding\reserved@c
    \fi}
    \def\cdp@elt#1#2#3#4{%
    \reserved@a{#1}{0T2}%
    \reserved@a{#1}{LWN}%
    \reserved@a{#1}{LCY}%
    \reserved@a{#1}{X2}%
    \reserved@a{#1}{T2C}%
    \reserved@a{#1}{T2B}%
    \reserved@a{#1}{T2A}}
55.21 \reser
```

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).

```
55.23\ifx\cyrillicencoding\undefined
55.24\IfFileExists{ot2enc.def}{\def\cyrillicencoding{OT2}}\relax
55.25\IfFileExists{lwnenc.def}{\def\cyrillicencoding{LWN}}\relax
55.26 \IfFileExists{lcyenc.def}{\def\cyrillicencoding{LCY}}\relax
55.27\IfFileExists{x2enc.def}{\def\cyrillicencoding{X2}}\relax
55.28\IfFileExists{t2cenc.def}{\def\cyrillicencoding{T2C}}\relax
55.29 \IfFileExists{t2benc.def}{\def\cyrillicencoding{T2B}}\relax
55.30\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.

```
55.31 \ifx\cyrillicencoding\undefined
55.32 \PackageError{babel}%
55.33 {No Cyrillic encoding definition files were found}%
55.34 {Your installation is incomplete. \MessageBreak
55.35 You need at least one of the following files: \MessageBreak
55.36 \space\space
55.37 x2enc.def, t2aenc.def, t2benc.def, t2cenc.def, \MessageBreak
5 5 . 3 8 ~ \ s p a c e \ s p a c e
55.39 lcyenc.def, lwnenc.def, ot2enc.def.}%
55.40 \else
```

We avoid sepackage[\cyrillicencoding]\{fontenc\}becausewedon'twanttoforcetheswitchof\encodingdefault.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
55.42 \expandafter{\expandafter\input\cyrillicencoding enc.def\relax}%
55.43 \fi
55.44 \fi
\PackageInfo{babel}
                                    {Using '\cyrillicencoding' as a default Cyrillic encoding}%
55.45\DeclareRobustCommand{\Bulgarian}{%
55.46 \fontencoding\cyrillicencoding\selectfont
55.47 \let\encodingdefault\cyrillicencoding
55.48 \expandafter\set@hyphenmins\bulgarianhyphenmins
55.49 \language\l@bulgarian}
55.50\DeclareRobustCommand{\English}{%
55.51 \fontencoding\latinencoding\selectfont
55.52 \let\encodingdefault\latinencoding
55.53 \expandafter\set@hyphenmins\englishhyphenmins
55.54 \language\l@english}
55.55\let\Bul\Bulgarian
55.56\let\Bg\Bulgarian
55.57\let\cyrillictext\Bulgarian
55.58\let\cyr\Bulgarian
55.59\let\Eng\English
55.60\def\selectenglanguage{\selectlanguage{english}}
55.61\def\selectbglanguage{\selectlanguage{bulgarian}}
```

Since the X2 encoding does not contain Latin letters, we should make some redefinitions of $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ macros which implicitly produce Latin letters.

```
55.62\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 ' $\backslash$ fontencoding OT1' ( $\backslash$ fontencoding is robust).

```
\def\@Alph@eng#1{{\fontencoding{\latinencoding}\selectfont
    \ifcase#1\or 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 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 \@ctrerr\fi}}%
\def\@alph@eng#1{{\fontencoding{\latinencoding}\selectfont
    \ifcase#1\or 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 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 \@ctrerr\fi}}%
\let\@Alph\@Alph@eng
\let\@alph\@alph@eng
```

Unfortunately, the commands $\backslash \mathrm{AA}$ and $\backslash$ aa are not encoding dependent in $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$ (unlike e.g., \oe or $\backslash \mathrm{DH}$ ). They are defined as $\backslash \mathrm{r}\{\mathrm{A}\}$ and $\backslash \mathrm{r}\{\mathrm{a}\}$. This leads to unpredictable results when the font encoding does not contain the Latin letters ' $A$ ' and ' $a$ ' (like X2).

```
55.73\DeclareTextSymbolDefault{\AA}{OT1}
55.74 \DeclareTextSymbolDefault{\aa}{OT1}
```

```
55.75 \DeclareTextCommand{\AA}{OT1}{\r A}
55.76\DeclareTextCommand{\aa}{OT1}{\r a}
55.77 \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.

```
55.78 \begingroup\catcode'\\"=12
55.79 % uppercase greek letters:
55.80\def\@tempa#1{\expandafter\@tempb\meaning#1\relax\relax\relax\relax
55.81 "0000\@nil#1}
55.82\def\@tempb#1"#2#3#4#5#6\@nil#7{%
55.83 \ifnum"#2=7 \count@"1#3#4#5\relax
55.84 \ifnum\count@<"1000 \else \global\mathchardef#7="0#3#4#5\relax \fi
55.85 \fi}
55.86\@tempa\Gamma\@tempa\Delta\@tempa\Theta\@tempa\Lambda\@tempa\Xi
55.87\@tempa\Pi\@tempa\Sigma\@tempa\Upsilon\@tempa\Phi\@tempa\Psi
55.88 \@tempa\Omega
55.89% some accents:
55.90\def\@tempa#1#2\@nil{\def\@tempc{#1}}\def\@tempb{\mathaccent}
55.91 \expandafter\@tempa\hat\relax\relax\@nil
55.92\ifx\@tempb\@tempc
55.93\def\@tempa#1\@nil{#1}%
55.94\def\@tempb#1{\afterassignment\@tempa\mathchardef\@tempc=}%
55.95\def\do#1"#2{}
55.96\def\@tempd#1{\expandafter\@tempb#1\@nil
55.97 \ifnum\@tempc>"FFF
55.98 \xdef#1{\mathaccent"\expandafter\do\meaning\@tempc\space}%
55.99 \fi}
55.100\@tempd\hat\@tempd\check\@tempd\tilde\@tempd\acute\@tempd\grave
55.101 \@tempd\dot\@tempd\ddot\@tempd\breve\@tempd\bar
55.102\fi
55.103 \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 backslash\)AtBeginDocument,soinputencshouldbeusedbeforebabel.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
55.104 \@ifpackageloaded{inputenc}{}{%
55.105\def\reserved@a{LWN}%
55.106\ifx\reserved@a\cyrillicencoding\else
55.107 \def\reserved@a{OT2}%
55.108 \ifx\reserved@a\cyrillicencoding\else
55.109\PackageWarning{babel}%
55.110 {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
\latintext 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.

```
55.111% \DeclareRobustCommand{\latintext}{%
55.112% \fontencoding{\latinencoding}\selectfont
55.113% \def\encodingdefault{\latinencoding}}
55.114\let\lat\latintext
```

\textcyrillic These commands take an argument which is then typeset using the requested font \textlatin encoding. \textlatin is commented out since it is defined in the core of babel. (It is defined there with \DeclareRobustCommand instead.)

```
55.115 \DeclareTextFontCommand{\textcyrillic}{\cyrillictext}
55.116% \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 $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$. The two commands $\backslash \mathrm{cyr}$ and $\backslash$ lat activate Cyrillic resp. Latin encoding.

```
55.117 \addto\captionsbulgarian{%
55.118 \def\prefacename{%
            {\cyr\CYRP\cyrr\cyre\cyrd\cyrg\cyro\cyrv\cyro\cyrr}}%
    \def\refname{%
        {\cyr\CYRL\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
    \def\abstractname{%
        {\cyr\CYRA\cyrb\cyrs\cyrt\cyrr\cyra\cyrk\cyrt}}%
    \def\bibname{%
        {\cyr\CYRB\cyri\cyrb\cyrl\cyri\cyro\cyrg\cyrr\cyra\cyrf\cyri\cyrya}}%
    \def\chaptername{%
        {\cyr\CYRG\cyrl\cyra\cyrv\cyra}}%
    \def\appendixname{%
        {\cyr\CYRP\cyrr\cyri\cyrl\cyro\cyrzh\cyre\cyrn\cyri\cyre}}%
    \def\contentsname{%
        {\cyr\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
    \def\listfigurename{%
            {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ \cyrn\cyra\ \cyrf\cyri\cyrg\cyru\cyrr\cyri\cyrt
    \def\listtablename{%
            {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ \cyrn\cyra\\cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyrj
        \def\indexname{%
```

```
55.137
55.138
55.139
55.140
55.141
55.142
55.143
55.144
55.145
55.146
55.147
55.148
55.14
55.15
55.15
55.15
55.153
55.15
55.15
55.15
55.157
55.15
55.15
55.160}
```

\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.

```
55.161 \def\datebulgarian{%
55.162 \def\month@bulgarian{\ifcase\month\or
55.163 \cyrya\cyrn\cyru\cyra\cyrr\cyri\or
55.164 \cyrf\cyre\cyrv\cyrr\cyru\cyra\cyrr\cyri\or
55.165 \cyrm\cyra\cyrr\cyrt\or
55.166 \cyra\cyrp\cyrr\cyri\cyrl\or
55.167 \cyrm\cyra\cyrishrt\or
55.168 \cyryu\cyrn\cyri\or
55.169 \cyryu\cyrl\cyri\or
55.170 \cyra\cyrv\cyrg\cyru\cyrs\cyrt\or
55.171 \cyrs\cyre\cyrp\cyrt\cyre\cyrm\cyrv\cyrr\cyri\or
55.172 \cyro\cyrk\cyrt\cyro\cyrm\cyrv\cyrr\cyri\or
55.173 \cyrn\cyro\cyre\cyrm\cyrv\cyrr\cyri\or
55.174 \cyrd\cyre\cyrk\cyre\cyrm\cyrv\cyrr\cyri\fi}%
55.175 \def\month@Roman{\expandafter\@Roman\month}%
55.176 \def\today{\number\day~\month@bulgarian\ \number\year~\cyrg.}%
55.177 \def\todayRoman{\number\day.\,\month@Roman.\,\number\year~\cyrg.}%
55.178}
```

\todayRoman The month is often written with roman numbers in Bulgarian dates. Here we define date in this format:

```
55.179 \def\Romannumeral#1{\uppercase\expandafter{\romannumeral #1}}
55.180\def\todayRoman{\number\day.\Romannumeral{\month}.\number\year~ \cyrg.}
```

\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＇．

```
55.181 \addto\extrasbulgarian{\cyrillictext}
```

When the encoding definition file was processed by $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ the current font encoding is stored in \latinencoding，assuming that $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ uses T1 or OT1 as default．Therefore we switch back to \latinencoding whenever the Bulgarian language is no longer＇active＇．

```
55.182 \addto\noextrasbulgarian{\latintext}
```

For Bulgarian the＂character also is made active．

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

The code above is necessary because we need extra active characters．The character＂is used as indicated in table 29 ．We specify that the Bulgarian group of shorthands should be used．

```
55.184 \addto\extrasbulgarian{\languageshorthands{bulgarian}}
```

These characters are＇turned on＇once，later their definition may vary．

```
55.185 \addto\extrasbulgarian{%
55.186 \bbl@activate{"}}
55.187\addto\noextrasbulgarian{%
55.188 \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．

```
55.189 <*spanishligs〉
55.190 \declare@shorthand{bulgarian}{?`}{\UseTextSymbol{OT1}\textquestiondown}
55.191 \declare@shorthand{bulgarian}{!`}{\UseTextSymbol{OT1}\textexclamdown}
55.192〈/spanishligs〉
```

To be able to define the function of＇＂＇，we first define a couple of＇support＇ macros．
$\backslash \mathrm{dq}$ We save the original double quote character in $\backslash \mathrm{dq}$ to keep it available，the math accent \＂can now be typed as＇＂＇．
55.193 \begingroup \catcode‘\＂12

```
55.194 \def\reserved@a{\endgroup
55.195 \def\@SS{\mathchar"7019}
55.196 \def\dq{"}}
55.197 \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.

```
55.198\declare@shorthand{bulgarian}{"'}{\glqq}
55.199 \declare@shorthand{bulgarian}{"'}{\grqq}
55.200 \declare@shorthand{bulgarian}{"<}{\flqq}
55.201 \declare@shorthand{bulgarian}{">}{\frqq}
Some additional commands:
55.202 \declare@shorthand{bulgarian}{""}{\hskip\z@skip}
55.203\declare@shorthand{bulgarian}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
55.204 \declare@shorthand{bulgarian}{"=}{\nobreak-\hskip\z@skip}
55.205 \declare@shorthand{bulgarian}{"|}{%
55.206 \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
55.207 \allowhyphens}{}}
```

The next two macros for "- and "--- are somewhat different. We must check whether the second token is a hyphen character:

```
55.208\declare@shorthand{bulgarian}{"-}{%
```

If the next token is ' - ', we typeset an emdash, otherwise a hyphen sign:

```
55.209 \def\bulgarian@sh@tmp{%
55.210 \if\bulgarian@sh@next-\expandafter\bulgarian@sh@emdash
55.211 \else\expandafter\bulgarian@sh@hyphen\fi
55.212 }%
```

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.
$55.213 \backslash$ futurelet $\backslash$ bulgarian@sh@next $\backslash$ bulgarian@sh@tmp\}
Here are the definitions of hyphen and emdash. First the hyphen:

```
55.214\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 ....

```
55.215\def\bulgarian@sh@emdash#1#2{\cdash-#1#2}
```

\cdash ... these two parameters are useful for another macro: \cdash:

```
55.216 \ifx\cdash\undefined % should be defined earlier
55.217 \def\cdash#1#2#3{\def\tempx@{#3}%
55.218\def\tempa@{-}\def empb@{~}}\def empc@{*}
55.219 \ifx\tempx@\tempa@\@Acdash\else
55.220 \ifx\tempx@\tempb@\@Bcdash\else
55.221 \ifx\tempx@\tempc@\@Ccdash\else
55.222 \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 $T_{E} X$ uses horizontal mode. Maybe there is a misunderstanding? Maybe there is another way?) After a dash a breakable thinspace is always placed;

```
55.223 % What is more grammatically: .2em or . 2\fontdimen6\font?
55.224\def\@Acdash{\ifdim\lastskip>\z@\unskip\nobreak\hskip.2em\fi
55.225\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

```
55.226\def\@Bcdash{\leavevmode\ifdim\lastskip>\z@\unskip\fi
55.227 \nobreak\cyrdash\penalty\exhyphenpenalty\hskip\z@skip\ignorespaces}%
```

"--* for denoting direct speech (a space like \enskip must follow the emdash);

```
55.228\def\@Ccdash{\leavevmode
```

55.229 \nobreak\cyrdash\nobreak\hskip.35em\ignorespaces\}\%
55.230 \% $\backslash$ 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 ---.

```
55.231 % Is there an IF necessary?
55.232\ifx\cyrdash\undefined
55.233\def\cyrdash{\hbox to.8em{--\hss--}}
55.234 \fi
```

Here a really new macro - to place thinspace between initials. This macro used instead of $\backslash$, allows hyphenation in the following surname.

```
55.235\declare@shorthand{bulgarian}{",}{\nobreak\hskip.2em\ignorespaces}
```

The Bulgarian hyphenation patterns can be used with \lefthyphenmin and $\backslash$ righthyphenmin set to 2 .

```
55.236\providehyphenmins{\CurrentOption}{\tw@\tw@}
```

55.237 \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.

```
55.238\addto\extrasbulgarian{\bbl@frenchspacing}
55.239 \addto\noextrasbulgarian{\bbl@nonfrenchspacing}
```

Make the double quotes produce the traditional quotes used in Bulgarian texts (these are the German quotes).

```
55.240 % \initiate@active@char{`}
55.241% \initiate@active@char{'}
55.242% \addto\extrasbulgarian{%
55.243 % \bbl@activate{`}}
55.244% \addto\extrasbulgarian{%
55.245 % \bbl@activate{'}}
55.246 % \addto\noextrasbulgarian{%
55.247% \bbl@deactivate{`}}
55.248% \addto\noextrasbulgarian{%
55.249 % \bbl@deactivate{'}}
55.250% \def\mlron{\bbl@activate{'}\bbl@activate{'}}
55.251 % \def\mlroff{\bbl@deactivate{'}\bbl@deactivate{'}}
55.252% \declare@shorthand{bulgarian}{'`}{\glqq}
55.253 % \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 intead of Latin ones. The letters ISHRT, HRDSN and SFTSN are skipped, as usual for such enumeration.

```
55.254\def\enumBul{\let\@Alph\@Alph@bul \let\@alph\@alph@bul}
55.255\def\enumEng{\let\@Alph\@Alph@eng \let\@alph\@alph@eng}
55.256\def\enumLat{\let\@Alph\@Alph@eng \let\@alph\@alph@eng}
55.257\addto\extrasbulgarian{\enumBul}
55.258 \addto\noextrasbulgarian{\enumLat}
55.259 \def\@Alph@bul#1{%
55.260 \ifcase#1\or
55.261 \CYRA\or \CYRB\or \CYRV\or \CYRG\or \CYRD\or \CYRE\or \CYRZH\or
55.262 \CYRZ\or \CYRI\or \CYRK\or \CYRL\or \CYRM\or \CYRN\or \CYRO\or
55.263 \CYRP\or \CYRR\or \CYRS\or \CYRT\or \CYRU\or \CYRF\or \CYRH\or
55.264 \CYRC\or \CYRCH\or \CYRSH\or \CYRSHCH\or \CYRYU\or \CYRYA\else
55.265 \@ctrerr\fi
55.266 }
55.267\def\@Alph@eng#1{%
55.268 \ifcase#1\or
55.269 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
55.270 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
55.271 \@ctrerr\fi
55.272 }
```

\@alph@bul The macro \@alph@bul is similar to \@Alph@bul; it produces lowercase Bulgarian letters.

```
55.273\def\@alph@bul#1{%
55.274 \ifcase#1\or
55.275 \cyra\or \cyrb\or \cyrv\or \cyrg\or \cyrd\or \cyre\or \cyrzh\or
55.276 \cyrz\or \cyri\or \cyrk\or \cyrl\or \cyrm\or \cyrn\or \cyro\or
55.277 \cyrp\or \cyrr\or \cyrs\or \cyrt\or \cyru\or \cyrf\or \cyrh\or
55.278 \cyrc\or \cyrch\or \cyrsh\or \cyrshch\or \cyryu\or \cyrya\else
55.279 \@ctrerr\fi
55.280 }
55.281 \def\@alph@eng#1{%
55.282 \ifcase#1\or
55.283 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
55.284 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
55.285 \@ctrerr\fi
55.286 }
```

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).

```
55.287%\RequirePackage{textmath}
55.288\@ifundefined{sym\cyrillicencoding letters}{}{%
55.289 \SetSymbolFont{\cyrillicencoding letters}{bold}\cyrillicencoding
55.290 \rmdefault\bfdefault\updefault
55.291 \DeclareSymbolFontAlphabet\cyrmathrm{\cyrillicencoding letters}
```

And we need a few commands to be able to switch to different variants.

```
55.292\DeclareMathAlphabet\cyrmathbf\cyrillicencoding
55.293 \rmdefault\bfdefault\updefault
55.294\DeclareMathAlphabet\cyrmathsf\cyrillicencoding
55.295 \sfdefault\mddefault\updefault
55.296\DeclareMathAlphabet\cyrmathit\cyrillicencoding
55.297 \rmdefault\mddefault\itdefault
55.298\DeclareMathAlphabet\cyrmathtt\cyrillicencoding
55.299 \ttdefault\mddefault\updefault
55.300 \SetMathAlphabet\cyrmathsf{bold}\cyrillicencoding
55.301 \sfdefault\bfdefault\updefault
55.302 \SetMathAlphabet\cyrmathit{bold}\cyrillicencoding
55.303 \rmdefault\bfdefault\itdefault
55.304}
```

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:
$55.305 \backslash$ def $\backslash$ sh $\{\backslash$ mathop $\{\backslash$ operator@font sh\}
$\backslash$ cosh:
$55.306 \backslash \operatorname{def} \backslash c h\{\backslash$ mathop $\{\backslash$ operator@font ch\}
\tan:
$55.307 \backslash \operatorname{def} \backslash \mathrm{tg}\{\backslash$ mathop $\{\backslash$ operator@font tg$\} \backslash$ nolimits $\}$

```
\arctan: \(55.308 \backslash \operatorname{def} \backslash \operatorname{arctg}\{\backslash\) mathop\{\operator@font arctg\}\nolimits\}
\arccot: 55.309 \def \(\backslash \operatorname{arcctg}\{\backslash\) mathop \(\{\backslash\) operator@font arcctg\}
```

The following macro conflicts with \th defined in Latin 1 encoding: \tanh: 55.310 \addto\extrasrussian\{\%
55.311 \babel@save\{\th\}\%
55.312 \let\ltx@th
55.313 \def $\backslash$ th $\{\backslash$ textormath\{ $\backslash$ ltx@th\} $\%$
$55.314 \quad\{\backslash$ mathop\{\operator@font th\} $\backslash$ nolimits $\}\} \%$
$55.315\}$
\cot:
55.316 \def $\backslash c t g\{\backslash$ mathop $\{\backslash$ operator@font ctg\} $\operatorname{lnolimits\} }$
\coth:
55.317 \def \cth\{ \mathop\{\operator@font cth\}\nolimits\}
$\backslash c s c:$
$55.318 \backslash$ def $\backslash \operatorname{cosec}\{\backslash$ mathop $\{$ \operator@font cosec $\} \backslash$ nolimits $\}$
This is for compatibility with older Bulgarian packages.
55.319 \DeclareRobustCommand\{ \No\}\{\%
$55.320 \backslash i f m m o d e\{\backslash n f s s @ t e x t\{\backslash t e x t n u m e r o\}\} \backslash e l s e \backslash t e x t n u m e r o \backslash f i\}$
The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document $\}$ and resetting the category code of © to its original value.
55.321 \ldf@finish\{bulgarian\}
55.322 〈/code〉

## 56 The Ukrainian language

The file ukraineb. $\mathrm{dtx}^{64}$ defines all the language-specific macros for the Ukrainian language. It needs the file cyrcod for success documentation with Ukrainian encodings (see below).

For this language the character " is made active. In table 31 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 compund
    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, allow-
        ing 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 31: The extra definitions made by ukraineb

The quotes in table 31 (see, also table 27) can also be typeset by using the commands in table 32 (see, also table 28).

```
\cdash--- Cyrillic emdash in plain text.
\cdash--~ Cyrillic emdash in compound names (surnames).
\cdash--* Cyrillic emdash for denoting direct speech.
\glqq for German left double quotes (looks like „).
\grqq for German right double quotes (looks like ").
\(" \backslash f l q q \quad\) for French left double quotes (looks like \(\ll\) ).
\(\backslash f r q q \quad\) for French right double quotes (looks like \(\gg\) ).
\(\backslash \mathrm{dq} \quad\) the original quotes character (").
```

Table 32: More commands which produce quotes, defined by babel

[^54]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.

```
56.1 〈*code\rangle
56.2\LdfInit{ukrainian}{captionsukrainian}
```

When this file is read as an option, i.e., by the command,ukrainebwillbean'unknown'language,inwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@ukrainiantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
56.3\ifx\l@ukrainian\@undefined
56.4 \@nopatterns{Ukrainian}
56.5 \adddialect\l@ukrainian0
56.6\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!
56.7\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 X 2 , 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 $\mathrm{LA}_{\mathrm{E}} \mathrm{X}$ 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.

```
\def\reserved@a#1#2{%
    \edef\reserved@b{#1}%
    \edef\reserved@c{#2}%
    \ifx\reserved@b\reserved@c
        \let\cyrillicencoding\reserved@c
```

```
56.13 \fi}
56.14 \def\cdp@elt#1#2#3#4{%
56.15 \reserved@a{#1}{0T2}%
56.16 \reserved@a{#1}{LWN}%
56.17 \reserved@a{#1}{LCY}%
56.18 \reserved@a{#1}{X2}%
56.19 \reserved@a{#1}{T2C}%
56.20 \reserved@a{#1}{T2B}%
56.21 \reserved@a{#1}{T2A}}
56.22 \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).
6.23 \ifx\cyrillicencoding\undefined
56.24 \IfFileExists\{ot2enc.def\}\{\def\cyrillicencoding\{OT2\}\}\relax
$56.25 \backslash I f F i l e E x i s t s\{l w n e n c . d e f\}\{\backslash d e f \backslash c y r i l l i c e n c o d i n g\{L W N\}\} \backslash r e l a x$
56.26 \IfFileExists\{lcyenc.def\}\{\def\cyrillicencoding\{LCY\}\}\relax
56.27 \IfFileExists\{x2enc.def\}\{\def\cyrillicencoding\{X2\}\}\relax
56.28 \IfFileExists\{t2cenc.def\}\{\def\cyrillicencoding\{T2C\}\}\relax
56.29 \IfFileExists\{t2benc.def\}\{\def\cyrillicencoding\{T2B\}\}\relax
56.30 \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.

```
56.31 \ifx\cyrillicencoding\undefined
56.32 \PackageError{babel}%
        {No Cyrillic encoding definition files were found}%
        {Your installation is incomplete.\MessageBreak
        You need at least one of the following files:\MessageBreak
        \space\space
        x2enc.def, t2aenc.def, t2benc.def, t2cenc.def,\MessageBreak
        \space\space
        lcyenc.def, lwnenc.def, ot2enc.def.}%
\else
```

We avoid sepackage[\cyrillicencoding]\{fontenc\}becausewedon'twanttoforcetheswitchof\encodingdefault.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
56.41 \lowercase
56.42 \expandafter{\expandafter\input\cyrillicencoding enc.def\relax}%
56.43 \fi
56.44 \fi
\PackageInfo{babel}
    {Using '\cyrillicencoding' as a default Cyrillic encoding}%
```

```
56.45\DeclareRobustCommand{\Ukrainian}{%
56.46 \fontencoding\cyrillicencoding\selectfont
56.47 \let\encodingdefault\cyrillicencoding
56.48 \expandafter\set@hyphenmins\ukrainianhyphenmins
56.49 \language\l@ukrainian}%
56.50 \DeclareRobustCommand{\English}{%
56.51 \fontencoding\latinencoding\selectfont
56.52 \let\encodingdefault\latinencoding
56.53 \expandafter\set@hyphenmins\englishhyphenmins
56.54 \language\l@english}%
56.55\let\Ukr\Ukrainian
56.56\let\Eng\English
56.57\let\cyrillictext\Ukrainian
56.58\let\cyr\Ukrainian
```

Since the X2 encoding does not contain Latin letters, we should make some redefinitions of $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ macros which implicitly produce Latin letters.

```
56.59\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 ' $\backslash$ fontencoding OT1' ( $\backslash$ fontencoding is robust).

```
6.60\\def\@alph#1{{\fontencoding{\latinencoding}\selectfont
56.61 \ifcase#1\or
56.62 a\or b\or c\or d\or e\or f\or g\or h\or
56.63 i\or j\or k\or l\or m\or n\or o\or p\or
56.64 q\or r\or s\or t\or u\or v\or w\or x\or
56.65 y\or z\else\@ctrerr\fi}}%
56.66\\def\@Alph#1{{\fontencoding{\latinencoding}\selectfont
56.67 \ifcase#1\or
56.68 A\or B\or C\or D\or E\or F\or G\or H\or
56.69 I\or J\or K\or L\or M\or N\or O\or P\or
56.70 Q\or R\or S\or T\or U\or V\or W\or X\or
56.71 Y\or Z\else\@ctrerr\fi}}%
```

Unfortunately, the commands $\backslash \mathrm{AA}$ and $\backslash$ aa are not encoding dependent in $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$ (unlike e.g., \oe or $\backslash \mathrm{DH}$ ). They are defined as $\backslash \mathrm{r}\{\mathrm{A}\}$ and $\backslash \mathrm{r}\{\mathrm{a}\}$. This leads to unpredictable results when the font encoding does not contain the Latin letters ' A ' and 'a' (like X2).

```
56.72 \DeclareTextSymbolDefault{\AA}{0T1}
56.73\DeclareTextSymbolDefault{\aa}{0T1}
56.74 \DeclareTextCommand{\aa}{0T1}{\r a}
56.75 \DeclareTextCommand{\AA}{OT1}{\r A}
56.76 \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.

```
56.77% \begingroup\catcode`\"=12
56.78% % uppercase greek letters:
56.79% \def\@tempa#1{\expandafter\@tempb\meaning#1\relax\relax\relax\relax
56.80% "0000\@nil#1}
56.81% \def\@tempb#1"#2#3#4#5#6\@nil#7{%
56.82% \ifnum"#2=7 \count@"1#3#4#5\relax
56.83% \ifnum\count@<"1000 \else \global\mathchardef#7="0#3#4#5\relax \fi
56.84% \fi}
56.85% \@tempa\Gamma\@tempa\Delta\@tempa\Theta\@tempa\Lambda\@tempa\Xi
56.86 % \@tempa\Pi\@tempa\Sigma\@tempa\Upsilon\@tempa\Phi\@tempa\Psi
56.87% \@tempa\Omega
56.88% % some accents:
56.89% \def\@tempa#1#2\@nil{\def\@tempc{#1}}\def\@tempb{\mathaccent}
56.90% \expandafter\@tempa\hat\relax\relax\@nil
56.91 % \ifx\@tempb\@tempc
56.92%\\def\@tempa#1\@nil{#1}%
56.93% \def\@tempb#1{\afterassignment\@tempa\mathchardef\@tempc=}%
56.94% \def\do#1"#2{}
56.95% \def\@tempd#1{\expandafter\@tempb#1\@nil
56.96 % \ifnum\@tempc>"FFF
56.97% \xdef#1{\mathaccent"\expandafter\do\meaning\@tempc\space}%
56.98% \fi}
56.99% \@tempd\hat\@tempd\check\@tempd\tilde\@tempd\acute\@tempd\grave
56.100% \@tempd\dot\@tempd\ddot\@tempd\breve\@tempd\bar
56.101 % \fi
56.102% \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 backslash\)AtBeginDocument,soinputencshouldbeusedbeforebabel.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
56.103\@ifpackageloaded{inputenc}{}{%
56.104 \def\reserved@a{LWN}%
56.105 \ifx\reserved@a\cyrillicencoding\else
56.106 \def\reserved@a{0T2}%
56.107 \ifx\reserved@a\cyrillicencoding\else
56.108 \PackageWarning{babel}%
56.109 {No input encoding specified for Ukrainian language}
56.110 \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 \latintext 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.

```
56.111%\DeclareRobustCommand{\latintext}{%
56.112% \fontencoding{\latinencoding}\selectfont
56.113% \def\encodingdefault{\latinencoding}}
56.114\let\lat\latintext
```

\textcyrillic These commands take an argument which is then typeset using the requested font \textlatin encoding.

```
56.115\DeclareTextFontCommand{\textcyrillic}{\cyrillictext}
56.116 %\DeclareTextFontCommand{\textlatin}{\latintext}
```

We make the $\mathrm{T}_{\mathrm{E}} \mathrm{X}$
$56.117 \%$ ifx $\backslash$ ltxTeX $\backslash$ undefined $\backslash l e t \backslash l t x T e X \backslash T e X \backslash f i$ $56.118 \% \backslash$ ProvideTextCommandDefault $\{\backslash$ TeX $\}\{\backslash$ textlatin $\{\backslash$ ltxTeX $\}\}$
and $\mathrm{A}_{\mathrm{E}} \mathrm{X}$ logos encoding independent.
$56.119 \%$ ifx $\backslash$ ltxLaTeX $\backslash u n d e f i n e d \backslash l e t \backslash l t x L a T e X \backslash L a T e X \backslash f i$ $56.120 \%$ \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 $\mathrm{AN}_{\mathrm{E}} \mathrm{X}$. The two commands $\backslash \mathrm{cyr}$ and $\backslash$ lat activate Cyrillic resp. Latin encoding.

```
56.121 \addto\captionsukrainian{%
56.122 \def\prefacename{{\cyr\CYRV\cyrs\cyrt\cyru\cyrp}}%
56.123% \def\prefacename{{\cyr\CYRP\cyre\cyrr\cyre\cyrd\cyrm\cyro\cyrv\cyra}}%
56.124\\def\refname{%
56.125 {\cyr\CYRL\cyrii\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
56.126 % \def\refname{%
56.127% {\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
56.128% \ \cyrp\cyro\cyrs\cyri\cyrl\cyra\cyrn\cyrsftsn}}%
56.129 \def\abstractname{%
56.130 {\cyr\CYRA\cyrn\cyro\cyrt\cyra\cyrc\cyrii\cyrya}}%
56.131 % \def\abstractname{{\cyr\CYRR\cyre\cyrf\cyre\cyrr\cyra\cyrt}}%
56.132 \def\bibname{%
56.133 {\cyr\CYRB\cyrii\cyrb\cyrl\cyrii\cyro\cyrgup\cyrr\cyra\cyrf\cyrii\cyrya}}%
56.134% \def\bibname{{\cyr\CYRL\cyrii\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
56.135 \def\chaptername{{\cyr\CYRR\cyro\cyrz\cyrd\cyrii\cyrl}}%
56.136% \def\chaptername{{\cyr\CYRG\cyrl\cyra\cyrv\cyra}}%
56.137 \def\appendixname{{\cyr\CYRD\cyro\cyrd\cyra\cyrt\cyro\cyrk}}%
56.138\def\contentsname{{\cyr\CYRZ\cyrm\cyrii\cyrs\cyrt}}%
56.139 \def\listfigurename{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
                            \ \cyrii\cyrl\cyryu\cyrs\cyrt\cyrr\cyra\cyrc\cyrii\cyrishrt}}%
56.140 \ \cyrii\cyrl\cyryu\cyrs\cyrt\cyrr\cyra\cyrc\cyrii\cy
\def\listtablename{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
                    \ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyrsftsn}}%
    \def\indexname{{\cyr\CYRP\cyro\cyrk\cyra\cyrzh\cyrch\cyri\cyrk}}%
    \def\authorname{{\cyr\CYRII\cyrm\cyre\cyrn\cyrn\cyri\cyrishrt
                    \ \cyrp\cyro\cyrk\cyra\cyrzh\cyrch\cyri\cyrk}}%
    \def\figurename{{\cyr\CYRR\cyri\cyrs.}}%
```

```
56.147% \def\figurename{\cyr\CYRR\cyri\cyrs\cyru\cyrn\cyro\cyrk}}%
56.148 \def\tablename{{\cyr\CYRT\cyra\cyrb\cyrl.}}%
56.149 % \def\tablename{\cyr\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyrya}}%
56.150 \def\partname{{\cyr\CYRCH\cyra\cyrs\cyrt\cyri\cyrn\cyra}}%
56.151 \def\enclname{{\cyr\cyrv\cyrk\cyrl\cyra\cyrd\cyrk\cyra}}%
56.152 \def\ccname{{\cyr\cyrk\cyro\cyrp\cyrii\cyrya}}%
56.153 \def\headtoname{{\cyr\CYRD\cyro}}%
56.154 \def\pagename{{\cyr\cyrs.}}%
56.155% \def\pagename{{\cyr\cyrs\cyrt\cyro\cyrr\cyrii\cyrn\cyrk\cyra}}%
56.156 \def\seename{{\cyr\cyrd\cyri\cyrv.}}%
56.157 \def\alsoname{{\cyr\cyrd\cyri\cyrv.\ \cyrt\cyra\cyrk\cyro\cyrzh}}
56.158 \def\proofname{{\cyr\CYRD\cyro\cyrv\cyre\cyrd\cyre\cyrn\cyrn\cyrya}}%
56.159 \def\glossaryname{{\cyr\CYRS\cyrl\cyro\cyrv\cyrn\cyri\cyrk\ %
56.160 \cyrt\cyre\cyrr\cyrm\cyrii\cyrn\cyrii\cyrv}}%
56.161 }
```

\dateukrainian The macro \dateukrainian redefines the command \today to produce Ukrainian dates.

```
56.162\def\dateukrainian{%
56.163\\def\today{\number\day \\ifcase\month\or
56.164 \cyrs\cyrii\cyrch\cyrn\cyrya\or
56.165 \cyrl\cyryu\cyrt\cyro\cyrg\cyro\or
56.166 \cyrb\cyre\cyrr\cyre\cyrz\cyrn\cyrya\or
56.167 \cyrk\cyrv\cyrii\cyrt\cyrn\cyrya\or
56.168 \cyrt\cyrr\cyra\cyrv\cyrn\cyrya\or
56.169 \cyrch\cyre\cyrr\cyrv\cyrn\cyrya\or
56.170 \cyrl\cyri\cyrp\cyrn\cyrya\or
56.171 \cyrs\cyre\cyrr\cyrp\cyrn\cyrya\or
56.172 \cyrv\cyre\cyrr\cyre\cyrs\cyrn\cyrya\or
56.173 \cyrzh\cyro\cyrv\cyrt\cyrn\cyrya\or
56.174 \cyrl\cyri\cyrs\cyrt\cyro\cyrp\cyra\cyrd\cyra\or
56.175 \cyrg\cyrr\cyru\cyrd\cyrn\cyrya\fi
56.176 \space\number\year~\cyrr.}}
```

\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'.
56.177 \addto\extrasukrainian\{\cyrillictext\}

When the encoding definition file was processed by $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ the current font encoding is stored in \latinencoding, assuming that $\mathrm{EAT}_{\mathrm{E}} \mathrm{u}$ uses T 1 or OT1 as default. Therefore we switch back to \latinencoding whenever the Ukrainian language is no longer 'active'.
56.178 \addto \noextrasukrainian\{\latintext\}

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>

```
56.179 \addto\extrasukrainian{\lccode'\'='\'}
56.180 \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 $\mathrm{EAT}_{\mathrm{EX}}$ command somewhat：

```
56.181%\def\verbatim@font{%
56.182% \let\encodingdefault\latinencoding
56.183 % \normalfont\ttfamily
56.184% \expandafter\def\csname\cyrillicencoding-cmd\endcsname##1##2{%
56.185% \ifx\protect\@typeset@protect
56.186 % \begingroup\UseTextSymbol\cyrillicencoding##1\endgroup
56.187% \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 re－ quires 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 $\backslash$ AutoSpaceBeforeFDP switches back to the default behavior of ukraineb．

```
56.188 <*frenchpunct\rangle
```

56.189 \initiate@active@char\{:\}
56.190 \initiate@active@char\{;\}
56.191 〈/frenchpunct〉
56.192 〈 $*$ frenchpunct | spanishligs〉
56.193 \initiate@active@char\{!\}
56.194 \initiate@active@char\{?\}
56.195 〈/frenchpunct | spanishligs〉
56.196 \initiate@active@char\{"\}

The code above is necessary because we need extra active characters．The character＂is used as indicated in table 31.

We specify that the Ukrainian group of shorthands should be used．

```
56.197 \addto\extrasukrainian{\languageshorthands{ukrainian}}
```

These characters are＇turned on＇once，later their definition may vary．

```
56.198 \addto\extrasukrainian{%
56.199 <frenchpunct\rangle \bbl@activate{:}\bbl@activate{;}%
56.200 <frenchpunct|spanishligs\rangle \bbl@activate{!}\bbl@activate{?}%
```

```
56.201 \bbl@activate{"}}
56.202 \addto\noextrasukrainian{%
56.203 <frenchpunct\rangle \bbl@deactivate{:}\bbl@deactivate{;}%
56.204 <frenchpunct | spanishligs\rangle \bbl@deactivate{!}\bbl@deactivate{?}%
56.205 \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.

```
56.206 <*spanishligs\rangle
56.207\declare@shorthand{ukrainian}{?`}{\UseTextSymbol{OT1}\textquestiondown}
56.208 \declare@shorthand{ukrainian}{!`}{\UseTextSymbol{OT1}\textexclamdown}
56.209 〈/spanishligs〉
```

\ukrainian@sh@;@ We have to reduce the amount of white space before ; , : and !. This should only \ukrainian@sh@:@ happen in horizontal mode, hence the test with \ifhmode.

```
\ukrainian@sh@!@6.210 <*frenchpunct>
```

\ukrainian@sh@?@6.211 \declare@shorthand\{ukrainian\}\{;\}\{\%
56.212 \ifhmode

In horizontal mode we check for the presence of a 'space', 'unskip' if it exists and place a 0.1 em kerning.

```
56.213 \ifdim\lastskip>\z@
56.214 \unskip\nobreak\kern.1em
56.215 \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.

```
56.216 \FDP@thinspace
56.217 \fi
56.218 \fi
```

Now we can insert a ';' character
56.219 \string;\}

The other definitions are very similar.
56.220 \declare@shorthand\{ukrainian\}\{:\}\{\%
56.221 \ifhmode
56.222 \ifdim\lastskip>\z@
$56.223 \backslash$ unskip\nobreak $\backslash$ kern.1em
56.224 \else
$56.225 \quad \backslash F D P @ t h i n s p a c e$
$56.226 \quad \backslash f i$
56.227 \fi
56.228 \string:\}

| 56.229 | \declare＠shorthand\｛ukrainian\}\{!\}\{\% |
| :---: | :---: |
| 56.230 | \ifhmode |
| 56.231 | \ifdim\lastskip＞\z＠ |
| 56.232 | \unskip\nobreak ${ }^{\text {kern．1em }}$ |
| 56.233 | $\backslash e l s e$ |
| 56.234 | $\backslash F D P @ t h i n s p a c e$ |
| 56.235 | $\backslash f i$ |
| 56.236 | $\backslash f i$ |
| 56.237 | \string！\} |
| 56.238 | \declare＠shorthand\｛ukrainian\}\{?\}\{\% |
| 56.239 | \ifhmode |
| 56.240 | \ifdim\lastskip＞\z＠ |
| 56.241 | \unskip\nobreak\kern．1em |
| 56.242 | $\backslash e l s e$ |
| 56.243 | $\backslash F D P @ t h i n s p a c e$ |
| 56.244 | \fi |
| 56.245 | \fi |
| 56.246 | \string？\} |

$\backslash$ AutoSpaceBeforeFDP $\backslash$ FDP＠thinspace is defined as unbreakable spaces if $\backslash$ AutoSpaceBeforeFDP is $\backslash$ NoAutoSpaceBeforeFDP activated or as \＠empty if \NoAutoSpaceBeforeFDP is in use．The default is $\backslash$ FDP＠thinspace \AutoSpaceBeforeFDP．

```
56.247\def\AutoSpaceBeforeFDP{%
56.248 \def\FDP@thinspace{\nobreak\kern.1em}}
56.249 \def\NoAutoSpaceBeforeFDP{\let\FDP@thinspace\@empty}
56.250 \AutoSpaceBeforeFDP
```

$\backslash$ FDPon The next macros allow to switch on／off activeness of double punctuation signs．

| $\backslash$ FDPoff ${ }_{56.251} \backslash$ def $\backslash$ FDPon\｛ $\backslash \mathrm{bbl@activate} \mathrm{\{:} \mathrm{\}} \mathrm{\%}$ |  |
| :---: | :---: |
| 56.252 | \bbl＠activate\｛；\}\% |
| 56.253 | \bbl＠activate\｛？\}\% |
| 56.254 | \bbl＠activate\｛！\}\} |
| 56.255 | \def \FDPoff\｛\bbl＠deactivate\｛：\}\% |
| 56.256 | \bbl＠deactivate\｛；\}\% |
| 56.257 | \bbl＠deactivate\｛？\}\% |
| 56.258 | \bbl＠deactivate\｛！\}\} |

\system＠sh＠：＠When the active characters appear in an environment where their Ukrainian be－ \system＠sh＠！＠haviour is not wanted they should give an＇expected＇result．Therefore we define \system＠sh＠？＠shorthands at system level as well．
\system＠sh＠；＠6．259 \declare＠shorthand\｛system\}\{:\}\{\string:\}
56.260 \declare＠shorthand\｛system\}\{;\}\{\string;\}
56.261 〈／frenchpunct〉
56.262 〈＊frenchpunct\＆！spanishligs〉
56.263 \declare＠shorthand\｛system\}\{!\}\{\string!\}
56.264 \declare＠shorthand\｛system\}\{?\}\{\string?\}
56.265 〈／frenchpunct\＆！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 '"'

```
56.266 \begingroup \catcode`\"12
56.267\def\reserved@a{\endgroup
56.268 \def\@SS{\mathchar"7019 }
56.269 \def\dq{"}}
56.270 \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.

```
56.271 \declare@shorthand{ukrainian}{"'}{\glqq}
56.272\declare@shorthand{ukrainian}{"'}{\grqq}
56.273\declare@shorthand{ukrainian}{"<}{\flqq}
56.274\declare@shorthand{ukrainian}{">}{\frqq}
```

Some additional commands:

```
56.275\declare@shorthand{ukrainian}{""}{\hskip\z@skip}
56.276 \declare@shorthand{ukrainian}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
56.277 \declare@shorthand{ukrainian}{"=}{\nobreak-\hskip\z@skip}
56.278 \declare@shorthand{ukrainian}{"|}{%
56.279 \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
56.280
    \allowhyphens}{}}
```

The next two macros for "- and "--- are somewhat different. We must check whether the second token is a hyphen character:

```
56.281 \declare@shorthand{ukrainian}{"-}{%
```

If the next token is ' - ', we typeset an emdash, otherwise a hyphen sign:

```
56.282 \def\ukrainian@sh@tmp{%
56.283 \if\ukrainian@sh@next-\expandafter\ukrainian@sh@emdash
56.284 \else\expandafter\ukrainian@sh@hyphen\fi
56.285 }%
```

$\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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.
56.286 \futurelet\ukrainian@sh@next\ukrainian@sh@tmp\}

Here are the definitions of hyphen and emdash. First the hyphen:
56.287 \def \ukrainian@sh@hyphen\{\%
56.288 \nobreak\-\bbl@allowhyphens\}

For the emdash definition, there are the two parameters: we must 'eat' two last hyphen signs of our emdash...
56.289 \def \ukrainian@sh@emdash\#1\#2\{\cdash-\#1\#2\}

```
\cdash ... these two parameters are useful for another macro: \cdash:
56.290 %\ifx\cdash\undefined % should be defined earlier
56.291 \def\cdash#1#2#3{\def\tempx@{#3}%
56.292 \def\tempa@{-}\def\tempb@{~}\def\tempc@{*}%
56.293 \ifx\tempx@\tempa@\@Acdash\else
56.294 \ifx\tempx@\tempb@\@Bcdash\else
56.295 \ifx\tempx@\tempc@\@Ccdash\else
56.296 \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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ uses horizontal mode. Maybe there is a misunderstanding? Maybe there is another way?) After a dash a breakable thinspace is always placed;

```
56.297% What is more grammatically: .2em or .2\fontdimen6\font ?
56.298\def\@Acdash{\ifdim\lastskip>\z@\unskip\nobreak\hskip.2em\fi
56.299 \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

```
56.300\def\@Bcdash{\leavevmode\ifdim\lastskip>\z@\unskip\fi
```

56.301 \nobreak\cyrdash\penalty\exhyphenpenalty\hskip\z@skip\ignorespaces\}\%
"--* for denoting direct speech (a space like \enskip must follow the emdash);

```
56.302\def\@Ccdash{\leavevmode
56.303 \nobreak\cyrdash\nobreak\hskip.35em\ignorespaces}%
56.304 %\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 ---

```
56.305% Is there an IF necessary?
56.306 \ifx\cyrdash\undefined
56.307 \def\cyrdash{\hbox to.8em{--\hss--}}
56.308\fi
```

Here a really new macro - to place thinspace between initials. This macro used instead of $\backslash$, allows hyphenation in the following surname.

[^55]```
\mdqon All that's left to do now is to define a couple of commands for ".
\mdqoff6.310 \def\mdqon{\bbl@activate{"}}
    56.311 \def\mdqoff{\bbl@deactivate{"}}
```

The Ukrainian hyphenation patterns can be used with \lefthyphenmin and $\backslash$ righthyphenmin set to 2.
$56.312 \backslash$ providehyphenmins $\{\backslash$ CurrentOption $\}\{\backslash$ tw@ $\backslash$ tw $@\}$
56.313 \% temporary hack:
56.314 \ifx $\backslash$ englishhyphenmins \undefined
56.315 \def $\backslash e n g l i s h h y p h e n m i n s\{\backslash t w @ \backslash t h r @ @\}$
56.316 \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.

```
56.317 \addto\extrasukrainian{\bbl@frenchspacing}
56.318 \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.
$\backslash$ Asbuk We begin by defining \Asbuk which works like $\backslash$ Alph, but produces (uppercase) Cyrillic letters intead of Latin ones. The letters CYRGUP, and SFTSN are skipped, as usual for such enumeration.

```
56.319\def\Asbuk#1{\expandafter\@Asbuk\csname c@#1\endcsname}
56.320\def\@Asbuk#1{\ifcase#1\or
56.321 \CYRA\or\CYRB\or\CYRV\or\CYRG\or\CYRD\or\CYRE\or\CYRIE\or
56.322 \CYRZH\or\CYRZ\or\CYRI\or\CYRII\or\CYRYI\or\CYRISHRT\or
56.323\CYRK\or\CYRL\or\CYRM\or\CYRN\or\CYRO\or\CYRP\or\CYRR\or
56.324\CYRS\or\CYRT\or\CYRU\or\CYRF\or\CYRH\or\CYRC\or\CYRCH\or
56.325\CYRSH\or\CYRSHCH\or\CYRYU\or\CYRYA\else\@ctrerr\fi}
```

\asbuk The macro \asbuk is similar to \alph; it produces lowercase Ukrainian letters.

```
56.326\def\asbuk#1{\expandafter\@asbuk\csname c@#1\endcsname}
56.327\def\@asbuk#1{\ifcase#1\or
56.328 \cyra\or\cyrb\or\cyrv\or\cyrg\or\cyrd\or\cyre\or\cyrie\or
56.329 \cyrzh\or\cyrz\or\cyri\or\cyrii\or\cyryi\or\cyrishrt\or
56.330 \cyrk\or\cyrl\or\cyrm\or\cyrn\or\cyro\or\cyrp\or\cyrr\or
56.331 \cyrs\or\cyrt\or\cyru\or\cyrf\or\cyrh\or\cyrc\or\cyrch\or
56.332 \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).

```
56.334\@ifundefined{sym\cyrillicencoding letters}{}{%
56.335\SetSymbolFont{\cyrillicencoding letters}{bold}\cyrillicencoding
56.336 \rmdefault\bfdefault\updefault
56.337\DeclareSymbolFontAlphabet\cyrmathrm{\cyrillicencoding letters}
```

And we need a few commands to be able to switch to different variants.

```
56.338\DeclareMathAlphabet\cyrmathbf\cyrillicencoding
56.339 \rmdefault\bfdefault\updefault
56.340\DeclareMathAlphabet\cyrmathsf\cyrillicencoding
56.341 \sfdefault\mddefault\updefault
56.342\DeclareMathAlphabet\cyrmathit\cyrillicencoding
56.343 \rmdefault\mddefault\itdefault
56.344\DeclareMathAlphabet\cyrmathtt\cyrillicencoding
56.345 \ttdefault\mddefault\updefault
56.346 %
56.347\SetMathAlphabet\cyrmathsf{bold}\cyrillicencoding
56.348 \sfdefault\bfdefault\updefault
56.349 \SetMathAlphabet\cyrmathit{bold}\cyrillicencoding
56.350 \rmdefault\bfdefault\itdefault
56.351}
```

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.
$\backslash$ sinh:
$56.352 \backslash \operatorname{def} \backslash \mathrm{sh}\{\backslash$ mathop $\{\backslash$ operator@font sh\}\nolimits $\}$
$\backslash \cosh$ :
$56.353 \backslash \operatorname{def} \backslash c h\{\backslash$ mathop $\{\backslash$ operator@font ch\} $\operatorname{cnolimits\} }$
\tan:
$56.354 \backslash \operatorname{def} \backslash \mathrm{tg}\{\backslash$ mathop $\{\backslash$ operator@font tg$\} \backslash$ nolimits $\}$
\arctan:
$56.355 \backslash \operatorname{def} \backslash \operatorname{arctg}\{\backslash$ mathop\{\operator@font arctg\}$\backslash$ nolimits $\}$
arcctg:
56.356 \def $\backslash \operatorname{arcctg}\{\backslash$ mathop $\{\backslash$ operator@font arcctg\}

The following macro conflicts with $\backslash$ th defined in Latin 1 encoding
\tanh:
56.357 \addto\extrasrussian\{\%
56.358 \babel@save\{\th\}\%
56.359 \let\ltx@th \th
56.360 \def $\backslash$ th $\{\backslash$ textormath $\{\backslash$ ltx@th\}\%
56.361 \{\mathop\{\operator@font th\} $\backslash$ nolimits\}\}\%
$56.362\}$
$\backslash \cot$ :
$56.363 \backslash \operatorname{def} \backslash c t g\{\backslash$ mathop $\{\backslash$ operator@font ctg$\} \backslash$ nolimits $\}$
\coth:
$56.364 \backslash \operatorname{def} \backslash c t h\{\backslash$ mathop $\{\backslash$ operator@font $c t h\} \backslash$ nolimits $\}$

```
    \csc:
56.365\def\cosec{\mathop{\operator@font cosec}\nolimits}
    And finally some other Ukrainian and Russian mathematical symbols:
56.366\def\Prob{\mathop{\kern\z@\mathsf{P}}\nolimits}
56.367\def\Variance{\mathop{\kern\z@\mathsf{D}}\nolimits}
56.368\def\nsd{\mathop{\cyrmathrm{\cyrn.\cyrs.\cyrd.}}\nolimits}
56.369\def\nsk{\mathop{\cyrmathrm{\cyrn.\cyrs.\cyrk.}}\nolimits}
56.370\def\NSD{\mathop{\cyrmathrm{\CYRN\CYRS\CYRD}}\nolimits}
56.371\def\NSK{\mathop{\cyrmathrm{\CYRN\CYRS\CYRK}}\nolimits}
56.372 \def\nod{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrd.}}\nolimits} % ??????
56.373 \def\nok{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrk.}}\nolimits} % ??????
56.374 \def\NOD{\mathop{\cyrmathrm{\CYRN\CYRO\CYRD}}\nolimits} % ??????
56.375 \def\NOK{\mathop{\cyrmathrm{\CYRN\CYRO\CYRK}}\nolimits} % ??????
56.376\def\Proj{\mathop{\cyrmathrm{\CYRP\cyrr}}\nolimits}
This is for compatibility with older Ukrainian packages.
56.377 \DeclareRobustCommand{\No}{%
56.378 \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.
```

```
56.379 \ldf@finish{ukrainian}
```

56.379 \ldf@finish{ukrainian}
56.380 </code〉

```
56.380 </code〉
```


## 57 The Lower Sorbian language

The file lsorbian.dtx ${ }^{65}$ 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.

```
57.1 〈*code〉
57.2\LdfInit{lsorbian}\captionslsorbian
```

When this file is read as an option, i.e. by the command,lsorbianwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\l@lsorbiantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
57.3\ifx\l@lsorbian\@undefined
57.4 \@nopatterns{Lsorbian}
57.5 \adddialect\l@lsorbian\l@usorbian\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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$.

```
57.6 \addto\captionslsorbian{%
57.7 \def\prefacename{Zawod}%
57.8\\def\refname{Referency}%
57.9 \def\abstractname{Abstrakt}%
57.10\def\bibname{Literatura}%
57.11 \def\chaptername{Kapitl}%
57.12\def\appendixname{Dodawki}%
57.13\\def\contentsname{Wop\'simje\'se}%
57.14 \def\listfigurename{Zapis wobrazow}%
57.15\def\listtablename{Zapis tabulkow}%
57.16 \def\indexname{Indeks}%
57.17 \def\figurename{Wobraz}%
57.18 \def\tablename{Tabulka}%
57.19 \def\partname{\'Z\v el}%
57.20 \def\enclname{P\'si\l oga}%
57.21 \def\ccname{CC}%
57.22 \def\headtoname{Komu}%
57.23\def\pagename{Strona}%
57.24 \def\seename{gl.}%
57.25\def\alsoname{gl.~}\mp@subsup{}{}{~
57.26 \def\proofname{Proof}% <-- needs translation
57.27 \def\glossaryname{Glossary}% <-- Needs translation
57.28 }%
```

[^56]\newdatelsorbian The macro \newdatelsorbian redefines the command \today to produce Lower Sorbian dates.

```
57.29\def \newdatelsorbian{%
57.30 \def\today{\number\day.~\ifcase\month\or
57.31 januara\or februara\or m\v erca\or apryla\or maja\or
57.32 junija\or julija\or awgusta\or septembra\or oktobra\or
57.33 nowembra\or decembra\fi
57.34 \space \number\year}}
```

\olddatelsorbian The macro \olddatelsorbian redefines the command \today to produce old-style Lower Sorbian dates.

```
57.35 \def\olddatelsorbian{%
57.36\def\today{\number\day. \\ifcase\month\or
57.37 wjelikego ro\v zka\or
57.38 ma\l ego ro\v zka\or
57.39 nal\v etnika\or
57.40 jat\v sownika\or
57.41 ro\v zownika\or
57.42 sma\v znika\or
57.43 pra\v znika\or
57.44 \v znje\'nca\or
57.45 po\v znje\'nca\or
57.46 winowca\or
57.47 nazymnika\or
57.48 godownika\fi \space \number\year}}
```

The default will be the new-style dates.
57.49 \let $\backslash d a t e l$ sorbian $\backslash$ newdatelsorbian
\extraslsorbian The macro \extraslsorbian will perform all the extra definitions needed for the \noextraslsorbian lsorbian 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.

```
57.50 \addto\extraslsorbian{}
57.51 \addto\noextraslsorbian{}
```

The macro \ldf@finish takes care of looking for a configuration file, setting the main language to be switched on at $\backslash$ begin\{document\} and resetting the category code of © to its original value.

```
57.52\ldf@finish{lsorbian}
57.53 \/code\
```


## 58 The Upper Sorbian language

The file usorbian.dtx ${ }^{66}$ 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.

```
58.1 <*code〉
```

$58.2 \backslash$ LdfInit\{usorbian\}\captionsusorbian

When this file is read as an option, i.e. by the command,usorbianwillbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\l@usorbiantoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
58.3\ifx\l@usorbian\@undefined
58.4 \@nopatterns{Usorbian}
58.5 \adddialect\l@usorbian0\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 $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$.

```
58.6 \addto\captionsusorbian{%
58.7 \def\prefacename{Zawod}%
58.8\def\refname{Referency}%
58.9 \def\abstractname{Abstrakt}%
58.10\def\bibname{Literatura}%
58.11 \def\chaptername{Kapitl}%
58.12\def\appendixname{Dodawki}%
58.13\def\contentsname{Wobsah}%
58.14 \def\listfigurename{Zapis wobrazow}%
58.15\def\listtablename{Zapis tabulkow}%
58.16 \def\indexname{Indeks}%
58.17 \def\figurename{Wobraz}%
58.18 \def\tablename{Tabulka}%
58.19 \def\partname{D\'z\v el}%
58.20 \def\enclname{P\v r\l oha}%
58.21 \def\ccname{CC}%
58.22 \def\headtoname{Komu}%
58.23\def\pagename{Strona}%
58.24 \def\seename{hl.}%
58.25\def\alsoname{hl.~
58.26 \def\proofname{Proof}% <-- needs translation
58.27 \def\glossaryname{Glossary}% <-- Needs translation
58.28 }%
```

[^57]\newdateusorbian The macro \newdateusorbian redefines the command \today to produce Upper Sorbian dates.

```
58.29\def\newdateusorbian{%
58.30 \def\today{\number\day. ~\ifcase\month\or
58.31 januara\or februara\or m\v erca\or apryla\or meje\or junija\or
58.32 julija\or awgusta\or septembra\or oktobra\or
58.33 nowembra\or decembra\fi
58.34 \space \number\year}}
```

\olddateusorbian The macro \olddateusorbian redefines the command \today to produce old-style Upper Sorbian dates.

```
8.35 \def\olddateusorbian{%
58.36 \def\today{\number\day. ~\ifcase\month\or
58.37 wulkeho r\'o\v zka\or ma\l eho r\'o\v zka\or nal\v etnika\or
58.38 jutrownika\or r\'o\v zownika\or sma\v znika\or pra\v znika\or
58.39 \v znjenca\or po\v znjenca\or winowca\or nazymnika\or
58.40 hodownika\fi \space \number\year}}
```

The default will be the new-style dates.
58.41 \let $\backslash$ dateusorbian $\backslash$ newdateusorbian
\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.

```
58.42 \initiate@active@char{"}
58.43\addto\extrasusorbian{\languageshorthands{usorbian}}
58.44 \addto\extrasusorbian{\bbl@activate{"}}
```

Don't forget to turn the shorthands off again.
58.45 \addto\noextrasusorbian\{\bbl@deactivate\{"\}\}

In order for $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ to be able to hyphenate German Upper Sorbian words which contain ' $\beta$ ' we have to give the character a nonzero \lccode (see Appendix H, the TEXbook).
58.46 \addto\extrasusorbian\{\babel@savevariable\{\lccode‘\~~Y\}\%
58.47 \lccode‘\~~Y‘\~~Y\}

The umlaut accent macro $\backslash "$ is changed to lower the umlaut dots. The redefinition is done with the help of \umlautlow.
58.48 \addto\extrasusorbian\{ \babel@save\"\umlautlow\}
58.49 \addto\noextrasusorbian\{\umlauthigh\}

The Upper Sorbian hyphenation patterns can be used with \lefthyphenmin and \righthyphenmin set to 2 .

\dq We save the original double quote character in \dq to keep it available, the math accent $\backslash "$ can now be typed as ". Also we store the original meaning of the command \" for future use.

```
58.51 \begingroup \catcode'\"12
58.52\def\x{\endgroup
58.53\\def\@SS{\mathchar"7019 }
58.54 \def\dq{"}}
58.55\x
```

Now we can define the doublequote macros: the umlauts,

```
58.56 \declare@shorthand{usorbian}{"a}{\textormath{\"{a}}{\ddot a}}
```

58.57 \declare@shorthand\{usorbian\}\{"o\}\{\textormath $\{\backslash$ "\{o\}\}\{\ddot o\}\}
$58.58 \backslash$ declare@shorthand\{usorbian\}\{"u\}\{\textormath $\{\backslash$ "\{u\}\}\{\ddot u\}\}
58.59 \declare@shorthand\{usorbian\}\{"A\}\{\textormath $\{\backslash$ "\{A\}\}\{\ddot A\}\}
$58.60 \backslash$ declare@shorthand\{usorbian\}\{"0\}\{\textormath $\{\backslash$ "\{0\}\}\{\ddot 0$\}\}$
58.61 \declare@shorthand\{usorbian\}\{"U\}\{\textormath $\{\backslash$ " $\{U\}\}\{\backslash$ ddot $U\}\}$
tremas,

```
58.62\declare@shorthand{usorbian}{"e}{\textormath{\"{e}}{\ddot e}}
```

58.63 \declare@shorthand\{usorbian\}\{"E\}\{\textormath $\{\backslash$ " $\{\mathrm{E}\}\}\{\backslash$ ddot E$\}\}$
58.64 \declare@shorthand\{usorbian\}\{"i\}\{\textormath $\{\backslash$ " $\{\backslash i\}\}\{\backslash d d o t \backslash i m a t h\}\}$
58.65 \declare@shorthand\{usorbian\}\{"I\}\{\textormath $\{\backslash$ "\{I\}\}\{\ddot I\}\}
usorbian es-zet (sharp s),
$58.66 \backslash$ declare@shorthand\{usorbian\}\{"s\}\{\textormath $\{\backslash \mathrm{ss}\}\}\{\backslash @ S S\}\}\}$
58.67 \declare@shorthand\{usorbian\}\{"S\}\{SS\}
german and french quotes,
58.68 \declare@shorthand\{usorbian\}\{"'\}\{\%
58.69 \textormath\{\quotedblbase\}\{\mbox\{\quotedblbase\}\}\}
$58.70 \backslash$ declare@shorthand\{usorbian\}\{"'\}\{\%
58.71 \textormath $\{\backslash$ textquotedblleft $\}\{\backslash$ mbox $\{\backslash$ textquotedblleft $\}\}\}$
58.72 \declare@shorthand\{usorbian\}\{"<\}\{\%
$58.73 \backslash$ textormath $\{\backslash$ guillemotleft $\}\{\backslash$ mbox $\{\backslash$ guillemotleft $\}\}\}$
58.74 \declare@shorthand\{usorbian\}\{">\}\{\%
58.75 \textormath $\{\backslash$ guillemotright $\}\{\backslash$ mbox $\{\backslash$ guillemotright $\}\}\}$
discretionary commands
58.76 \declare@shorthand\{usorbian\}\{"c\}\{\textormath\{\bbl@disc ck\}\{c\}\}
58.77 \declare@shorthand\{usorbian\}\{"C\}\{\textormath\{\bbl@disc CK\}\{C\}\}
58.78 \declare@shorthand\{usorbian\}\{"f\}\{\textormath\{\bbl@disc f\{ff $\}\}\{f\}\}$
58.79 \declare@shorthand\{usorbian\}\{"F\}\{\textormath\{\bbl@disc F\{FF\}\}\{F\}\}
$58.80 \backslash$ declare@shorthand\{usorbian\}\{"l\}\{\textormath\{\bbl@disc $1\{11\}\}\{1\}\}$
58.81 \declare@shorthand\{usorbian\}\{"L\}\{\textormath\{\bbl@disc L\{LL\}\}\{L\}\}
58.82 \declare@shorthand\{usorbian\}\{"m\}\{\textormath\{\bbl@disc m\{mm\}\}\{m\}\}
58.83 \declare@shorthand\{usorbian\}\{"M\}\{\textormath\{\bbl@disc M\{MM\}\}\{M\}\}
58.84 \declare@shorthand\{usorbian\}\{"n\}\{\textormath\{\bbl@disc $n\{n n\}\}\{n\}\}$
58.85 \declare@shorthand\{usorbian\}\{"N\}\{\textormath\{\bbl@disc N\{NN\}\}\{N\}\}
58.86 \declare@shorthand\{usorbian\}\{"p\}\{\textormath\{\bbl@disc p\{pp\}\}\{p\}\}
58.87 \declare@shorthand\{usorbian\}\{"P\}\{\textormath\{\bbl@disc P\{PP\}\}\{P\}\}
58.88 \declare@shorthand\{usorbian\}\{"t\}\{\textormath\{\bbl@disc t\{tt\}\}\{t\}\}

```
58.89\declare@shorthand{usorbian}{"T}{\textormath{\bbl@disc T{TT}}{T}}
and some additional commands:
58.90\declare@shorthand{usorbian}{"-}{\nobreak\-\bbl@allowhyphens}
58.91 \declare@shorthand{usorbian}{"|}{%
58.92 \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
5 8 . 9 3 ~ \ a l l o w h y p h e n s \} \{ \} \}
58.94\declare@shorthand{usorbian}{""}{\hskip\z@skip}
```

$\backslash$ mdqon All that's left to do now is to define a couple of commands for reasons of compat$\backslash$ mdqoff ibility with german.sty.
$\backslash \mathrm{ck}_{58.95} \backslash$ def $\backslash$ mdqon $\{\backslash$ shorthandon\{" $\left.\}\right\}$
$58.96 \backslash$ def $\backslash m d q o f f\{\backslash$ shorthandoff $\{$ " $\}\}$
58.97 \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.

```
58.98 \ldf@finish{usorbian}
58.99 \/code\
```


## 59 The Turkish language

The file turkish.dtx ${ }^{67}$ defines all the language definition macros for the Turkish language ${ }^{68}$.

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 \frenhspacing 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.

```
59.1 〈*code〉
59.2\LdfInit{turkish}\captionsturkish
```

When this file is read as an option, i.e. by the command,turkishcouldbean'unknown'languageinwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\l@turkishtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
59.3\ifx\l@turkish\@undefined
59.4 \@nopatterns{Turkish}
59.5 \adddialect\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 $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$.

```
59.6 \addto\captionsturkish\{\%
59.7 \def\prefacename\{\"Ons\"oz\}\%
59.8 \def \refname\{Kaynaklar\}\%
59.9 \def \abstractname\{\"Ozet\}\%
\(59.10 \backslash d e f \backslash b i b n a m e\{K a y n a k \backslash c ~ c a\} \%\)
59.11 \def\chaptername\{B\"ol\"um\}\%
59.12 \def \appendixname\{Ek\}\%
59.13 \def \contentsname\{\.I\c cindekiler\}\%
59.14 \def\listfigurename\{\c Sekil Listesi\}\%
59.15 \def \(\backslash l i s t t a b l e n a m e\{T a b l o ~ L i s t e s i\} \% ~\)
59.16 \def\indexname\{Dizin\}\%
59.17 \def \(\backslash\) figurename\{\c Sekil\}\%
59.18 \def \(\backslash\) tablename\{Tablo\}\%
\(59.19 \backslash \operatorname{def} \backslash p a r t n a m e\{K \backslash i \quad s \backslash i ~ m\} \%\)
59.20 \def \(\backslash e n c l n a m e\{\backslash . I l i \backslash c\) sik\}\%
59.21 \def\ccname\{Di\u ger Al\i c\i lar\}\%
59.22 \def \headtoname\{Al\i c\i\}\%
59.23 \def \(\backslash\) pagename\{Sayfa\}\%
59.24 \def\subjectname\{\.Ilgili\}\%
```

[^58]```
59.25\def\seename{bkz.}%
59.26 \def\alsoname{ayr\i ca bkz.}%
59.27 \def\proofname{Kan\i t}%
59.28 \def\glossaryname{Glossary}% <-- Needs translation
59.29 }%
```

\dateturkish The macro \dateturkish redefines the command \today to produce Turkish dates.

```
59.30\def\dateturkish{%
59.31 \def\today{\number\day~\ifcase\month\or
59.32 Ocak\or \c Subat\or Mart\or Nisan\or May\i{}s\or Haziran\or
59.33 Temmuz\or A\u gustos\or Eyl\"ul\or Ekim\or Kas\i{}m\or
59.34 Aral\i{}k\fi
59.35 \space\number\year}}
```

\extrasturkish The macro \extrasturkish will perform all the extra definitions needed for the \noextrasturkish 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.
59.36 \initiate@active@char\{:\}
59.37 \initiate@active@char\{!\}
59.38 \initiate@active@char\{=\}

We specify that the turkish group of shorthands should be used.
59.39 \addto \extrasturkish\{\languageshorthands\{turkish\}\}

These characters are 'turned on' once, later their definition may vary.
59.40 \addto\extrasturkish\{\%
59.41 \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.

```
59.42 \addto\extrasturkish{\bbl@frenchspacing}
59.43\addto\noextrasturkish{\bbl@nonfrenchspacing}
```

\turkish@sh@!@ The definitions for the three active characters were made using intermediate \turkish@sh@=@ macros. These are defined now. The insertion of extra 'white space' should only \turkish@sh@:@ happen outside math mode, hence the check \ifmmode in the macros.

```
59.44 \declare@shorthand{turkish}{:}{%
59.45 \ifmmode
    5.46 \string:%
59.47 \else\relax
59.48 \ifhmode
59.49 \ifdim\lastskip>\z@
59.50 \unskip\penalty\@M\thinspace
59.51 \fi
59.52 \fi
```

```
59.53 \string:%
59.54 \fi}
59.55 \declare@shorthand{turkish}{!}{%
59.56 \ifmmode
59.57 \string!%
59.58 \else\relax
59.59 \ifhmode
59.60 \ifdim\lastskip>\z@
                    \unskip\penalty\@M\thinspace
                \fi
                \fi
                \string!%
    \i}
    declare@shorthand{turkish}{=}{%
    \ifmmode
        \string=%
    \else\relax
        \ifhmode
            \ifdim\lastskip>\z@
                    \unskip\kern\fontdimen2\font
                    \kern-1.4\fontdimen3\font
                \fi
                \i
        \string=%
        \i}
```

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.
59.78 \ldf@finish\{turkish\}
59.79 〈/code〉

## 60 The Hebrew language

The file hebrew. $\mathrm{dtx}^{69}$ 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 $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$ commands and environments. It is designed to be used with other right-to-left languages, not only with Hebrew.
hebcal.sty 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 61 on page 400).
hebrew.fdd contains Hebrew font encodings, related font definition files and hebfont package that provides Hebrew font switching commands (see Section 62 on page 408 for further details).
${ }^{\mathrm{LA}} \mathrm{T}_{\mathrm{E}} \mathrm{X} 2.09$ compatibility files are included with heb209.dtx and gives possibility to compile existing $\mathrm{L}_{\mathrm{E}} \mathrm{T} X$ 2.09 Hebrew documents with small (if any) changes (see Section 63 on page 427 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

### 60.1 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.

[^59]
### 60.2 The DOCSTRIP modules

The following modules are used in the implementation to direct DOCSTRIP in generating external files:

$$
\begin{array}{ll}
\text { driver } & \text { produce a documentation driver file } \\
\text { hebrew } & \text { produce Hebrew language support file } \\
\text { rightleft } & \text { create right-to-left support file } \\
\text { calendar } & \text { create Hebrew calendar package }
\end{array}
$$

A typical DOCSTRIP command file would then have entries like:

```
\generateFile{hebrew.ldf}{t}{\from{hebrew.dtx}{hebrew}}
```


### 60.3 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.

```
60.1 〈*hebrew>
60.2\LdfInit{hebrew}{captionshebrew}
```

When this file is read as an option, i.e., by the command,hebrewwillbean'unknown'language,inwhichcasewehavetomakeitknown.Sowecheckfortheexistenceof\1@hebrewtoseewhetherwehavetodosomethinghere.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

```
60.3 \ifx\1@hebrew\@undefined
60.4 \@nopatterns{Hebrew}%
60.5 \adddialect\l@hebrew0
60.6 \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 cnverted 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 ${ }^{A} T_{E} \mathrm{X}$ 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}
```

```
60.7\providecommand{\HeblatexEncoding}{LHE}%
60.8 \providecommand{\HeblatexEncodingFile}{lheenc}%
60.9 \newcommand{\heblatex@set@encoding} [2] {
60.10}
60.11 \AtEndOfPackage{%
60.12 \@ifpackageloaded{fontenc}{%
60.13 \@ifl@aded{def}{%
    \HeblatexEncodingFile}{\def\hebrewencoding{\HeblatexEncoding}}{}%%
    }{%
        \input{\HeblatexEncodingFile.def}%
        \def \hebrewencoding{\HeblatexEncoding}%
    }}
```

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\}
60.19 \AtEndOfPackage\{\%
$60.20 \backslash$ @ifpackageloaded\{inputenc\} $\}\{\backslash$ 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 $\backslash l e f t h y p h e n m i n ~ a n d ~ \backslash r i g h t h y p h e n m i n . ~ T h e y ~ a r e ~ s e t ~ t o ~ 2 . ~$
60.21 \providehyphenmins $\{\backslash$ CurrentOption $\}\{$ tw@ $\backslash t w @\}$
\captionshebrew The macro \captionshebrew replaces all captions used in the four standard document classes provided with $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ with their Hebrew equivalents.

[^60]```
60.29\\def\chaptername{\@ensure@R{\hebpe\hebresh\hebqof}}%
60.30\\def\appendixname{\@ensure@R{\hebnun\hebsamekh\hebpe\hebhet}}%
60.31 \def\contentsname{\@ensure@R{%
60.32\\hebtav\hebvav\hebkaf\hebfinalnun\ %
60.33\\hebayin\hebnun\hebyod\hebyod\hebnun\hebyod\hebfinalmem}}%
60.34 \def\listfigurename{\@ensure@R{%
60.35\hebresh\hebshin\hebyod\hebmem\hebtav\ %
60.36 \hebalef\hebyod\hebvav\hebresh\hebyod\hebfinalmem}}%
60.37 \def\listtablename{\@ensure@R{%
0.38\\hebresh\hebshin\hebyod\hebmem\hebtav\
60.39 \hebtet\hebbet\heblamed\hebalef\hebvav\hebtav}}%
60.40\def\indexname{\@ensure@R{\hebmem\hebpe\hebtav\hebhet}}%
60.41 \def\figurename{\@ensure@R{\hebalef\hebyod\hebvav\hebresh}}%
60.42\def\tablename{\@ensure@R{\hebtet\hebbet\heblamed\hebhe}}%
60.43\\def\partname{\@ensure@R{\hebhet\heblamed\hebqof}}%
60.44\\def\enclname{\@ensure@R{\hebresh\hebtsadi"\hebbet}}%
60.45\def\ccname{\@ensure@R{\hebhe\hebayin\hebtav \hebqof \hebyod%
60.46 \hebfinalmem}}%
60.47\def\headtoname{\@ensure@R{\hebalef \heblamed}}%
60.48\\def\pagename{\@ensure@R{\hebayin\hebmem\hebvav\hebdalet}}%
60.49 \def\psname{\@ensure@R{\hebnun.\hebbet.}}%
60.50\\def\seename{\@ensure@R{\hebresh\hebalef\hebhe}}%
60.51\\def\alsoname{\@ensure@R{\hebresh\hebalef\hebhe \hebgimel%
6 0 . 5 2 ~ \ h e b m e m e s o f \} \} \%
60.53\\def\proofname{\@ensure@R{\hebhe\hebvav\hebkaf\hebhet\hebhe}}
60.54 \def\glossaryname{\@ensure@L{Glossary}}% <-- Needs translation
60.55}
```

\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

```
60.56 \@ifclassloaded{seminar}{%
60.57\def\slidelabel{\bf \if@rl\R{\hebshin\hebqof \hebfinalpe{} \theslide}%
60.58 \else\L{Slide \theslide}%
6 0 . 5 9
    \fi}%
60.60}{}
```

Here we provide an user with translation of Gregorian dates to Hebrew. In addition, the hebcal package can be used to create Hebrew calendar dates.
\hebmonth The macro \hebmonth\{month\} produces month names in Hebrew.
60.61 \def $\backslash$ hebmonth\#1 1 \%
60.62 \ifcase\#1\or \hebyod\hebnun\hebvav\hebalef $\backslash$ hebresh \or \%
$60.63 \backslash$ hebpe $\backslash$ hebbet $\backslash$ hebresh $\backslash h e b v a v \backslash h e b a l e f \backslash h e b r e s h \backslash o r ~ \% ~$
$60.64 \backslash$ hebmem $\backslash$ hebresh $\backslash h e b f i n a l t s a d i \backslash o r ~ \% ~$
$60.65 \backslash$ hebalef $\backslash$ hebpe $\backslash$ hebresh $\backslash$ hebyod $\backslash$ heblamed $\backslash o r$ \%
$60.66 \backslash$ hebmem $\backslash$ hebalef $\backslash$ hebyod \or $\backslash$ hebyod $\backslash h e b v a v \backslash h e b n u n \backslash h e b y o d \backslash o r ~ \% ~$
60.67 \hebyod \hebvav \heblamed $\backslash$ hebyod $\backslash$ or \%
60.68 \hebalef \hebvav \hebgimel \hebvav\hebsamekh \hebtet\or \%
60.69 \hebsamekh \hebpe $\backslash$ hebtet $\backslash$ hebmem $\backslash$ hebbet $\backslash$ hebresh ${ }^{2}$ or
$60.70 \backslash$ hebalef $\backslash$ hebvav $\backslash$ hebqof $\backslash$ hebtet $\backslash$ hebvav $\backslash$ hebbet $\backslash$ hebresh $\backslash$ or $\%$

```
60.71 \hebnun\hebvav\hebbet\hebmem\hebbet\hebresh\or %
60.72 \hebdalet\hebtsadi\hebmem\hebbet\hebresh\fi}
```

\hebdate The macro \hebdate\{day\}\{month\}\{year\} translates a given Gregorian date to Hebrew.
60.73 \def $\backslash$ hebdate\#1\#2\#3\{\%
$60.74 \backslash$ beginR \beginL\number\#1\endL\ \hebbet\hebmonth\{\#2\}
60.75 \beginL\number\#3\endL\endR\}
\hebday The macro \hebday will replace \today command when in Hebrew mode. $60.76 \backslash$ def $\backslash$ hebday\{ $\backslash$ hebdate $\{\backslash$ day $\}\{$ month\} $\{\backslash$ year $\}\}$
\datehebrew The macro \datehebrew redefines the command \today to produce Gregorian dates in Hebrew. It uses the macro \hebday.

```
60.77\def\datehebrew{\let\today=\hebday}
```

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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 's restricted horizontal mode, the $\backslash \mathrm{L}\{$ latin text $\}$ and $\backslash \mathrm{R}\{h e b r e w$ text\}, or \embox\{latin text\} and \hmbox\{hebrew text\} should be used.

Hence, we use \beginR and \beginL switching commands only when not in restricted horizontal mode.

```
60.78 \addto\extrashebrew{%
60.79 \tohebrew%
60.80\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.

```
60.81 \addto\noextrashebrew{%
60.82 \fromhebrew%
60.83\ifhmode\ifinner\else\beginL\fi\fi}
```

Generally, we can switch to- and from- Hebrew by means of standard babeldefined 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 he\unsethebrew brew 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 \setrllanguage\{ r2l language name\} and \unsetrllanguage\{r2l language name\}, that defined in the rlbabel.def file.
60.84 \def \sethebrew\{ \setrllanguage\{hebrew\}\}
60.85 \def \unsethebrew\{\unsetrllanguage\{hebrew\}\}
\hebrewtext The following two commands are obsolete and work only in $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2.09$ compatibil\nohebrewtext ity mode. They are synonyms of \sethebrew and \unsethebrew defined above.

```
60.86 \if@compatibility
60.87 \let\hebrewtext=\sethebrew
60.88 \let\nohebrewtext=\unsethebrew
60.89 \fi
```

\tohebrew These two commands change only the current font encoding to- and from- He$\backslash$ fromhebrew brew 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.

```
60.90\def\tohebrew{\@torl{hebrew}}%
60.91 \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.

```
60.92\def\@@hebrew#1{\beginR{{\tohebrew#1}}\endR}
60.93\def\@hebrew{\protect\@@hebrew}
```


### 60.3.1 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, \if@gim@final 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.
60.94 \newif \if@gim@apost $\%$ whether we print apostrophes
60.95 \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 $\backslash$ Hebrewnumeral $\backslash$ Hebrewnumeralfinal on a global style option could cause a counter to print in an inconsistent mannerfor 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).
\alph Counter-printing commands are based on the above commands. The natural name \@alph for the counter-printing commands is \alph, because Hebrew numerals are the only $\backslash$ Alph 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 $\backslash \mathrm{Alph}$; we therefore define \alph to print counters as Hebrew numerals without apostrophes, and $\backslash A l p h$ to print with apostrophes. A third form, $\backslash A l p h f i n a l$, is provided to print with apostrophes and final letters, as is required for Hebrew year designators. The commands \alph and $\backslash$ Alph are defined in latex.ltx, and we only need to redefine the internal commands \@alph and \@Alph; for $\backslash A l p h f i n a l$ 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.

```
60.102 \addto\extrashebrew{%
60.103 \let\saved@alph=\@alph%
60.104 \let\saved@Alph=\@Alph%
60.105 \renewcommand*{\@alph}[1]{\protect\hebrewnumeral{\number#1}}%
60.106 \renewcommand*{\@Alph}[1]{\protect\Hebrewnumeral{\number#1}}%
60.107 \def\Alphfinal#1{\expandafter\@Alphfinal\csname c@#1\endcsname}%
60.108\providecommand*{\@Alphfinal}[1]{\protect\Hebrewnumeralfinal{\number#1}}}
60.109 \addto\noextrashebrew{%
60.110 \let\@alph=\saved@alph%
60.111 \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 $\mathrm{LA}_{\mathrm{E}} \mathrm{X}$ labels need to be redefined for appropriate cross-referencing, see below. $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ default class files specify $\backslash \mathrm{Alph}$
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
\theenumiv environment in which a label was issued, not the environment in which it is called:
\label 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 $\mathrm{IvriT}_{\mathrm{E}} \mathrm{X}$ list. We therefore redefine $\backslash$ theenumii and $\backslash$ theenumiv, so that an explicit language instruction gets written to the .aux file.

```
60.112 \renewcommand{\theenumii}
60.113 {\if@rl\protect\hebrewnumeral{\number\c@enumii}%
60.114 \else\protect\L{\protect\@@alph{\number\c@enumii}}\fi}
60.115 \renewcommand{\theenumiv}
60.116 {\if@rl\protect\Hebrewnumeral{\number\c@enumiv}%
60.117 \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 enviroment 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) lookes 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 $\cdots \backslash$ 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.

```
60.118 \def \label\#1\{\@bsphack
60.119 \if@rl
\(60.120 \backslash\) def \(\backslash\) p@enumiii\{ \(\backslash\) p@enumii) \theenumii(\}\%
60.121 \protected@write\@auxout\{\}\%
60.122 \{\string\newlabel\{\#1\}\{\{\beginR\@currentlabel\endR\}\{\thepage\}\}\}\%
60.123 \else
\(60.124 \backslash\) def \(\backslash\) p@enumiii\{ \(\backslash\) p@enumii (\theenumii) \}\%
60.125 \protected@write\@auxout\{\}\%
60.126 \{\string \(\backslash\) newlabel\{ \(\# 1\}\{\{\backslash\) beginL\@currentlabel\endL\}\{\thepage \(\}\}\} \%\)
60.127 \fi
60.128 \@esphack\}
```

NOTE: it appears that the definition of $\backslash$ 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 headingsone 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 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 $\mathrm{IATEX}_{\mathrm{E}}$ [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.


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.
$\backslash$ @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 $\backslash \mathrm{g}$ im@no@mil to typeset each three-digit block. The algorithm is recursive, but the maximum recursion depth is 4 because $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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: \@tempenta holds the value, \@tempcntb is used for calculations.

```
60.141 \newcommand*{\@hebrew@numeral} [1]
```

60.142 \{ \@hebrew\{\@tempcnta=\#1\@tempcntb=\#1 \relax
60.143 \divide\@tempentb by 1000

If we're under 1000 , call $\backslash$ gim@nomil

```
60.144 \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.

```
60.145 \else{\@gim@apostfalse\@gim@finalfalse\@hebrew@numeral\@tempcntb},%
60.146 \multiply\@tempcntb by 1000\relax
60.147 \advance\@tempcnta by -\@tempcntb\relax
60.148 \gim@nomil\@tempenta\relax
6 0 . 1 4 9 ~ \ f i
60.150 }}
```

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 $\backslash$ gim@nomil typesets an integer between 0 and 999 (for 0 it typesets nothing). The code has been modified from the old hebcal.sty (appropriate credits-Boris Lavva and Michail Rozman ?). \@tempenta 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 \@tempenta, and call on \gim@print to subtract this value and print the letter.

Initialize: \@tempcnta holds the value, there is no previous letter.
60.151 \newcommand*\{\gim@nomil\}[1] \{\@tempcnta=\#1\@gim@prevfalse

Find the hundreds digit.
60.152 \@tempentb=\@tempenta\divide\@tempentb by $100 \backslash r e l a x \%$ hundreds digit
60.153 \ifcase\@tempentb $\%$ print nothing if no hundreds
60.154 \or\gim@print\{100\}\{\hebqof\}\%
60.155 \or\gim@print\{200\}\{\hebresh\}\%
60.156 \or $\backslash$ gim@print $\{300\}\{\backslash$ hebshin\}\%
60.157 \or\gim@print\{400\}\{\hebtav\}\%
60.158 \or $\backslash$ hebtav \@gim@prevtrue $\backslash$ gim@print $\{500\}\{\backslash$ hebqof $\} \%$
60.159 \or $\backslash$ hebtav \@gim@prevtrue\gim@print $\{600\}\{\backslash$ hebresh\}\%
60.160 \or $\backslash$ hebtav \@gim@prevtrue\gim@print\{700\}\{\hebshin\}\%
60.161 \or $\backslash h e b t a v \backslash @ g i m @ p r e v t r u e \backslash g i m @ p r i n t\{800\}\{\backslash h e b t a v\} \%$
60.162 \or $\backslash$ hebtav \@gim@prevtrue $\backslash$ hebtav $\backslash$ gim@print $\{900\}\{\backslash$ hebqof $\} \%$
60.163 \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.
60.164 \@tempcntb=\@tempcnta\divide\@tempcntb by $10 \backslash$ relax $\%$ tens digit
60.165 \ifcase\@tempentb \% print nothing if no tens
60.166 lor $\%$ number between 10 and 19
60.167 \ifnum\@tempcnta = 16 \gim@print \{9\}\{\hebtet\}\% tet-zayin
60.168 \else\ifnum\@tempenta = 15 \gim@print \{9\}\{\hebtet\}\% tet-vav
60.169 \else $\backslash$ gim@print\{10\}\{\hebyod\}\%
$60.170 \quad$ fi $\%$ \@tempcnta $=15$
60.171
\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.

| 60.172 | \or\gim@print\{20\}\{\if@gim@final \hebfinalkaf \else\hebkaf \fi\}\% |
| :---: | :---: |
| 60.173 | \or\gim@print $\{30\}\{\backslash$ heblamed\}\% |
| 60.174 | \or\gim@print\{40\}\{\if@gim@final\hebfinalmem\else\hebmem\fi\}\% |
| 60.175 |  |
| 60.176 | \or\gim@print\{60\}\{\hebsamekh\}\% |
| 60.177 | \or\gim@print\{70\}\{\hebayin\}\% |
| 60.178 | \or\gim@print\{80\}\{\if@gim@final\hebfinalpe\else\hebpe\fi\}\% |
| 60.179 | \or\gim@print\{90\}\{\if@gim@final\hebfinaltsadi\else\hebtsadi\fi\}\% |
|  |  |

Print the ones digit.
60.18
\ifcase\@tempcnta
\% print nothing if no ones
\or\gim@print\{1\}\{\hebalef\}\%
\or\gim@print\{2\}\{\hebbet\}\%
\or\gim@print $\{3\}\{\backslash$ hebgimel\}\%
\or\gim@print $\{4\}\{\backslash$ hebdalet $\} \%$
\or\gim@print\{5\}\{\hebhe\}\%
\or\gim@print\{6\}\{\hebvav\}\%
\or\gim@print\{7\}\{\hebzayin\}\%

```
60.189 \or\gim@print{8}{\hebhet}%
60.190 \or\gim@print{9}{\hebtet}%
60.191 \fi
60.192 }
```

\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.

```
60.193\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 \@tempenta is zero if and only if this is the last letter.

```
60.194 \newcommand*{\gim@print}[2]{% #2 is a letter, #1 is its value.
60.195 \advance\@tempenta 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.

```
60.196 \ifnum\@tempcnta=0% if this is the last letter
60.197 \if@gim@prev\if@gim@apost"\fi#2%
60.198 \else#2\if@gim@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.

```
60.199 \else{\@gim@finalfalse#2}\@gim@prevtrue\fi}
```

\hebr The older Hebrew counter commands \hebr and \gim are retained in order to \gim keep older documents from breaking. They are set to be equivalent to $\backslash a l p h$, 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.
$60.200 \backslash$ let $\backslash$ hebr $=\backslash$ alph
60.201 \let $\backslash g i m=\backslash a l p h$

For backward compatibility with 'older' hebrew.sty packages, we define Hebrew equivalents of some useful $\mathrm{LAT}_{\mathrm{EX}}$ commands. Note, however, that 8-bit macros defined in Hebrew are no longer supported.
$60.202 \backslash$ def $\backslash$ hebcopy $\{\backslash$ protect $\backslash R\{\backslash$ hebhe $\backslash$ hebayin $\backslash$ hebtav $\backslash$ hebqof $\}\}$
$60.203 \backslash \operatorname{def} \backslash$ hebincl\{\protect $\backslash R\{\backslash$ hebresh $\backslash$ hebtsadi" $\backslash$ hebbet $\}\}$

```
60.204\def\hebpage{\protect\R{\hebayin\hebmem\hebvav\hebdalet}}
60.205\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.
60.206 \def $\backslash$ hadgesh\#1\{\leavevmode\setbox0=\hbox\{\#1\}\%
60.207 \kern-.025em \copy0\kern-\wd0
60.208 \kern.05em \copy0 ${ }^{2}$ kern-\wd0
60.209 \kern-.025em \raise.0433em \box0 \}
$\backslash$ piska and \piskapiska sometimes used in 'older' hebrew sources, and should not be used in $\mathrm{IAT}_{\mathrm{EX}} 2_{\varepsilon}$.
60.210 \if@compatibility
$60.211 \backslash$ def $\backslash$ piska\#1\{- \{\#1\}\hangindent=-\hangindent\}
\(60.212 \backslash\) def \(\backslash\) piskapiska\#1\{
- item\{\#1\}\hangindent=- \(\backslash\) hangindent \(\}\)
60.213 \fi


The following commands are simply synonyms for the standard ones, provided with T}_{\mathrm{E}}\mathrm{X}2_{\varepsilon}\).60.214\let$\backslashmakafgadol=\textendash~$60.215\let$\backslashmakafanak=\textemdash$60.216\let$\backslash$geresh=$\backslash$textquoteright60.217\let\opengeresh=’60.218\let\closegeresh=‘60.219\let$\backslashopenquote=$textquotedblright60.220\let\closequote=“60.221\let$\backslash$leftquotation=”WeneedtoensurethatHebrewisusedasthedefaultright-to-leftlanguageat\begin\{document\}.Themechanismofdefiningthe\@rllanguagenameis}thesameasinbabel's\languagename:thelastright-to-leftlanguageinthe\usepackage\{babel\}lineissetasthedefaultright-to-leftlanguageatdocumentbeginning.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

For example, the following code:
ackage[russian,hebrew,arabic,greek,english]\{babel\}willsettheArabiclanguageasthedefaultright-to-leftlanguageandtheEnglishlanguageasthedefaultlanguage.Asaresult,thecommands$\backslashL\}$and$\backslash$embox$\}$willuseEnglishand$\backslashR\}$and$\backslashhmbox\}$willuseArabicbydefault.Thesedefaultscanbechangedwiththenext\sethebrewor\selectlanguage\{languagename\}command.60.223\AtBeginDocument\{\def\@rllanguagename\{hebrew\}\}undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

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.
60.224 \ldf@finish\{hebrew\}
60.225 〈/hebrew〉

### 60.4 Right to left support

This file rlbabel. def defines necessary bidirectional macro support for $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$. 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ extensions, namely $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ primitives $\backslash$ beginL, \endL and $\backslash$ beginR, \endR, currently implemented only in $\varepsilon-\mathrm{T}_{\mathrm{E}} \mathrm{X}$ and in $\mathrm{T}_{\mathrm{E}} \mathrm{X}-\mathrm{X}_{\mathrm{E}} \mathrm{T}$.

If $\varepsilon-\mathrm{T}_{\mathrm{E}} \mathrm{X}$ is used, we should switch it to the enhanced mode:
60.226 〈*rightleft〉
60.227 \ifx $\backslash$ TeXXeTstate $\backslash$ undefined $\backslash e l s e \%$
60.228 \TeXXeTstate=1
60.229 \fi

Note, that $\varepsilon-\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 's format file should be created for extended mode. Mode can be checked by running $\varepsilon-\mathrm{T}_{\mathrm{E}} \mathrm{X}$ on some $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ :
60.230 \ifx\beginL\@undefined\%
60.231 \newlinechar'<br>~~J
60.232 \typeout\{~~JTo avoid this error message, ~~J\%
60.233 run TeX--XeT or e-TeX engine instead of regular TeX. $\left.{ }^{-\sim J}\right\}$
60.234 \errmessage\{Right-to-Left Support Error: use TeX--XeT or e-TeX
60.235 engine\}\%
60.236 \fi

### 60.4.1 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 languagedefinition files and should be used to change default language (and direction).

Local direction changing commands (for small pieces of text): $\backslash \mathrm{L}\}, \backslash \mathrm{R}\{ \}$, $\backslash e m b o x\}$ and $\backslash$ hmbox\{ $\}$ 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.
60.237 \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.

```
60.238\AtBeginDocument{% Here we set the main document direction
60.239 \newif\if@rlmain%
60.240 \if@rl% e.g: if the options to babel were [english,hebrew]
60.241 \@rlmaintrue%
60.242 \else% e.g: if the options to babel were [hebrew,english]
60.243 \@rlmainfalse%
6 0 . 2 4 4 ~ \ f i \%
60.245}
```

\@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 derection.

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.

```
60.246 \DeclareRobustCommand{\@torl}[1]{%
60.247 \if@rl\else%
60.248 \let\lr@encodingdefault=\encodingdefault%
6 0 . 2 4 9 ~ \ f i \% ~
60.250\\def\@rllanguagename{#1}%
60.251 \def\encodingdefault{\csname#1encoding\endcsname}%
60.252 \fontencoding{\encodingdefault}%
60.253 \selectfont%
60.254 \@rltrue}
```

\@fromrl Opposite to \@torl, switches current direction to Left-to-Right: restores saved Left-to-Right encoding ( $\backslash$ lr@encodingdefault) and changes direction.

```
60.255 \DeclareRobustCommand{\@fromrl}{%
60.256 \if@rl%
60.257 \let\encodingdefault=\lr@encodingdefault%
6 0 . 2 5 8 ~ \ f i \%
60.259 \fontencoding{\encodingdefault}%
60.260 \selectfont%
60.261 \@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.
60.262 \expandafter $\backslash$ def $\backslash c s n a m e ~ s e l e c t l a n g u a g e ~ \ e n d c s n a m e \# 1\{\% ~$
60.263 \edef $\backslash$ languagename $\{\%$
$60.264 \backslash$ ifnum $\backslash$ escapechar=\expandafter'\string\#1\@empty
\@@selectlanguage This new internal macro redefines a final part of the standard babel's \selectlanguage implementation.

Standard $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ provides us with 3 tables: Table of Contents (.toc), List of Figures (.lof), and List of Tables (.lot). In multi-lingual texts mixing Left-toRight 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).

```
60.267 \def\@@selectlanguage#1{%
60.268 \select@language{#1}%
60.269 \if@filesw
60.270 \protected@write\@auxout{}{\string\select@language{#1}}%
60.271 \if@rl%
60.272 \addtocontents{cot}{\xstring\select@language{#1}}%
60.273 \addtocontents{fol}{\xstring\select@language{#1}}%
60.274 \addtocontents{tol}{\xstring\select@language{#1}}%
60.275 \else%
60.276 \addtocontents{toc}{\xstring\select@language{#1}}%
60.277 \addtocontents{lof}{\xstring\select@language{#1}}%
60.278 \addtocontents{lot}{\xstring\select@language{#1}}%
60.279 \fi%
60.280 \fi}
```

\setrllanguage The \setrllanguage and \unsetrllanguage pair of macros is proved to very \unsetrllanguage useful in bilingual texts, for example, in Hebrew-English texts. The languagespecific commands, for example, \sethebrew and \unsethebrew use these macros as basis.

Implementation saves and restores other language in \other@languagename variable, and uses internal macro \@@selectlanguage, defined above, to switch between languages.

```
60.281 \let\other@languagename=\languagename
60.282\DeclareRobustCommand{\setrllanguage}[1] {%
6 0 . 2 8 3 ~ \ i f @ r l \ e l s e \% ~
60.284 \let\other@languagename=\languagename%
60.285 \fi%
60.286 \def\languagename{#1}%
60.287 \@@selectlanguage{\languagename}}
60.288 \DeclareRobustCommand{\unsetrllanguage} [1] {%
6 0 . 2 8 9 ~ \ i f @ r l \% ~
60.290 \let\languagename=\other@languagename%
6 0 . 2 9 1 ~ \ f i ~
60.292 \@@selectlanguage{\languagename}}
```

$\backslash \mathrm{L}$ Macros for changing direction, originally taken from TUGboat. Usage: \L\{Left to
$\backslash \mathrm{R}$ Right text\} and $\backslash$ R\{Right to Left text $\}$. Numbers should also be enclosed in $\backslash \mathrm{L}\}$, $\backslash H e b l a t e x R e d e f i n e L$ as in $\backslash L\{123\}$.

Note, that these macros do not receive language name as parameter. Instead, the saved \@rllanguagename will be used. We assume that each Right-to-Left language defines \tolanguagename and \fromlanguagename macros in language definition file, for example, for Hebrew: \tohebrew and \fromhebrew macros in hebrew.ldf file.

The macros $\backslash \mathrm{L}$ and $\backslash \mathrm{R}$ include 'protect' to to make them robust and allow use, for example, in tables.

Due to the fact that some packages have different definitions for $\backslash \mathrm{L}$ the macro $\backslash$ HeblatexRedefineL is provided to overide them. This may be required with hyperref, for instance.

```
60.293 \let\next=\
60.294 \def\HeblatexRedefineL{%
60.295\def\L{\protect\pL}%
60.296 }
60.297 \HeblatexRedefineL
60.298\def\pL{\protect\afterassignment\moreL \let\next= }
60.299\def\moreL{\bracetext \aftergroup\endL \beginL\csname
60.300 from\@rllanguagename\endcsname}
60.301 \def \R{\protect\pR}
60.302\def\pR{\protect\afterassignment\moreR \let\next= }
60.303\def\moreR{\bracetext \aftergroup\endR \beginR\csname
60.304 to\@rllanguagename\endcsname}
60.305\def\bracetext{\ifcat\next{\else\ifcat\next}\fi
60.306 \errmessage{Missing left brace has been substituted}\fi \bgroup}
60.307 \everydisplay{\if@rl\aftergroup\beginR\fi }
```

\@ensure@R Two small internal macros, a-la \ensuremath
\@ensure@fo.308 \def \@ensure@R\#1\{\if@rl\#1\else\R\{\#1\}\fi\}

```
60.309\def\@ensure@L#1{\if@rl\L{#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 implementation was rewritten by Alon Ziv using an idea by Chris Rowley: \noindent now works unmodified.

```
60.310\def\rl@everypar{\if@rl{\setbox\z@\lastbox\beginR\usebox\z@}\fi}
```

60.311 \let\o@everypar=\everypar
$60.312 \backslash$ def \everypar\#1\{\o@everypar\{\rl@everypar\#1\}\}
$\backslash$ hmbox Useful vbox commands. All text in math formulas is best enclosed in these: LR \embox text in $\backslash$ embox and RL text in $\backslash$ hmbox. $\backslash \operatorname{mbox}\}$ is useless for both cases, since it typesets in Left-to-Right even for Right-to-Left languages (additions by Yaniv Bargury).
60.313 \newcommand $\{\backslash$ hmbox $\}[1]\{\backslash \operatorname{mbox}\{\backslash R\{\# 1\}\}\}$
60.314 \newcommand\{\embox\} [1] \{\mbox $\{\backslash L\{\# 1\}\}\}$
\@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.
60.315 \def\@brackets\#1\#2\#3\{\protect\if@rl \#3\#2\#1\protect\else
60.316 \#1\#2\#3\protect\fi\}
\@number \@number preserves numbers direction from Left to Right. \@latin in addition \@latin switches current encoding to the latin.

```
60.317\def\@@number#1{\ifmmode\else\beginL\fi#1\ifmmode\else\endL\fi}
60.318\def\@@latin#1{\@@number{{\@fromrl#1}}}
60.319 \def\@number{\protect\@@number}
60.320\def\@latin{\protect\@@latin}
```


### 60.4.2 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 languagespecific counters, such as \hebr or \gim are provided with language definition file.

We start with saving the original definitions:

```
60.321 \let\@@arabic=\@arabic
60.322 \let\@@roman=\@roman
60.323\let\@@Roman=\@Roman
60.324\let\@@alph=\@alph
60.325\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.

60.327 \def \@roman\#1\{\@latin\{\@@roman\#1\}\} 60.328 \def \@Roman\#1\{\@latin\{\@@Roman\#1\}\}
\arabicnorl This macro preserves the original definition of \arabic (overrides the overriding of \@arabic)
$60.329 \backslash$ def \arabicnorl\#1\{\expandafter\@@arabic\csname c@\#1\endcsname\}
$\backslash$ make@lr In Right to Left documents all counters defined in the standard document classes article, report and book provided with $\mathrm{HAT}_{\mathrm{E}} 2_{\varepsilon}$, such as $\backslash$ thesection, \thefigure, \theequation should be typed as numbers from left to right. To ensure direction, we use the following \make@lr\{counter\} macro:

```
60.330 \def\make@lr#1{\begingroup
    60.331 \toks@=\expandafter{#1}%
    60.332 \edef\x{\endgroup
    60.333\def\noexpand#1{\noexpand\@number{\the\toks@}}}%
    60.334 \x}
    60.335\@ifclassloaded{letter}{}{%
    60.336 \@ifclassloaded{slides}{}{%
    60.337 \make@lr\thesection
```

```
60.338 \make@lr\thesubsection
60.339 \make@lr\thesubsubsection
60.340 \make@lr\theparagraph
60.341 \make@lr\thesubparagraph
60.342 \make@lr\thefigure
60.343 \make@lr0ble
60.344 }
60.345 \make@lr\theequation
60.346 }
```


### 60.4.3 Preserving logos


$\backslash \mathrm{TeX}$
60.347 \let $\backslash @$ @TeX $\backslash \mathrm{TeX}$
$60.348 \backslash \operatorname{def} \backslash \mathrm{TeX}\{\backslash @ 1$ atin\{ \@@TeX\}\}
$\backslash$ LaTeX
60.349 \let $\backslash @$ LaTeX $\backslash \mathrm{LaTeX}$
60.350 \def $\backslash$ LaTeX\{\@latin\{ $\backslash @$ @aTeX $\}\}$
\LaTeXe

```
60.351 \let\@@LaTeXe\LaTeXe
60.352 \def\LaTeXe{\@latin{\@@LaTeXe}}
```


### 60.4.4 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. $\mathrm{HT}_{\mathrm{E}} \mathrm{X}$ 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.

```
60.353 \def\list#1#2{%
60.354 \ifnum \@listdepth >5\relax
6 0 . 3 5 5 ~ \ @ t o o d e e p
60.356 \else
60.357 \global\advance\@listdepth\@ne
60.358 \fi
60.359 \rightmargin\z@
60.360 \listparindent\z@
60.361 \itemindent\z@
60.362 \csname @list\romannumeral\the\@listdepth\endcsname
60.363 \def\@itemlabel{#1}%
60.364 \let\makelabel\@mklab
60.365 \@nmbrlistfalse
60.366 #2\relax
```

```
60.367 \@trivlist
60.368 \parskip\parsep
60.369 \parindent\listparindent
60.370 \advance\linewidth -\rightmargin
60.371 \advance\linewidth -\leftmargin
```

The only change in the macro is the \if@rl case

```
60.372 \if@rl
60.373 \advance\@totalleftmargin \rightmargin
60.374 \else
60.375 \advance\@totalleftmargin \leftmargin
60.376 \fi
60.377 \parshape \@ne \@totalleftmargin \linewidth
60.378 \ignorespaces}
```

\labelenumii The \labelenumii and \p@enumiii commands use parentheses. They are revised $\backslash$ @@enumiii to work Right-to-Left with the help of \@brackets macro defined above.
60.379 \def $\backslash$ labelenumii $\{$ @brackets ( $\backslash$ theenumii) \}
$60.380 \backslash$ def $\backslash p @ e n u m i i i\{\backslash p @ e n u m i i \backslash @ b r a c k e t s(\backslash t h e e n u m i i)\}$

### 60.4.5 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 $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$ 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.
\@tableof contents We define 3 new macros similar to the standard $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$ tables, but with one pa\@listoffigures rameter - table file extension. These macros will help us to define our additional \@listoftables tables below.
60.381 \@ifclassloaded\{letter\}\{\}\{\% other
60.382 \@ifclassloaded\{slides\}\{\}\{\% other
60.383 \@ifclassloaded\{article\}\{\% article
60.384 \newcommand $\backslash$ @tableof contents [1] \{\%
60.385 \section $*\{$ \contentsname $\backslash$ @mkboth\%
$60.386 \quad\{\backslash$ MakeUppercase $\backslash$ contentsname \}\%
$60.387 \quad\{\backslash$ MakeUppercase $\backslash$ contentsname $\}\} \%$
60.388 \@starttoc\{\#1\}\}
60.389 \newcommand \@listoffigures [1] \{\%
60.390 \section*\{\listfigurename\@mkboth\%
$60.391 \quad\{\backslash$ MakeUppercase $\backslash$ listfigurename\}\%
$60.392 \quad\{\backslash$ MakeUppercase $\backslash$ listfigurename $\}$ \}
60.393 \@starttoc\{\#1\}\}
60.394 \newcommand \@listoftables[1] \{\%
60.395 \section*\{\listtablename\@mkboth\%
$60.396 \quad\{\backslash$ MakeUppercase $\backslash$ listtablename $\} \%$
$60.397 \quad\{\backslash$ MakeUppercase $\backslash$ listtablename $\}\} \%$

```
60.398 \@starttoc{#1}}}%
6 0 . 3 9 9 ~ \{ \% ~ e l s e ~ r e p o r t ~ o r ~ b o o k
60.400 \newcommand\@tableofcontents[1]{%
60.401 \@restonecolfalse\if@twocolumn\@restonecoltrue\onecolumn%
60.402 \fi\chapter*{\contentsname\@mkboth%
60.403 {\MakeUppercase\contentsname}%
60.404 {\MakeUppercase\contentsname}}%
60.405 \@starttoc{#1}\if@restonecol\twocolumn\fi}
60.406 \newcommand\@listoffigures[1]{%
60.407 \@restonecolfalse\if@twocolumn\@restonecoltrue\onecolumn%
60.408 \fi\chapter*{\listfigurename\@mkboth%
60.409 {\MakeUppercase\listfigurename}%
60.410 {\MakeUppercase\listfigurename}}%
60.411 \@starttoc{#1}\if@restonecol\twocolumn\fi}
60.412 \newcommand\@listoftables[1]{%
60.413 \if@twocolumn\@restonecoltrue\onecolumn\else\@restonecolfalse\fi%
60.414 \chapter*{\listtablename\@mkboth%
60.415 {\MakeUppercase\listtablename}%
60.416 {\MakeUppercase\listtablename}}%
60.417
```

```
\lrtableof contents Left-to-Right tables are called now \lr}xxx\mathrm{ and defined with the aid of three macros
    \lrlistoffigures defined above (extensions toc, lof, and lot).
    \lrlistoftables0.418 \newcommand\lrtableofcontents{\@tableofcontents{toc}}%
    60.419 \newcommand\lrlistoffigures{\@listoffigures{lof}}%
    60.420 \newcommand\lrlistoftables{\@listoftables{lot}}%
```

$\backslash r l t a b l e o f$ contents Right-to-Left tables will be called $\backslash r l x x x$ and defined with the aid of three macros

\rllistoftableso.421 \newcommand \rltableof contents\{\@tableof contents\{cot\}\}\%
60.422 \newcommand $\backslash r l l i s t o f f i g u r e s\{\backslash @ l i s t o f f i g u r e s\{f o l\}\} \%$
60.423 \newcommand $\backslash r l l i s t o f t a b l e s\{\backslash @ 1 i s t o f t a b l e s\{t o l\}\} \%$
$\backslash$ tableof contents Let $\backslash x x x$ be $\backslash r l x x x$ if the current direction is Right-to-Left and $\backslash \operatorname{lr} x x x$ if it is
\listoffigures Left-to-Right.
\listoftableso. 424 }
$60.426 \backslash$ renewcommand $\backslash$ listoffigures $\{\backslash$ if@rl\rllistoffigures\%
$\backslash e l s e \backslash l r l i s t o f f i g u r e s \backslash f i\}$

$60.429 \quad$ \else\lrlistoftables $\backslash f i\}\}\}$
\@dottedtocline The following makes problems when making a Right-to-Left tables, since it uses $\backslash$ leftskip and $\backslash$ rightskip which are both mode dependent.

```
60.430 \def \@dottedtocline\#1\#2\#3\#4\#5\{\%
60.431 \ifnum \#1>\c@tocdepth \else
60.432 \vskip \z@ \@plus.2\p@
\(60.433 \quad\{\backslash i f @ r l \backslash r i g h t s k i p \backslash e l s e \backslash l e f t s k i p \backslash f i ~ \# 2 \backslash r e l a x\)
\(60.434 \backslash i f @ r l \backslash l e f t s k i p \backslash e l s e \backslash r i g h t s k i p \backslash f i ~ \ @ t o c r m a r g ~ \ p a r f i l l s k i p\)
```

```
60.435 -\if@rl\leftskip\else\rightskip\fi
60.436 \parindent #2\relax\@afterindenttrue
60.437 \interlinepenalty\@M
60.438 \leavevmode
60.439 \@tempdima #3\relax
60.440 \advance\if@rl\rightskip\else\leftskip\fi \@tempdima
60.441 \null\nobreak\hskip -\if@rl\rightskip\else\leftskip\fi
60.442 {#4}\nobreak
60.443 \leaders\hbox{$\m@th
60.444 \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
60.445 mu$}\hfill
60.446 \nobreak
60.447 \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor \beginL#5\endL}%
60.448 \par}%
60.449 \fi}
```

\1@part This standard macro was redefined for table of contents since it uses \rightskip which is mode dependent.

```
60.450 \@ifclassloaded{letter}{}{% other
60.451 \@ifclassloaded{slides}{}{% other
60.452 \renewcommand*\l@part [2] {%
60.453 \ifnum \c@tocdepth >-2\relax
60.454 \addpenalty{-\@highpenalty}%
60.455 \addvspace{2.25em \@plus\p@}%
60.456 \begingroup
60.457 \setlength\@tempdima{3em}%
60.458 \parindent \z@ \if@rl\leftskip\else\rightskip\fi \@pnumwidth
60.459 \parfillskip -\@pnumwidth
60.460 {\leavevmode
60.461 \large \bfseries #1\hfil \hb@xt@\@pnumwidth{\hss#2}}\par
60.462 \nobreak
60.463 \global\@nobreaktrue
60.464 \everypar{\global\@nobreakfalse\everypar{}}%
60.465 \endgroup
60.466 \fi}}}
```

\@part Part is redefined to support new Right-to-Left table of contents (cot) as well as the Left-to-Right one (toc).

```
60.467 \@ifclassloaded{article}{% article class
60.468 \def\@part[#1]#2{%
60.469 \ifnum \c@secnumdepth >\m@ne
60.470 \refstepcounter{part}%
60.471 \addcontentsline{toc}{part}{\thepart\hspace{1em}#1}%
60.472 \addcontentsline{cot}{part}{\thepart\hspace{1em}#1}%
60.473 \else
60.474 \addcontentsline{toc}{part}{#1}%
60.475 \addcontentsline{cot}{part}{#1}%
60.476 \fi
60.477 {\parindent \z@ \raggedright
```

```
60.478 \interlinepenalty \@M
60.479 \normalfont
60.480 \ifnum \c@secnumdepth >\m@ne
60.481 \Large\bfseries \partname~\thepart
60.482 \par\nobreak
60.483 \fi
60.484 \huge \bfseries #2%
60.485 \markboth{}{}\par}%
60.486 \nobreak
60.487 \vskip 3ex
60.488 \@afterheading}%
60.489 }{% report and book classes
60.490\def\@part[#1]#2{%
60.491 \ifnum \c@secnumdepth >-2\relax
60.492 \refstepcounter{part}%
60.493 \addcontentsline{toc}{part}{\thepart\hspace{1em}#1}%
60.494 \addcontentsline{cot}{part}{\thepart\hspace{1em}#1}%
60.495 \else
60.496 \addcontentsline{toc}{part}{#1}%
60.497 \addcontentsline{cot}{part}{#1}%
60.498 \fi
60.499 \markboth{}{}%
60.500 {\centering
60.501 \interlinepenalty \@M
60.502 \normalfont
60.503 \ifnum \c@secnumdepth >-2\relax
60.504 \huge\bfseries \partname~\thepart
60.505 \par
60.506 \vskip 20\p@
60.507 \fi
60.508 \Huge \bfseries #2\par}%
60.509 \@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.

```
60.510 \def \@sect#1#2#3#4#5#6[#7] #8{%
60.511 \ifnum #2>\c@secnumdepth
60.512 \let\@svsec\@empty
60.513 \else
60.514 \refstepcounter{#1}%
60.515 \protected@edef\@svsec{\@seccntformat{#1}\relax}%
60.516 \fi
60.517 \@tempskipa #5\relax
60.518 \ifdim \@tempskipa>\z@
6 0 . 5 1 9 ~ \ b e g i n g r o u p
60.520 #6{%
60.521 \@hangfrom{\hskip #3\relax\@svsec}%
60.522 \interlinepenalty \@M #8\@@par}%
6 0 . 5 2 3 ~ \ e n d g r o u p ~
60.524 \csname #1mark\endcsname{#7}%
```

```
60.525 \addcontentsline{toc}{#1}{%
60.526 \ifnum #2>\c@secnumdepth \else
60.527 \protect\numberline{\csname the#1\endcsname}%
60.528 \fi
60.529 #7}%
60.530 \addcontentsline{cot}{#1}{%
60.531 \ifnum #2>\c@secnumdepth \else
60.532 \protect\numberline{\csname the#1\endcsname}%
60.533 \fi
60.534 #7}%
60.535 \else
60.536 \def\@svsechd{%
60.537 #6{\hskip #3\relax
60.538 \@svsec #8}%
60.539 \csname #1mark\endcsname{#7}%
60.540 \addcontentsline{toc}{#1}{%
60.541 \ifnum #2>\c@secnumdepth \else
60.542 \protect\numberline{\csname the#1\endcsname}%
60.543 \fi
60.544 #7}%
60.545 \addcontentsline{cot}{#1}{%
60.546 \ifnum #2>\c@secnumdepth \else
60.547 \protect\numberline{\csname the#1\endcsname}%
60.548 \fi
60.549 #7}}%
60.550 \fi
60.551 \@xsect{#5}}
```

\@caption Caption was redefined from the latex.ltx file. It is changed to support Left-toRight list of figures and list of tables (lof and lot) as well as new Right-to-Left lists (fol and tol) simultaneously.
60.552 \long\def \@caption\#1[\#2] \#3\{\%
60.553 \par
60.554 \addcontentsline\{\csname ext@\#1 \endcsname\}\{\#1\}\%

60.556 \{\ignorespaces \#2\}\}\%
60.557 \def \@fignm\{figure\}
60.558 \ifx\#1\@fignm\addcontentsline\{fol\}\{\#1\}\%
$60.559 \quad\{\backslash$ protect $\backslash$ numberline\{ $\backslash$ csname the\#1 \endcsname\}\%
$60.560 \quad\{\backslash i g n o r e s p a c e s ~ \# 2\}\} \backslash f i \%$
60.561 \def \@tblnm\{table\}
60.562 \ifx\#1 \@tblnm\addcontentsline\{tol\}\{\#1\}\%
60.563 \{\protect $\backslash$ numberline\{
$60.564 \quad\{\backslash i g n o r e s p a c e s$ \#2\}\}\fi\%
60.565 \begingroup
60.566 \@parboxrestore
60.567 \if@minipage
60.568 \@setminipage
60.569 \fi
60.570 \normalsize

```
60.571 \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
60.572 \endgroup}
```

\1@chapter This standard macro was redefined for table of contents since it uses \rightskip which is mode dependent.
60.573 \@ifclassloaded\{letter\}\{\}\{\%
60.574 \@ifclassloaded\{slides\}\{\}\{\%
60.575 \@ifclassloaded\{article\}\{\}\{\%
60.576 \}\}\}
\1@section The toc entry for section did not work in article style. Also it does not print dots, $\backslash 1 @$ subsection which is funny when most of your work is divided into sections.

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@subparagrap $\boldsymbol{H}_{0.591 ~ \ @ i f c l a s s l o a d e d\{a r t i c l e\}\{\% ~}^{\text {( }}$
60.592
60.593
$60.594 \backslash$ renewcommand* $\backslash 1 @$ subsubsection\{\@dottedtocline\{3\}\{7.0em\}\{4.1em\}\}
60.595
60.596 \}\{\}

### 60.4.6 Two-column mode

This is the support of twocolumn option for the standard $\mathrm{IAT}_{\mathrm{E}} 2_{\varepsilon}$ classes. The following code was originally borrowed from the $\mathrm{ArabT}_{\mathrm{E}} \mathrm{X}$ 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]

```
60.597\let\@@outputdblcol\@outputdblcol
60.598%\def\set@outputdblcol{%
```

```
60.599 % \if@rl\renewcommand{\@outputdblcol}{\rl@outputdblcol}%
60.600% \else\renewcommand{\@outputdblcol}{\@@outputdblcol}\fi}
60.601 \renewcommand{\@outputdblcol}{%
60.602 \if@rlmain%
60.603 \rl@outputdblcol%
60.604 \else%
60.605 \@@outputdblcol%
60.606 \fi%
60.607 }
60.608 \newcommand{\rl@outputdblcol}{%
60.609 \if@firstcolumn
60.610 \global \@firstcolumnfalse
60.611 \global \setbox\@leftcolumn \box\@outputbox
6.612 \else
60.613 \global \@firstcolumntrue
60.614 \setbox\@outputbox \vbox {\hb@xt@\textwidth {%
6 0 . 6 1 5 ~ \ h s k i p \ c o l u m n w i d t h \% ~
60.616 \hfil\vrule\@width\columnseprule\hfil
6.617 \hb@xt@\columnwidth {%
60.618 \box\@leftcolumn \hss}%
6 0 . 6 1 9 ~ \ h b @ x t @ \ c o l u m n w i d t h ~ \{ \% ~
6 0 . 6 2 0 ~ \ h s k i p - \ t e x t w i d t h \% ~
6 0 . 6 2 1 ~ \ b o x \ @ o u t p u t b o x ~ \ h s s \} \%
60.622 \hskip\columnsep%
6 0 . 6 2 3 ~ \ h s k i p \ c o l u m n w i d t h \} \} \% \%
60.624 \@combinedblfloats
6 0 . 6 2 5 ~ \ @ o u t p u t p a g e
6 0 . 6 2 6 ~ \ b e g i n g r o u p
60.627 \@dblfloatplacement
60.628 \@startdblcolumn
60.629 \@whilesw\if@fcolmade \fi
60.630 {\@outputpage
6 0 . 6 3 1 ~ \ @ s t a r t d b l c o l u m n \} \%
60.632 \endgroup
60.633 \fi}
```


### 60.4.7 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
60.634 \let $\backslash @$ @ootnoterule=\footnoterule

60.636 \else\@@footnoterule\fi\}

### 60.4.8 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@myheading80.637 \@ifclassloaded\{letter\}\{\%

```
headeveæ0.638 \def\headodd{\protect\if@rl\beginR\fi\headtoname{}
headodd0.639 \ignorespaces\toname
60.640 \hfil \@date
60.641 \hfil \pagename{} \thepage\protect\if@rl\endR\fi}
60.642 \if@twoside
60.643 \def\ps@headings{%
60.644 \let\@oddfoot\@empty\let\@evenfoot\@empty
60.645 \def\@oddhead{\select@language{\headlanguage}\headodd}
60.646 \let\@evenhead\@oddhead}
60.647 \else
60.648 \def\ps@headings{%
6 0 . 6 4 9 ~ \ l e t \ @ o d d f o o t \ @ e m p t y ~
60.650 \def\@oddhead{\select@language{\headlanguage}\headodd}}
60.651 \fi
60.652 \def\headfirst{\protect\if@rl\beginR\fi\fromlocation \hfill %
60.653 \telephonenum\protect\if@rl\endR\fi}
60.654 \def\ps@firstpage{%
60.655 \let\@oddhead\@empty
60.656 \def\@oddfoot{\raisebox{-45\p@}[\z@]{%
60.657 \hb@xt@\textwidth{\hspace*{100\p@}%
60.658 \ifcase \@ptsize\relax
6 0 . 6 5 9 ~ \ n o r m a l s i z e
6 0 . 6 6 0 ~ \ o r
60.661 \small
60.662 \or
6 0 . 6 6 3 ~ \ f o o t n o t e s i z e
```

60.664 \f
60.665 \select@language\{\headlanguage\}\headfirst\}\}\hss\}\}
60.666 \%
60.667  [1] \{\%
60.668 \let $\backslash$ headlanguage=\languagename\%
60.669 \ifx $\backslash$ @empty $\backslash$ fromaddress\%
60.670 \thispagestyle\{firstpage\}\%
$60.671 \quad\{\backslash r a g g e d l e f t \backslash @ d a t e \backslash p a r\} \%$
60.672 \else \% home address
60.673 \thispagestyle\{empty\}\%
$60.674 \quad\{\backslash r a g g e d l e f t$
60.675 \if@rl\begin\{tabular\}\{@\{\beginR\csname\% }
60.676 to\@rllanguagename\endcsname\}r@\{\endR\}\}\ignorespaces
60.677 \fromaddress <br>*[2\parskip]\%
60.678 \@date \end\{tabular\} } \backslash par\%
60.679
60.680
60.681
60.682
60.683
60.684
60.686 \vspace\{2\parskip\}\%
60.687 60.688 \}

```
60.685 \{ \(\backslash\) raggedright \(\backslash\) toname \(\backslash \backslash\) \toaddress \(\backslash\) par\}\%
\else\begin{tabular}{1}\ignorespaces
    \fromaddress \\*[2\parskip]%
    \@date \end{tabular}\par%
        \fi}%
    #1\par\nobreak}
```

    \fi
    Then, the article, report and book document classes are supported. Note, that in one-sided mode \markright was changed to \markboth.
60.689 \{\% article, report, book
$60.690 \backslash$ def $\backslash$ headeven $\backslash$ protect $\backslash i f @ r l \backslash$ beginR $\backslash$ thepage $\backslash h f i l \backslash r i g h t m a r k \backslash e n d R ~$
\protect\else\thepage\hfil\{\slshape\leftmark\} \protect\fi\}
60.692
60.693
60.694
60.695
60.696
60.697 60.698
60.699
60.700
60.701
60.702
60.703 60.704
60.705
60.706 60.707
60.708
60.709
60.710
60.711
60.712
60.713
60.71
60.715 $\quad$ \% one-sided
$60.716 \backslash \operatorname{def} \backslash \mathrm{ps@headings}\{\%$
60.717 \let \@oddfoot \@empty
60.718 \def \@oddhead\{\headodd\}\%
60.719 \let \@mkboth $\backslash$ markboth
$60.720 \backslash$ def \sectionmark\#\#1\{\%
$60.721 \backslash$ markboth\{\MakeUppercase\{\%
60.722 \ifnum \c@secnumdepth >\m@ne
60.723 \thesection\quad
60.724 \fi
60.725 \#\#1\}\}\{\MakeUppercase\{\%

```
        \ifnum \c@secnumdepth >\m@ne
                        \thesection\quad
        \i
        ##1}}}}
\fi
\def\ps@myheadings{%
        \let\@oddfoot\@empty\let\@evenfoot\@empty
        \def\@evenhead{\select@language{\headlanguage}\headeven}%
        \def\@oddhead{\select@language{\headlanguage}\headodd}%
        \let\@mkboth\@gobbletwo
        \let\sectionmark\@gobble
        \let\subsectionmark\@gobble
}}{% report and book
    \if@twoside % two-sided
        \def\ps@headings{%
            \let\@oddfoot\@empty\let\@evenfoot\@empty
            \def\@evenhead{\select@language{\headlanguage}\headeven}
            \def\@oddhead{\select@language{\headlanguage}\headodd}
            \let\@mkboth\markboth
            \def\chaptermark##1{%
                \markboth{\MakeUppercase{%
                    \ifnum \c@secnumdepth >\m@ne
                            \@chapapp\ \thechapter. \ %
                    \fi
                    ##1}}{}}%
            \def\sectionmark##1{%
                \markright {\MakeUppercase{%
                    \ifnum \c@secnumdepth >\z@
                            \thesection. \ %
                    \fi
                    ##1}}}}
\else % one-sided
        \def\ps@headings{%
            \let\@oddfoot\@empty
            \def\@oddhead{\select@language{\headlanguage}\headodd}
            \let\@mkboth\markboth
            \def\chaptermark##1{%
                \markboth{\MakeUppercase{%
                    \ifnum \c@secnumdepth >\m@ne
                            \@chapapp\\thechapter. \ %
                            \fi
                            ##1}}{\MakeUppercase{%
                            \ifnum \c@secnumdepth >\m@ne
                            \@chapapp\\thechapter.\ %
                    \fi
                    ##1}}}}
\i
\def\ps@myheadings{%
    \let\@oddfoot\@empty\let\@evenfoot\@empty
```

```
60.776\\def\@evenhead{\select@language{\headlanguage}\headeven}%
60.777 \def\@oddhead{\select@language{\headlanguage}\headodd}%
60.778 \let\@mkboth\@gobbletwo
60.779 \let\chaptermark\@gobble
60.780 \let\sectionmark\@gobble
60.781 }}}
```


### 60.4.9 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 interpeted after the document has been converted by $\mathrm{T}_{\mathrm{E}} \mathrm{Xto}$ 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 interpeted by $\mathrm{T}_{\mathrm{E}} \mathrm{Xto}$ something like:
[postscript:make color cyan]some LTR text[postscript:make color black]
However, with the bidirectionality support we get:
\textcolor\{cyan\}\{\hebalef $\backslash$ 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

```
60.782\AtBeginDocument{%
60.783 %I assume that \@textcolor is only defined by the package color
60.784 \ifx\@textcolor\@undefined\else%
60.785 % If that macro was defined before the beginning of the document,
60.786 % that is: the package was loaded: redefine it with bidi support
60.787 \def\@textcolor#1#2#3{%
6 0 . 7 8 8 ~ \ i f @ r l \% ~
60.789 \beginL\protect\leavevmode{\color#1{#2}\beginR#3\endR}\endL%
60.790 \else%
60.791 \protect\leavevmode{\color#1{#2}#3}%
60.792 \fi%
60.793 }%
60.794 \fi%
6 0 . 7 9 5 \}
60.796 % \end{macrocode}
60.797% \end{macro}
60.798% \begin{macro}{\thetrueSlideCounter}
60.799% This macro probably needs to be overriden for when using lprosperl,
60.800 % (waiting for feedback. Tzafrir)
60.801 % \begin{macrocode}
60.802 \@ifclassloaded{prosper}{%
```

```
60.803 \def\thetrueSlideCounter{\arabicnorl{trueSlideCounter}}
```

60.804 \}\{\}

### 60.4.10 Miscellaneous internal $\mathrm{EA}_{\mathbf{E}} \mathrm{X}$ macros

\raggedright \raggedright was changed from latex.ltx file to support Right-to-Left mode, $\backslash$ raggedleft because of the bug in its implementation.

```
60.805\def\raggedright{%
60.806 \let\\\@centercr
60.807 \leftskip\z@skip\rightskip\@flushglue
60.808 \parindent\z@\parfillskip\z@skip}
```

Swap meanings of \raggedright and \raggedleft in Right-to-Left mode.
60.809 \let \@@raggedleft=\raggedleft
60.810 \let\@@raggedright=\raggedright
60.811 }\)
60.813 }\)
\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.

```
60.815 \let\@@author=\author
60.816 \renewcommand{\author}[1]{\@@author{\if@rl\beginR #1\endR\else #1\fi}}
```

$\backslash$ MakeUppercase There are no uppercase and lowercase letters in most Right-to-Left languages, $\backslash$ MakeLowercase therefore we should redefine $\backslash$ MakeUppercase and $\backslash$ MakeLowercase IATEX $2 \varepsilon$ commands.
60.817 \let \@@MakeUppercase=\MakeUppercase
60.818 \def $\backslash$ MakeUppercase\#1\{\if@rl\#1\else\@@MakeUppercase\{\#1\}\fi\}
60.819 \let \@@MakeLowercase=\MakeLowercase
$60.820 \backslash$ def $\backslash$ MakeLowercase\#1\{\if@rl\#1\else\@@MakeLowercase\{\#1\}\fi\}
\underline We should explicitly use $\backslash \mathrm{L}$ and $\backslash \mathrm{R}$ commands in \underlined text.
60.821 \let \@@@underline=\underline
60.822 \def \underline\#1\{\@@@underline\{\if@rl\R\{\#1\}\else \#1\fi\}\}
\undertext was added for $\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2.09$ compatibility mode.
60.823 \if@compatibility
60.824 \let\undertext=\underline
60.825 \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.

```
60.826 \def \@xnthm#1#2[#3] {%
    60.827 \expandafter\@ifdefinable\csname #1\endcsname
```

```
60.828 {\@definecounter{#1}\@addtoreset{#1}{#3}%
60.829 \expandafter\xdef\csname the#1\endcsname{\noexpand\@number
60.830 {\expandafter\noexpand\csname the#3\endcsname \@thmcountersep
60.831 \@thmcounter{#1}}}%
60.832 \global\@namedef{#1}{\@thm{#1}{#2}}%
60.833 \global\@namedef{end#1}{\@endtheorem}}}
60.834 %
60.835\def\@opargbegintheorem#1#2#3{%
60.836 \trivlist
60.837 \item[\hskip \labelsep{\bfseries #1\ #2\
60.838 \@brackets({#3})}]\itshape}
```

\@chapter The following was added for pretty printing of the chapter numbers, for supporting \@schapter 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.

| 60.839 \@ifclassloaded\{article\}\{\}\{\% |  |
| :---: | :---: |
| 60.840 | \% For pretty priniting |
| 60.841 | \def \@@chapapp\{Chapter\} |
| 60.842 | \def \@@thechapter\{\@@arabic\c@chapter\} |
| 60.843 | \def \@chapter[\#1] \#2\{\% |
| 60.844 | \let $\backslash$ headlanguage=\languagename\% |
| 60.845 | \%\set@outputdblcol\% |
| 60.846 | \ifnum \c@secnumdepth > \m@ne |
| 60.847 | $\backslash r e f s t e p c o u n t e r\{c h a p t e r\} \%$ |
| 60.848 | \typeout\{\@@chapapp\space\@@thechapter.\}\% |
| 60.849 | \addcontentsline\{toc\}\{chapter\}\% |
| 60.850 | \{\protect\numberline\{\thechapter\}\#1\} |
| 60.851 | \addcontentsline\{cot\}\{chapter\}\% |
| 60.852 | \{\protect $\backslash$ numberline\{\thechapter\}\#1\} |
| 60.853 | $\backslash \mathrm{else}$ |
| 60.854 | \addcontentsline\{toc\}\{chapter\}\{\#1\}\% |
| 60.855 | \addcontentsline\{cot\}\{chapter\}\{\#1\}\% |
| 60.856 | \fi |
| 60.857 | \chaptermark\{\#1\} |
| 60.858 | \addtocontents\{lof\}\{\protect\addvspace\{10\p@\}\}\% |
| 60.859 | \addtocontents\{fol\}\{\protect\addvspace\{10\p@\}\}\% |
| 60.860 | \addtocontents\{lot\}\{\protect\addvspace\{10\p@\}\}\% |
| 60.861 | \addtocontents\{tol\}\{\protect\addvspace\{10\p@\}\}\% |
| 60.862 | \if@twocolumn |
| 60.863 | \@topnewpage[\@makechapterhead\{\#2\}]\% |
| 60.864 | $\backslash \mathrm{else}$ |
| 60.865 | \@makechapterhead\{\#2\}\% |
| 60.866 | \@afterheading |
| 60.867 | \fi\} |
| 60.868 | \% |
| 60.869 | \def\@schapter\#1\{\% |
| 60.870 | $\backslash$ let $\backslash$ headlanguage=\languagename\% |
| 60.871 | \% \set@outputdblcol\% |
| 60.872 | \if@twocolumn |
| 60.873 | \@topnewpage[\@makeschapterhead\{\#1\}]\% |

```
60.874 \else
60.875 \@makeschapterhead{#1}%
60.876 \@afterheading
60.877 \fi}}
```

\appendix Changed mainly for pretty printing of appendix numbers, and to start two-column mode with the right language (if needed).

```
60.878 \@ifclassloaded{letter}{}{% other
60.879 \@ifclassloaded{slides}{}{% other
60.880 \@ifclassloaded{article}{% article
60.881 \renewcommand\appendix{\par
60.882 \setcounter{section}{0}%
60.883 \setcounter{subsection}{0}%
60.884 \renewcommand\thesection{\@Alph\c@section}}
60.885 }{% report and book
60.886 \renewcommand\appendix{\par
60.887 %\set@outputdblcol%
60.888 \setcounter{chapter}{0}%
60.889 \setcounter{section}{0}%
60.890 \renewcommand\@chapapp{\appendixname}%
60.891 % For pretty priniting
60.892 \def \@@chapapp{Appendix}%
60.893 \def\@@thechapter{\@@Alph\c@chapter}
60.894 \renewcommand\thechapter{\@Alph\c@chapter}}}}}
```


### 60.4.11 Bibliography and citations

\@cite Citations are produced by the macro \@cite\{LABEL\}\{NOTE\}. Both the citation \@biblabel label and the note is typeset in the current direction. We have to use \@brackets \@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.

```
60.895\def\@cite#1#2{\@brackets[{#1\if@tempswa , #2\fi}]}
60.896 \def\@biblabel#1{\@brackets[{#1}]}
60.897 \def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
60.898 {\let\protect\noexpand
60.899 \immediate
60.900 \if@rl\write\@auxout{\string\bibcite{#2}{\R{#1}}}%
60.901 \else\write\@auxout{\string\bibcite{#2}{\L{#1}}}\fi%
60.902 }\fi\ignorespaces}
```

thebibliography Use \rightmargin instead of \leftmargin when in RL mode.
60.903 \@ifclassloaded\{letter\}\{\}\{\% other
60.904 \@ifclassloaded\{slides\}\{\}\{\% other
60.905 \@ifclassloaded\{article\}\{\%
60.906 \renewenvironment\{thebibliography\} [1]
60.907 \{\section*\{\refname\@mkboth\%
$60.908 \quad\{\backslash$ MakeUppercase $\backslash$ refname $\} \%$
$60.909 \quad\{\backslash$ MakeUppercase $\backslash$ refname $\} \%$

```
60.910 \list{\@biblabel{\@arabic\c@enumiv}}%
60.911 {\settowidth\labelwidth{\@biblabel{#1}}%
60.912 \if@rl\leftmargin\else\rightmargin\fi\labelwidth
60.913 \advance\if@rl\leftmargin\else\rightmargin\fi\labelsep
6 0 . 9 1 4 \ \ @ o p e n b i b @ c o d e
60.915 \usecounter{enumiv}%
60.916 \let\p@enumiv\@empty
60.917 \renewcommand\theenumiv{\@arabic\c@enumiv}}%
6 0 . 9 1 8 ~ \ s l o p p y
6 0 . 9 1 9 ~ \ c l u b p e n a l t y 4 0 0 0 ~
60.920 \@clubpenalty \clubpenalty
60.921 \widowpenalty4000%
60.922 \sfcode`\.\@m}
60.923 {\def\@noitemerr
60.924 {\@latex@warning{Empty 'thebibliography' environment}}%
6 0 . 9 2 5 ~ \ e n d l i s t \} \} \% \%
60.926 {\renewenvironment{thebibliography} [1] {%
60.927 \chapter*{\bibname\@mkboth%
60.928 {\MakeUppercase\bibname}%
60.929 {\MakeUppercase\bibname}}%
60.930 \list{\@biblabel{\@arabic\c@enumiv}}%
60.931 {\settowidth\labelwidth{\@biblabel{#1}}%
60.932 \if@rl\leftmargin\else\rightmargin\fi\labelwidth
6 0 . 9 3 3
6 0 . 9 3 4
6 0 . 9 3
6 0 . 9 3 6
6 0 . 9 3 7
6 0 . 9 3 8
6 0 . 9 3 9
60.940 \@clubpenalty \clubpenalty
60.941 \widowpenalty4000%
60.942 \sfcode`\.\@m}
60.943 {\def\@noitemerr
60.944 {\@latex@warning{Empty 'thebibliography' environment}}%
60.945 \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.

```
60.946 \def\@verbatim{%
60.947 \let\do\@makeother \dospecials%
60.948 \obeylines \verbatim@font \@noligs}
```

\@makecaption Captions are set always centered. This allows us to use bilingual captions, for example: \caption\{\R\{RLtext\} $\backslash \backslash \backslash L\{L R t e x t\}\}$, which will be formatted as:

> Right to left caption here (RLtext)
> Left to right caption here (LRtext)

See also \bcaption command below.

```
60.949 \long\def\@makecaption#1#2{%
60.950 \vskip\abovecaptionskip%
6 0 . 9 5 1 ~ \ b e g i n \{ c e n t e r \} \%
60.952 #1: #2%
60.953 \end{center} \par%
60.954 \vskip\belowcaptionskip}
```


### 60.4.12 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 $\backslash \mathrm{R}$ 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 $\backslash \mathrm{b} x x x$ version of the sectioning commands in the text instead of the $\backslash x x x$ version ( $x x x$ is one of: part, chapter, section, subsection, subsubsection, caption)
2. Syntax of the $\backslash \mathrm{b} x x x$ command is $\backslash \mathrm{b} x x x\{R L$ text $\}\{L R$ 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 $\{R L$ text $\}$ will be typeset if Right-to-Left mode is in effect and $\{L R$ text $\}$ otherwise.
```
    \bpart
    60.955 \newcommand{\bpart}[2]{\part{\protect\if@rl%
    60.956 #1 \protect\else #2 \protect\fi}}
\bchapter
    60.957 \newcommand{\bchapter}[2]{\chapter{\protect\if@rl%
    60.958 #1 \protect\else #2 \protect\fi}}
\bsection
    60.959 \newcommand{\bsection}[2]{\section{\protect\if@rl%
    60.960 #1 \protect\else #2 \protect\fi}}
```

\bsubsection

```
60.961 \newcommand{\bsubsection} [2]{\subsection{\protect\if@rl%
60.962 #1 \protect\else #2 \protect\fi}}
```

\bsubsubsection
60.963 \newcommand\{\bsubsubsection\}[2]\{\subsubsection\{\protect\if@rl\%
60.964 \#1 \protect\else \#2 \protect\fi\}\}
\bcaption
60.965 \newcommand\{ $\backslash$ bcaption\} [2] \{\%
60.966 \caption[\protect\if@rl \R\{\#1\}\protect\else \L\{\#2\}\protect $\backslash f i]\{\%$
60.967 \if@rl\R\{\#1\}\protect<br> \L\{\#2\}
60.968 \else $\backslash L\{\# 2\} \backslash$ protect $\backslash \backslash \backslash R\{\# 1\} \backslash$ fi\}\}

The following definition is a modified version of \bchapter, meant as a bilingual twin for \chapter* and \section* (added by Irina Abramovici).

## \bchapternn

```
60.969 \newcommand{\bchapternn}[2]{\chapter*{\protect\if@rl%
60.970 #1 \protect\else #2 \protect\fi}}
```

\bsectionnn

```
60.971 \newcommand{\bsectionnn}[2]{\section*{\protect\if@rl%
60.972 #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.
60.973 \AtEndOfPackage\{\rlAtEndOfPackage\}
60.974 \%
60.975 \def $\backslash r l$ AtEndOfPackage\{\%
60.976 \global\let $\backslash$ headlanguage=\languagename\% $\backslash$ set@outputdblcol\%
60.977 \}
60.978 〈/rightleft〉

### 60.5 Hebrew calendar

The original version of the package hebcal.sty ${ }^{70}$ for $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ and $\mathrm{IA}_{\mathrm{E}} \mathrm{X} 2.09$, entitled "TEX \& EATEX macros for computing Hebrew date from Gregorian one" was created by Michail Rozman, misha@iop.tartu.ew.su ${ }^{71}$

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)

[^61]The package was adjusted for babel and $\mathrm{LAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ 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{hebcal}
```

Two main user-level commands are provided by this package:
$\backslash$ 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).
$\backslash$ 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:

```
\newcount\hd \newcount\hm \newcount\hy
\hd=10 \hm=3 \hy=1992
\Hebrewdate{\hd}{\hm}{\hy}
```

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.

### 60.5.1 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 $4^{\text {th }}$ 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.

### 60.5.2 Registers, Commands, Formatting Macros

The command \Hebrewtoday produces today's date for Hebrew calendar. It is similar to the standard $\mathrm{AA}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ command $\backslash$ today. In addition three numerical registers \Hebrewday, \Hebrewmonth and \Hebrewyear are set. For setting this registers without producing of date string command $\backslash$ 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:
$\backslash$ HebrewFromGregorian \{ Gday\}\{Gmonth\}\{ Gyear\}\{Hday\}\{Hmonth\}\{Hyear\}
can be used.
60.979 〈*calendar〉
60.980 \newif \if@full@hebrew@year
60.981 \@full@hebrew@yearfalse
60.982 \DeclareOption\{full\}\{\@full@hebrew@yeartrue\}
60.983 \ProcessOptions
60.984 \newcount $\backslash$ Hebrewday \newcount $\backslash$ Hebrewmonth \newcount $\backslash$ Hebrewyear
$\backslash$ 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.
60.985 \def $\backslash$ Hebrewdate\#1\#2\#3\{\%
60.986 \HebrewFromGregorian\{\#1\}\{\#2\}\{\#3\}
60.987 \{\Hebrewday\}\{\Hebrewmonth\}\{\Hebrewyear\}\%
60.988 \ifundefined\{if@rl\}\%
$\backslash$ FormatForEnglish\{\Hebrewday\}\{\Hebrewmonth\}\{\Hebrewyear\}\%
60.989
60.990
$60.991 \backslash$ FormatDate $\{\backslash$ Hebrewday $\}\{\backslash$ Hebrewmonth $\}\{\backslash$ Hebrewyear $\} \%$
60.992 \fi\}
\Hebrewtoday Today's date in Hebrew calendar.
$60.993 \backslash$ def $\backslash$ Hebrewtoday $\{\backslash$ Hebrewdate $\{\backslash$ day $\}\{\backslash$ month $\}\{\backslash$ year $\}\}$
60.994 \let $\backslash$ hebrewtoday=\Hebrewtoday
$\backslash H e b r e w s e t r e g$ Set registers: today's date in hebrew calendar.
$\begin{array}{l}\text { 60.995 \def } \backslash \text { Hebrewsetreg }\{\% \\ 60.996 \quad \backslash \text { HebrewFromGregorian }\{\backslash \text { day }\}\{\backslash \text { month }\}\{\backslash \text { year }\} \\ 60.997\end{array} \quad\{\backslash$ Hebrewday $\}\{\backslash$ Hebrewmonth $\}\{\backslash$ Hebrewyear $\left.\}\right\}$
$\backslash$ FormatDate Prints a Hebrew calendar date Hebrewday. Hebrewmonth. Hebrewyear.

```
60.998 \def\FormatDate#1#2#3{%
60.999 \if@rl%
60.1000 \FormatForHebrew{#1}{#2}{#3}%
6 0 . 1 0 0 1 ~ \ e l s e \%
60.1002 \FormatForEnglish{#1}{#2}{#3}
60.1003 \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.
60.1004 \def $\backslash$ HebrewYearName\#1 \{ \{ \%
60.1005 \@tempcnta=\#1\divide\@tempenta by $1000 \backslash$ multiply $\backslash$ @tempcnta by 1000
60.1006 \ifnum\#1=\@tempcnta\relax \% divisible by 1000: disambiguate
$60.1007 \backslash$ Hebrewnumeralfinal\{\#1\}<br>) \heblamed\hebpe"\hebgimel(\%
60.1008 \else \% not divisible by 1000
60.1009 \ifnum\#1<1000\relax $\%$ first millennium: disambiguate
$60.1010 \backslash$ Hebrewnumeralfinal\{\#1\}\ ) \heblamed $\backslash$ hebpe" $\backslash$ hebgimel (\%
60.1011 \else
60.1012 \ifnum\#1<5000
$60.1013 \backslash$ Hebrewnumeralfinal\{\#1\}\%
60.1014 \else
60.1015 \ifnum\#1<6000 \% current millenium, print without thousands
60.1016 \@tempcnta=\#1 \relax
60.1017 \if@full@hebrew@year \else\advance\@tempcnta by -5000\fi
60.1018 \Hebrewnumeralfinal\{\@tempenta\}\%
$60.1019 \quad$ \else \% \#1>6000
60.1020 \Hebrewnumeralfinal\{\#1\}\%
60.1021 \fi
60.1022 \fi
60.1023 \fi
60.1024 \fi\}\}
\HebrewMonthName The macro \HebrewMonthName\{month\}\{year\} returns the name of month in the 'year'.

```
60.1025 \def\HebrewMonthName#1#2{%
60.1026 \ifnum #1 = 7 %
60.1027 \CheckLeapHebrewYear{#2}%
60.1028 \if@HebrewLeap \hebalef\hebdalet\hebresh\ \hebbet%%
60.1029 \else \hebalef\hebdalet\hebresh%
60.1030 \fi%
6 0 . 1 0 3 1 ~ \ e l s e \%
60.1032 \ifcase#1%
60.1033 % nothing for 0
60.1034 \or\hebtav\hebshin\hebresh\hebyod%
60.1035 \or\hebhet\hebshin\hebvav\hebfinalnun%
60.1036 \or\hebkaf \hebsamekh\heblamed\hebvav%
60.1037 \or\hebtet\hebbet\hebtav%
60.1038 \or\hebshin\hebbet\hebtet%
60.1039 \or\hebalef\hebdalet\hebresh\ \hebalef,%
60.1040 \or\hebalef\hebdalet\hebresh\ \hebbet,%
```

```
60.1041 \or\hebnun\hebyod\hebsamekh\hebfinalnun%
60.1042 \or\hebalef \hebyod\hebyod\hebresh%
60.1043 \or\hebsamekh\hebyod\hebvav\hebfinalnun%
60.1044 \or\hebtav\hebmem\hebvav\hebzayin%
60.1045 \or\hebalef \hebbet%
60.1046 \or\hebalef\heblamed\hebvav\heblamed%
60.1047 \fi%
60.1048 \fi}
```

$\backslash H e b r e w D a y N a m e ~ N a m e ~ o f ~ d a y ~ i n ~ H e b r e w ~ l e t t e r s ~(g i m a t r i a) . ~$
60.1049 \def $\backslash$ HebrewDayName\#1\{\Hebrewnumeral\{\#1\}\}
$\backslash$ FormatForHebrew The macro \FormatForHebrew\{hday\}\{hmonth \}\{hyear\} returns the formatted Hebrew date in Hebrew language.

```
60.1050 \def\FormatForHebrew#1#2#3{%
60.1051 \HebrewDayName{#1}~\hebbet\HebrewMonthName{#2}{#3}, ~%
60.1052 \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.
60.1053 \def \HebrewMonthNameInEnglish\#1\#2\{\%
60.1054 \ifnum \#1 = 7\%
60.1055 \CheckLeapHebrewYear\{\#2\}\%
60.1056 \if@HebrewLeap Adar II\else Adar $\backslash f i \%$
60.1057 \else\%
60.1058 \ifcase \#1\%
$60.1059 \quad$ \% nothing for 0
60.1060 \or Tishrei\%
60.1061 \or Heshvan\%
60.1062 \or Kislev\%
60.1063 \or Tebeth\%
60.1064 \or Shebat\%
60.1065 \or Adar I\%
60.1066 \or Adar II\%
60.1067 \or Nisan\%
60.1068 \or Iyar\%
60.1069 \or Sivan\%
60.1070 \or Tammuz\%
60.1071 \or Av\%
60.1072 \or Elul\%
60.1073 \fi
60.1074 \fi\}
\FormatForEnglish The macro \FormatForEnglish\{hday\}\{hmonth \}\{hyear\} is similar to \FormatForHebrew macro described above and returns the formatted Hebrew date in English.
60.1075 \def $\backslash$ FormatForEnglish\#1\#2\#3\{\%
60.1076 \HebrewMonthNameInEnglish\{\#2\}\{\#3\} \number\#1,\ \number\#3\}

### 60.5.3 Auxiliary Macros

60.1077 \newcount \@common
$\backslash$ Remainder $\backslash$ Remainder $\{a\}\{b\}\{c\}$ calculates $c=a \% b==a-b \times \frac{a}{b}$
60.1078 \def $\backslash$ Remainder\#1\#2\#3\{\%
60.1079 \#3 = \#1\% \% c = a
60.1080 \divide \#3 by \#2\% \% c = a/b
60.1081 \multiply \#3 by -\#2\% \% c = -b (a/b)
60.1082 \advance \#3 by \#1\}\% \% c = a - b(a/b)
60.1083 \newif $\backslash i f @ D i v i s i b l e$
$\backslash$ CheckIfDivisible \CheckIfDivisible\{a\}\{b\} sets \@Divisibletrue if $a \% b==0$
60.1084 \def $\backslash$ CheckIfDivisible\#1\#2\{\%
60.1085 \{\%
60.1086 \countdef $\backslash$ tmp $=0 \%$ tmp == \count0 - temporary variable
$60.1087 \backslash$ Remainder $\{\# 1\}\{\# 2\}\{\backslash$ tmp $\} \%$
60.1088 \ifnum \tmp = 0\%
$60.1089 \quad$ \global\@Divisibletrue\%
60.1090 \else\%
$60.1091 \quad$ \global\@Divisiblefalse\%
60.1092 \fi\}\}
\ifundefined From the $\mathrm{T}_{\mathrm{E}} \mathrm{Xb}$ book, ex. 7.7:
\ifundefined\{command\}<true text>\else<false text>\fi
$60.1093 \backslash$ def $\backslash$ ifundefined\#1 $\{\backslash$ expandafter $\backslash i f x \backslash c s n a m e \# 1 \backslash e n d c s n a m e \backslash r e l a x\}$

### 60.5.4 Gregorian Part

60.1094 \newif $\backslash$ if@GregorianLeap
$\backslash I f G r e g o r i a n L e a p ~ C o n d i t i o n a l ~ w h i c h ~ i s ~ t r u e ~ i f ~ G r e g o r i a n ~ ' y e a r ' ~ i s ~ a ~ l e a p ~ y e a r: ~(~(y e a r \% 4==0) ~ \wedge ~$
$($ year $\% 100 \neq 0)) \vee($ year $\% 400==0)$
60.1095 \def \IfGregorianLeap\#1\{\%
60.1096 \CheckIfDivisible\{\#1\}\{4\}\%
60.1097 \if@Divisible\%
60.1098 \CheckIfDivisible\{\#1\}\{100\}\%
60.1099 \if@Divisible\%
$60.1100 \quad$ CheckIfDivisible\{\#1\}\{400\}\%
60.1101 \if@Divisible\%
60.1102 \@GregorianLeaptrue\%
60.1103 \else\%
60.1104 \@GregorianLeapfalse\%
$60.1105 \quad \backslash f i \%$
60.1106 \else\%

```
60.1107 \@GregorianLeaptrue%
60.1108 \fi%
60.1109 \else%
60.1110 \@GregorianLeapfalse%
60.1111 \fi%
60.1112 \if@GregorianLeap}
```

\GregorianDaysInPriorMonths The macro \GregorianDaysInPriorMonths\{month\}\{year\}\{days\} calculates the number of days in months prior to 'month' in the 'year'.

```
60.1113\def\GregorianDaysInPriorMonths#1#2#3{%
60.1114 {%
60.1115 #3 = \ifcase #1%
6 0 . 1 1 1 6 ~ 0 ~ \ o r \% ~ \% ~ n o ~ m o n t h ~ n u m b e r ~ 0 ~
60.1117 0 \or%
60.1118 31 \or%
60.1119 - 59 \or%
60.1120 90 \or%
60.1121 120 \or%
60.1122 151 \or%
60.1123 181 \or%
60.1124 212 \or%
60.1125 243 \or%
60.1126 273 \or%
60.1127 304 \or%
60.1128 334%
6 0 . 1 1 2 9 ~ \ f i \%
60.1130 \IfGregorianLeap{#2}%
60.1131 \ifnum #1 > 2% % if month after February
60.1132 \advance #3 by 1% % add leap day
60.1133 \fi%
60.1134 \fi%
60.1135 \global\@common = #3}%
60.1136 #3 = \@common}
```

\GregorianDaysInPriorYears The macro \GregorianDaysInPriorYears\{year\}\{days\} calculates the number of days in years prior to the 'year'.
60.1137 \def $\backslash$ GregorianDaysInPriorYears\#1\#2\{\%
60.1138 \{\%
60.1139 \countdef $\backslash$ tmpc $=4 \%$ \tmpc==
$60.1140 \quad$ countdef $\backslash$ tmpb $=2 \% \quad \%$ ttmpb== \count2
$60.1141 \quad$ ไtmpb $=\# 1 \%$ \%
60.1142 \advance \tmpb by -1\% \%
$60.1143 \quad$ ไtmpc $=\backslash$ tmpb $\% \quad$ \% tmpc $=\backslash$ tmpb $=$ year-1
60.1144 \multiply \tmpc by 365\% \% Days in prior years =
$60.1145 \quad \# 2=$ tmpc\% \% = 365*(year-1) ...
$60.1146 \quad$ \tmpc $=$ \tmpb\% $\%$
$60.1147 \quad$ \divide $\backslash t m p c$ by $4 \% \quad \%$ \tmpc $=$ (year-1)/4
60.1148 \advance \#2 by \tmpc\% \% ... plus Julian leap days ...
$60.1149 \quad$ \tmpc $=\backslash t m p b \% \quad \%$

| 60.1150 | \divide \tmpc by 100\% | \% \tmpc = (year-1)/100 |
| :---: | :---: | :---: |
| 60.1151 | \advance \#2 by - \tmpc\% | \% ... minus century years |
| 60.1152 | $\backslash t m p c=\ t m p b \%$ | \% |
| 60.1153 | \divide \tmpc by 400\% | \% \tmpc = (year-1)/400 |
| 60.1154 | \advance \#2 by \tmpc\% | \% ... plus 4-century years. |
| 60.1155 | $\backslash \mathrm{global} \backslash$ @common $=$ \# 2$\} \%$ |  |
| 60.1156 | \@common\} |  |

 the absolute date (days since 01.01.0001) from Gregorian date day.month. year.

```
60.1157\def\AbsoluteFromGregorian#1#2#3#4{%
60.1158 {%
60.1159 \countdef\tmpd = 0% % \tmpd==\count0
60.1160 #4 = #1% % days so far this month
60.1161 \GregorianDaysInPriorMonths{#2}{#3}{\tmpd}%
60.1162 \advance #4 by \tmpd% % add days in prior months
60.1163 \GregorianDaysInPriorYears{#3}{\tmpd}%
60.1164 \advance #4 by \tmpd% % add days in prior years
60.1165 \global\@common = #4}%
60.1166 #4 = \@common}
```


### 60.5.5 Hebrew Part

60.1167 \newif \if@HebrewLeap
$\backslash$ CheckLeapHebrewYear Set \@HebrewLeaptrue if Hebrew 'year' is a leap year, i.e. if ( $1+7 \times$ year $) \% 19<7$ then true else false

```
60.1168 \def \CheckLeapHebrewYear#1{%
60.1169 {%
60.1170 \countdef\tmpa = 0% % \tmpa==\count0
60.1171 \countdef\tmpb = 1% % \tmpb==\count1
60.1172 %
60.1173 \tmpa = #1%
60.1174 \multiply \tmpa by 7%
60.1175 \advance \tmpa by 1%
60.1176 \Remainder{\tmpa}{19}{\tmpb}%
60.1177 \ifnum \tmpb < 7% % \tmpb = (7*year+1)%19
60.1178 \global\@HebrewLeaptrue%
60.1179 \else%
60.1180 \global\@HebrewLeapfalse%
60.1181 \fi}}
```

 months elapsed from the Sunday prior to the start of the Hebrew calendar to the mean conjunction of Tishri of Hebrew 'year'.

```
60.1182\def\HebrewElapsedMonths#1#2{%
60.1183 {%
60.1184 \countdef\tmpa = 0% % \tmpa==\count0
60.1185 \countdef\tmpb = 1% % \tmpb==\count1
```

| 60.1186 | $\backslash$ countdef $\backslash$ tmpc $=2 \%$ | \% \tmpc==\count2 |
| :---: | :---: | :---: |
| 60.1187 \% |  |  |
| 60.1188 | $\backslash$ tmpa $=\# 1 \%$ | \% |
| 60.1189 | \advance \tmpa by -1\% | \% |
| 60.1190 | \#2 = \tmpa\% | \% \#2 = \tmpa $=$ year-1 |
| 60.1191 | \divide \#2 by 19\% | \% Number of complete Meton cycles |
| 60.1192 | \multiply \#2 by 235\% | \% \#2 = 235* ( year-1)/19) |
| 60.1193 \% |  |  |
| 60.1194 | $\backslash$ Remainder $\{\backslash$ tmpa\}\{19\}\{\tmpb\}\% \tmpa $=$ years\%19-years this cycle |  |
| 60.1195 | $\backslash \mathrm{tmpc}=\$ tmpb\% & \%  \hline 60.1196 & $\backslash$ multiply \tmpb by 12\% | \% |
| 60.1197 | \advance \#2 by \tmpb\% | \% add regular months this cycle |
| 60.1198 \% |  |  |
| 60.1199 | $\backslash$ multiply \tmpc by 7\% | \% |
| 60.1200 | \advance \tmpc by 1\% | \% |
| 60.1201 | \divide \tmpc by 19\% | \% $\backslash$ tmpc $=(1+7 *(($ year -1$) \% 19)) / 19-$ |
| 60.1202 \% |  | \% number of leap months this cycle |
| 60.1203 | \advance \#2 by \tmpc\% | \% add leap months |
| 60.1204 \% |  |  |
| 60.1205 | $\backslash \mathrm{global} \backslash$ @common = \#2\}\% |  |
| 60.1206 | = \@common\} |  |

$\backslash H e b r e w E l a p s e d D a y s ~ T h e ~ m a c r o ~ \ H e b r e w E l a p s e d D a y s\{y e a r\}\{d a y s\} ~ d e t e r m i n e s ~ t h e ~ n u m b e r ~ o f ~ d a y s ~$ elapsed from the Sunday prior to the start of the Hebrew calendar to the mean conjunction of Tishri of Hebrew 'year'.
60.1207 \def \HebrewElapsedDays\#1\#2\{\%
60.1208 \{\%
$60.1209 \quad$ countdef $\backslash$ tmpa $=0 \% \quad \%$ ไtmpa== $\backslash$ count0
$60.1210 \quad$ Ccountdef $\backslash$ tmpb $=1 \%$ \% $1 \%$ tmpb==\count1
60.1211 \countdef $\backslash$ tmpc $=2 \%$ \tmpc==\count2
60.1212 \%
60.1213 \HebrewElapsedMonths\{\#1\}\{\#2\}\%
$60.1214 \quad$ ttmpa $=\# 2 \%$ \%
60.1215 \multiply \tmpa by 13753\% \%
60.1216 \advance \tmpa by 5604\% \% \tmpa=MonthsElapsed*13758 + 5604
$60.1217 \backslash$ Remainder $\{\backslash$ tmpa $\}\{25920\}\{\backslash$ tmpc $\} \%$ $\backslash$ tmpc $==$ ConjunctionParts
60.1218 \divide \tmpa by $25920 \%$
60.1219 \%
60.1220 \multiply \#2 by 29\%
60.1221 \advance \#2 by 1\%
60.1222 \advance \#2 by \tmpa\% \% \#2 = 1 + MonthsElapsed*29 +
$60.1223 \%$ \% PartsElapsed/25920
$60.1224 \backslash$ Remainder $\{\# 2\}\{7\}\{\backslash$ tmpa\}\% \% \tmpa == DayOfWeek
60.1225 \ifnum $\backslash$ tmpc < 19440\%
60.1226 \ifnum \tmpc < 9924\%
60.1227 \else\% New moon at 9 h. 204 p. or later
60.1228 \ifnum $\backslash$ tmpa $=2 \% \%$ on Tuesday $\ldots$
60.1229 \CheckLeapHebrewYear\{\#1\}\% of a common year
60.1230 \if@HebrewLeap\%
60.1231 \else\%
60.1232
60.1233
60.1234
60.1235
60.1236
60.1238
60.1239
60.1240
60.1241
60.1242
60.1243
60.1244
60.1245
60.1246
60.1248
60.1249 \%
60.1251
60.1252
60.1253
60.1254
60.1255
60.1256
60.1257
60.1258
60.1259
60.1260
60.1261
60.1263
60.1247 \advance \#2 by $1 \%$ new moon at or after midday
$60.1250 \backslash$ Remainder $\{\# 2\}\{7\}\{\backslash$ tmpa\} $\% ~ \% ~ \$ tmpa $==$ DayOfWeek
$60.1262 \quad$ gglobal \@common $=\# 2\} \%$
\advance \#2 by 1\% \fi\% \fi\%
\fi\%
\ifnum \tmpc < 16789\%
\else\% \% New moon at 15 h .589 p. or later \ifnum \tmpa $=1 \% \%$ on Monday..
\advance \#1 by $-1 \%$
$\backslash$ CheckLeapHebrewYear\{\#1\}\% at the end of leap year
\if@HebrewLeap\%
\advance \#2 by $1 \%$
\fi\% $\backslash f i \%$
$\backslash f i \%$
$\backslash e l s e \%$
$\backslash f i \%$
\ifnum \tmpa $=0 \% \quad \%$ if Sunday..
\advance \#2 by $1 \%$
$\backslash e l s e \% \quad \%$
\ifnum \tmpa $=3 \% \quad \%$ Wednesday.. \advance \#2 by $1 \%$
\else\%
\ifnum \tmpa $=5 \%$ \% or Friday
\advance \#2 by $1 \%$
$\backslash f i \%$
\fi\%
$\backslash f i \%$
\#2 = \@common\}
\DaysInHebrewYear The macro \DaysInHebrewYear\{year\}\{days\} calculates the number of days in Hebrew 'year'. 60.1264 \def \DaysInHebrewYear\#1\#2\{\%
60.1265 \{\%
60.1266 \countdef $\backslash$ tmpe $=12 \% \quad \%$ tmpe==$\backslash$ count12
60.1267 \%
60.1268 \HebrewElapsedDays\{\#1\}\{\tmpe\}\%
60.1269 \advance \#1 by 1\%
$60.1270 \backslash$ HebrewElapsedDays\{\#1\}\{\#2\}\%
60.1271 \advance \#2 by -\tmpe\%
$60.1272 \quad$ \global \@common $=\# 2\} \%$
$60.1273 \quad \# 2=$ \@common\}
$\backslash H e b r e w D a y s I n P r i o r M o n t h s ~ T h e ~ m a c r o ~ \ H e b r e w D a y s I n P r i o r M o n t h s\{m o n t h\}\{y e a r\}\{d a y s\} ~ c a l c u l a t e s ~ t h e ~ n u-~$ mber of days in months prior to 'month' in the 'year'.
60.1274 \def \HebrewDaysInPriorMonths\#1\#2\#3\{\%

```
60.1275 {%
60.1276 \countdef\tmpf= 14% % \tmpf==\count14
60.1278 #3 = \ifcase #1% % Days in prior month of regular year
6 0 . 1 2 7 9 ~ 0 ~ \ o r \% ~ \% ~ n o ~ m o n t h ~ n u m b e r ~ 0 ~
60.1280 0 \or% % Tishri
6 0 . 1 2 8 1 ~ 3 0 ~ \ o r \% ~ \% ~ H e s h v a n ~
6 0 . 1 2 8 2 ~ 5 9 ~ \ o r \% ~ \% ~ K i s l e v ~
60.1283 89 \or% % Tebeth
6 0 . 1 2 8 4 ~ 1 1 8 ~ \ o r \% ~ \% ~ S h e b a t
60.1285 148 \or% % Adar I
6 0 . 1 2 8 6 ~ 1 4 8 ~ \ o r \% ~ \% ~ A d a r ~ I I ~
6 0 . 1 2 8 7 ~ 1 7 7 ~ \ o r \% ~ \% ~ N i s a n
60.1288 207 \or% % Iyar
60.1289 236 \or% % Sivan
60.1290 266 \or% % Tammuz
6 0 . 1 2 9 1 ~ 2 9 5 ~ \ o r \% ~ \% ~ A v ~
60.1292 325 \or% % Elul
60.1293 400% % Dummy
60.1294 \fi%
60.1295 \CheckLeapHebrewYear{#2}%
60.1296 \if@HebrewLeap% % in leap year
60.129
6 0 . 1 2 9 8
6 0 . 1 2 9 9 ~ \ f i \%
60.1300 \fi%
60.1301 \DaysInHebrewYear{#2}{\tmpf}%
60.1302 \ifnum #1 > 3%
60.1303
6 0 . 1 3 0 4
60.1305 \fi% % Short Kislev
60.1306 \ifnum \tmpf = 383% %
60.1307 \advance #3 by -1% %
60.1308 \fi% %
60.1309
60.1310 %
60.1311 \ifnum #1 > 2%
60.1312 \ifnum \tmpf = 355% %
60.1313 \advance #3 by 1% %
6 0 . 1 3 1 4 ~ \ f i \% ~ \% ~ L o n g ~ H e s h v a n
60.1315 \ifnum \tmpf = 385% %
60.1316 \advance #3 by 1% %
6 0 . 1 3 1 7 ~ \ f i \% ~ \% ~
60.1318 \fi%
60.1319 \global\@common = #3}%
60.1320 #3 = \@common}
```

 absolute date of Hebrew date day.month. year.
60.1321 \def $\backslash$ AbsoluteFromHebrew\#1\#2\#3\#4\{\%

$\backslash$ HebrewFromGregorian The macro \HebrewFromGregorian\{Gday\}\{Gmonth\}\{Gyear\}\{Hday\}\{Hmonth\}\{Hyear\} evaluates Hebrew date Hday, Hmonth, Hyear from Gregorian date Gday, Gmonth, Gyear.
60.1331 \def $\backslash$ HebrewFromGregorian\#1\#2\#3\#4\#5\#6\{\%
60.1332 \{\%
60.1333 \% $\quad$ countdef $\backslash$ tmpx $=17 \%$ tmpx==\count17
60.1334 \countdef $\backslash$ tmpy $=18 \%$ \tmpy==\count18
$60.1335 \quad$ Ccountdef $\backslash t m p z=19 \% \quad$ \% $\backslash t m p z==\backslash c o u n t 19$
60.1336 \%
$60.1337 \quad \# 6=\# 3 \%$ \%
60.1338 \global\advance \#6 by $3761 \%$ approximation from above
$60.1339 \backslash$ AbsoluteFromGregorian\{\#1\}\{\#2\}\{\#3\}\{\#4\}\%
$60.1340 \quad$ \tmpz $=1 \quad$ \tmpy $=1 \%$
$60.1341 \backslash$ AbsoluteFromHebrew\{\tmpz\}\{\tmpy\}\{\#6\}\{\tmpx\}\%
60.1342 \ifnum \tmpx > \#4\% \%
60.1343 \global\advance \#6 by $-1 \%$ Hyear $=$ Gyear +3760
$60.1344 \backslash$ AbsoluteFromHebrew\{\tmpz\}\{\tmpy\}\{\#6\}\{\tmpx\}\%
60.1345 \fi\% \%
60.1346 \advance \#4 by -\tmpx\% \% Days in this year
60.1347 \advance \#4 by 1\% \%
$60.1348 \quad \# 5=\# 4 \%$ \%
$60.1349 \quad$ \% Approximation for month from below
60.1350 \loop\%
$60.1351 \backslash$ HebrewDaysInPriorMonths $\{\# 5\}\{\# 6\}\{\backslash$ tmpx $\} \%$
60.1352 \ifnum \tmpx < \#4\%
60.1353 \advance \#5 by 1\%
$60.1354 \quad$ ttmpy $=\backslash$ tmpx\%
60.1355 \repeat\%
60.1356 \global\advance \#5 by -1\%
60.1357 \global\advance \#4 by -\tmpy\}\}
60.1358 〈/calendar〉

## 61 Hebrew input encodings

Hebrew input encodings defined in file hebinp.dtx ${ }^{72}$ should be used with inputenc $\mathrm{EAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ package. This package allows the user to specify an input

[^62]encoding from this file (for example, ISO Hebrew/Latin 8859-8, IBM Hebrew codepage 862 or MS Windows Hebrew codepage 1255) by saying:
me]\{inputenc\}Theencodingcanalsobeselectedinthedocumentwith:\inputencoding\{encodingname\}Theonlypracticaluseofthiscommandwithinadocumentiswhenusingtextfromseveraldocumentstobuildupacompositeworksuchasavolumeofjournalarticles.Thereforethiscommandwillbeusedonlyinverticalmode.undefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefinedundefined

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}
```

This defines the input character slot to be the text material or math material respectively. For example, 8859-8.def defines slots "EA (letter hebalef) and "B5 ( $\mu$ ) by saying:

```
\DeclareInputText{224}{\hebalef}
\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{\frac14}}
\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 defintion depends on the fonts available.

See the documentation in inputenc.dtx for details of how to declare input definitions for various encodings.

### 61.1 Default definitions for characters

First, we insert a \makeatletter at the beginning of all .def files to use @ symbol in the macros' names.
61.1 〈-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 defintion 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.

```
61.2\langle*8859 - 8 | cp862 | cp1255\rangle
61.3\ProvideTextCommandDefault{\textdegree}{\ensuremath{{^\circ}}}
61.4 \ProvideTextCommandDefault{\textonehalf}{\ensuremath{\frac12}}
61.5\ProvideTextCommandDefault{\textonequarter}{\ensuremath{\frac14}}
61.6 </8859 - 8 | cp862 | cp1255>
61.7 <*8859 - 8 | cp1255\rangle
61.8 \ProvideTextCommandDefault{\textthreequarters}{\ensuremath{\frac34}}
61.9 </8859 - 8 | cp1255\rangle
61.10 <*cp862 | cp1255\rangle
61.11\ProvideTextCommandDefault{\textflorin}{\textit{f}}
61.12 </cp862 | cp1255\rangle
61.13 <*ср862\rangle
61.14 \ProvideTextCommandDefault{\textpeseta}{Pt}
61.15 </cp862\rangle
```

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.

```
61.16 \langle*cp862\rangle
61.17 \ProvideTextCommandDefault{\textblacksquare}
61.18 {\vrule \@width . 3em \@height .4em \@depth -.1em\relax}
61.19 </cp862\rangle
```

Some commands can't be faked, so we have them generate an error message.
$61.20\langle * 8859-8|$ cp862 | cp1255 $\rangle$
61.21 \ProvideTextCommandDefault\{\textcent\}
$61.22 \quad\{\backslash$ TextSymbolUnavailable\textcent\}
61.23 \ProvideTextCommandDefault\{\textyen\}
$61.24 \quad\{\backslash$ TextSymbolUnavailable $\backslash$ textyen $\}$
$61.25\langle/ 8859-8| c p 862|c p 1255\rangle$
$61.26\langle * 8859-8\rangle$
61．27 \ProvideTextCommandDefault\｛\textcurrency\}
61.28 \｛\TextSymbolUnavailable\textcurrency\}
$61.29\langle/ 8859-8\rangle$
$61.30\langle * \mathrm{cp} 1255\rangle$
61．31 \ProvideTextCommandDefault\｛\newsheqel\}
$61.32 \quad\{\backslash$ TextSymbolUnavailable\newsheqel\}
$61.33\langle/ \mathrm{cp} 1255\rangle$
$61.34\langle * 8859-8 \mid c p 1255\rangle$
61．35 \ProvideTextCommandDefault\｛\textbrokenbar\}
$61.36 \quad\{\backslash$ TextSymbolUnavailable\textbrokenbar\}
61.37 〈／8859－8｜cp1255〉
$61.38\langle * \mathrm{cp} 1255\rangle$
61．39 \ProvideTextCommandDefault \｛\textperthousand\}
$61.40 \quad\{\backslash$ TextSymbolUnavailable\textperthousand\}
61.41 〈／cp1255〉

Characters that are supposed to be used only in math will be defined by $\backslash$ providecommand because $\mathrm{AT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ assumes that the font encoding for math fonts is static．
$61.42\langle * 8859-8 \mid c p 1255\rangle$
$61.43 \backslash$ providecommand $\{\backslash$ mathonesuperior $\}\{\{\wedge 1\}\}$
$61.44 \backslash$ providecommand\｛\maththreesuperior\}\{\{~3\}\}
61.45 〈／8859－ 8 ｜cp1255
$61.46\langle * 8859-8|$ cp862｜cp1255 $\rangle$
$61.47 \backslash$ providecommand\｛\mathtwosuperior\}\{\{~2\}\}
61.48 〈／8859－8｜cp862｜cp1255 $\rangle$
61.49 〈＊ cp 862$\rangle$
$61.50 \backslash$ providecommand\｛\mathordmasculine\}\{\{^o\}\}
61.51 \providecommand\｛\mathordfeminine\}\{\{^a\}\}
61.52 〈／cp862〉

## 61．2 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．

## 61．3 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 reper－ toire 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．

```
61.53〈*ср1255\rangle
61.54 \DeclareInputText{130}{\quotesinglbase}
61.55\DeclareInputText{131}{\textflorin}
61.56\DeclareInputText{132}{\quotedblbase}
```

```
61.57\DeclareInputText{133}{\dots}
61.58\DeclareInputText{134}{\dag}
61.59 \DeclareInputText{135}{\ddag}
61.60\DeclareInputText{136}{\~{}}
61.61 \DeclareInputText{137}{\textperthousand}
61.62 \DeclareInputText{139}{\guilsinglleft}
61.63\DeclareInputText{145}{\textquoteleft}
61.64 \DeclareInputText{146}{\textquoteright}
61.65\DeclareInputText{147}{\textquotedblleft}
61.66\DeclareInputText{148}{\textquotedblright}
61.67 \DeclareInputText{149}{\textbullet}
61.68\DeclareInputText{150}{\textendash}
61.69\DeclareInputText{151}{\textemdash}
61.70 \DeclareInputText{152}{\~ {}}
61.71 \DeclareInputText{153}{\texttrademark}
61.72\DeclareInputText{155}{\guilsinglright}
61.73〈/cp1255\rangle
61.74\langle*8859 - 8 | cp1255\rangle
61.75\DeclareInputText{160}{\nobreakspace}
61.76 \DeclareInputText{162}{\textcent}
61.77\DeclareInputText{163}{\pounds}
61.78\langle+8859 - 8\rangle \DeclareInputText{164}{\textcurrency}
61.79 \langle+cp1255\rangle \DeclareInputText{164}{\newsheqel}
61.80\DeclareInputText{165}{\textyen}
61.81 \DeclareInputText{166}{\textbrokenbar}
61.82\DeclareInputText{167}{\S}
61.83\DeclareInputText{168}{\"{}}
61.84 \DeclareInputText{169}{\textcopyright}
61.85 <+8859 - 8\rangle \DeclareInputMath{170}{\times}
61.86\DeclareInputText{171}{\guillemotleft}
61.87\DeclareInputMath{172}{\\not}
61.88 \DeclareInputText{173}{\-}
61.89\DeclareInputText{174}{\textregistered}
61.90\DeclareInputText{175}{\@tabacckludge={}}
61.91 \DeclareInputText{176}{\textdegree}
61.92\DeclareInputMath{177}{\pm}
61.93\DeclareInputMath{178}{\mathtwosuperior}
61.94\DeclareInputMath{179}{\maththreesuperior}
61.95\DeclareInputText{180}{\@tabacckludge'{}}
61.96\DeclareInputMath{181}{\mu}
61.97\DeclareInputText{182}{\P}
61.98\DeclareInputText{183}{\textperiodcentered}
61.99 <+8859 - 8\rangle \DeclareInputText{184}{\c\ }
61.100 \DeclareInputMath{185}{\mathonesuperior}
61.101 \langle+8859 - 8\rangle \DeclareInputMath{186}{\div}
61.102 \DeclareInputText{187}{\guillemotright}
61.103\DeclareInputText{188}{\textonequarter}
61.104 \DeclareInputText{189}{\textonehalf}
61.105\DeclareInputText{190}{\textthreequarters}
```

```
61.106 </8859 - 8 | cp1255\rangle
```

Hebrew vowels and dots (nikud) are included only to MS Windows cp1255 page and start from the position 192.

```
61.107 \langle*cp1255\rangle
61.108\DeclareInputText{192}{\hebsheva}
61.109 \DeclareInputText{193}{\hebhatafsegol}
61.110 \DeclareInputText{194}{\hebhatafpatah}
61.111 \DeclareInputText{195}{\hebhatafqamats}
61.112 \DeclareInputText{196}{\hebhiriq}
61.113 \DeclareInputText{197}{\hebtsere}
61.114 \DeclareInputText{198}{\hebsegol}
61.115\DeclareInputText{199}{\hebpatah}
61.116 \DeclareInputText{200}{\hebqamats}
61.117\DeclareInputText{201}{\hebholam}
61.118 \DeclareInputText{203}{\hebqubuts}
61.119 \DeclareInputText{204}{\hebdagesh}
61.120\DeclareInputText{205}{\hebmeteg}
61.121 \DeclareInputText{206}{\hebmaqaf}
61.122 \DeclareInputText{207}{\hebrafe}
61.123\DeclareInputText{208}{\hebpaseq}
61.124 \DeclareInputText{209}{\hebshindot}
61.125 \DeclareInputText{210}{\hebsindot}
61.126 \DeclareInputText{211}{\hebsofpasuq}
61.127\DeclareInputText{212}{\hebdoublevav}
61.128 \DeclareInputText{213}{\hebvavyod}
61.129 \DeclareInputText{214}{\hebdoubleyod}
61.130 </cp1255\rangle
```

Hebrew letters start from the position 224 in both encodings.

```
61.131 \langle*8859 - 8 | cp1255\rangle
61.132\DeclareInputText{224}{\hebalef}
61.133 \DeclareInputText{225}{\hebbet}
61.134 \DeclareInputText{226}{\hebgimel}
61.135 \DeclareInputText{227}{\hebdalet}
61.136 \DeclareInputText{228}{\hebhe}
61.137 \DeclareInputText{229}{\hebvav}
61.138 \DeclareInputText{230}{\hebzayin}
61.139 \DeclareInputText{231}{\hebhet}
61.140\DeclareInputText{232}{\hebtet}
61.141 \DeclareInputText{233}{\hebyod}
61.142 \DeclareInputText{234}{\hebfinalkaf}
61.143\DeclareInputText{235}{\hebkaf}
61.144 \DeclareInputText{236}{\heblamed}
61.145 \DeclareInputText{237}{\hebfinalmem}
61.146 \DeclareInputText{238}{\hebmem}
61.147 \DeclareInputText{239}{\hebfinalnun}
61.148 \DeclareInputText{240}{\hebnun}
61.149 \DeclareInputText{241}{\hebsamekh}
61.150\DeclareInputText{242}{\hebayin}
```

```
61.151 \DeclareInputText{243}{\hebfinalpe}
61.152 \DeclareInputText{244}{\hebpe}
61.153 \DeclareInputText{245}{\hebfinaltsadi}
61.154 \DeclareInputText{246}{\hebtsadi}
61.155 \DeclareInputText{247}{\hebqof}
61.156 \DeclareInputText{248}{\hebresh}
61.157 \DeclareInputText{249}{\hebshin}
61.158\DeclareInputText{250}{\hebtav}
61.159 </8859 - 8 | cp1255>
```

Special symbols which define the direction of symbols explicitly. Currently, they are not used in $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$.

```
61.160\langle*cp1255\rangle
```

61.161 \DeclareInputText\{253\}\{\lefttorightmark\}
61.162 \DeclareInputText\{254\}\{\righttoleftmark\}
61.163 〈/cp1255〉

### 61.4 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.

```
61.164 <*ср862\rangle
61.165\DeclareInputText{128}{\hebalef}
61.166 \DeclareInputText{129}{\hebbet}
61.167 \DeclareInputText{130}{\hebgimel}
61.168 \DeclareInputText{131}{\hebdalet}
61.169 \DeclareInputText{132}{\hebhe}
61.170 \DeclareInputText{133}{\hebvav}
61.171 \DeclareInputText{134}{\hebzayin}
61.172 \DeclareInputText{135}{\hebhet}
61.173\DeclareInputText{136}{\hebtet}
61.174 \DeclareInputText{137}{\hebyod}
61.175 \DeclareInputText{138}{\hebfinalkaf}
61.176 \DeclareInputText{139}{\hebkaf}
61.177 \DeclareInputText{140}{\heblamed}
61.178 \DeclareInputText{141}{\hebfinalmem}
61.179 \DeclareInputText{142}{\hebmem}
61.180 \DeclareInputText{143}{\hebfinalnun}
61.181 \DeclareInputText{144}{\hebnun}
61.182 \DeclareInputText{145}{\hebsamekh}
61.183 \DeclareInputText{146}{\hebayin}
61.184 \DeclareInputText{147}{\hebfinalpe}
61.185\DeclareInputText{148}{\hebpe}
61.186 \DeclareInputText{149}{\hebfinaltsadi}
61.187 \DeclareInputText{150}{\hebtsadi}
61.188 \DeclareInputText{151}{\hebqof}
61.189 \DeclareInputText{152}{\hebresh}
```

```
61.190\DeclareInputText{153}{\hebshin}
61.191 \DeclareInputText{154}{\hebtav}
61.192 \DeclareInputText{155}{\textcent}
61.193\DeclareInputText{156}{\pounds}
61.194 \DeclareInputText{157}{\textyen}
61.195 \DeclareInputText{158}{\textpeseta}
61.196 \DeclareInputText{159}{\textflorin}
61.197 \DeclareInputText{160}{\@tabacckludge'a}
61.198\DeclareInputText{161}{\@tabacckludge'\i}
61.199 \DeclareInputText{162}{\@tabacckludge'o}
61.200 \DeclareInputText{163}{\@tabacckludge'u}
61.201 \DeclareInputText{164}{\~n}
61.202 \DeclareInputText{165}{\~N}
61.203 \DeclareInputMath{166}{\mathordfeminine}
61.204 \DeclareInputMath{167}{\mathordmasculine}
61.205 \DeclareInputText{168}{\textquestiondown}
61.206 \DeclareInputMath{170}{\lnot}
61.207\DeclareInputText{171}{\textonehalf}
61.208 \DeclareInputText{172}{\textonequarter}
61.209 \DeclareInputText{173}{\textexclamdown}
61.210\DeclareInputText{174}{\guillemotleft}
61.211 \DeclareInputText{175}{\guillemotright}
61.212 \DeclareInputMath{224}{\alpha}
61.213\DeclareInputText{225}{\ss}
61.214 \DeclareInputMath{226}{\Gamma}
61.215\DeclareInputMath{227}{\pi}
61.216\DeclareInputMath{228}{\Sigma}
61.217 \DeclareInputMath{229}{\sigma}
61.218\DeclareInputMath{230}{\mu}
61.219 \DeclareInputMath{231}{\tau}
61.220 \DeclareInputMath{232}{\Phi}
61.221 \DeclareInputMath{233}{\Theta}
61.222 \DeclareInputMath{234}{\Omega}
61.223\DeclareInputMath{235}{\delta}
61.224 \DeclareInputMath{236}{\infty}
61.225\DeclareInputMath{237}{\phi}
61.226 \DeclareInputMath{238}{\varepsilon}
61.227 \DeclareInputMath{239}{\cap}
61.228 \DeclareInputMath{240}{\equiv}
61.229 \DeclareInputMath{241}{\pm}
61.230\DeclareInputMath{242}{\ge}
61.231 \DeclareInputMath{243}{\le}
61.232 \DeclareInputMath{246}{\div}
61.233\DeclareInputMath{247}{\approx}
61.234 \DeclareInputText{248}{\textdegree}
61.235 \DeclareInputText{249}{\textperiodcentered}
61.236 \DeclareInputText{250}{\textbullet}
61.237 \DeclareInputMath{251}{\surd}
61.238\DeclareInputMath{252}{\mathnsuperior}
```

```
61.239 \DeclareInputMath{253}{\mathtwosuperior}
61.240 \DeclareInputText{254}{\textblacksquare}
61.241 \DeclareInputText{255}{\nobreakspace}
61.242 〈/cp862\rangle
```

$\backslash$ 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．

```
61.243 \langle*8859 - 8\rangle
61.244 \newcommand{\DisableNikud}{%
61.245 \DeclareInputText{192}{}%
61.246 \DeclareInputText{193}{}%
61.247 \DeclareInputText{194}{}%
61.248 \DeclareInputText{195}{}%
61.249 \DeclareInputText{196}{}%
61.250 \DeclareInputText{197}{}%
61.251 \DeclareInputText{198}{}%
61.252 \DeclareInputText{199}{}%
61.253 \DeclareInputText{200}{}%
61.254 \DeclareInputText{201}{}%
61.255 \DeclareInputText{203}{}%
61.256 \DeclareInputText{204}{}%
61.257 \DeclareInputText{205}{}%
61.258 \DeclareInputText{206}{}%
61.259 \DeclareInputText{207}{}%
61.260 \DeclareInputText{208}{}%
61.261 \DeclareInputText{209}{}%
61.262 \DeclareInputText{210}{}%
61.263 \DeclareInputText{211}{}%
61.264 \DeclareInputText{212}{}%
61.265 \DeclareInputText{213}{}%
61.266 \DeclareInputText{214}{}%
61.267 }
61.268 〈/8859-8\rangle
```

Finally，we reset the category code of the＠sign at the end of all ．def files． 61.269 〈－driver〉\makeatother

## 62 Hebrew font encodings

Don＇t forget to update the docs．．．

## 62．1 THIS SECTION IS OUT OF DATE．UPDATE DOCS TO MATCH HE8 ENCODING

The file hebrew．fdd ${ }^{73}$ contains the Local Hebrew Encoding（LHE）definition，the external font information needed to use the Hebrew 7－bit fonts（old code fonts）

[^63]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 then for any other encoding.
2. Available $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ Hebrew fonts do not include latin alphabet, and we can safely map Hebrew glyphs to the ASCII positions ( $0-127$ ).

Current definition of the LHE encoding supports only Hebrew letters ( $\backslash$ hebalef\hebtav), but not Hebrew points, such as \hebdagesh, \hebqamats, \hebpatah, \hebshindot, etc. We are working now on such addition.

### 62.2 The DOCSTRIP modules

The following modules are used in the implementation to direct DOCSTRIP in generating external files:
driver
HE8enc
HE8cmr
HE8cmss
HE8cmtt
HE8OmegaHebrew
HE8aharoni
HE8david
HE8drugulin
HE8ellinia
HE8frankruehl
HE8KtavYad
HE8MiriamMono
HE8Nachlieli
HE8CourierShalom
HE8HelveticaNarkissTam
HE8TimesNarkissim
HE8mfdavid
HE8mffrank
HE8mffrankthick
HE8mffrankthin
HE8mfmiriam
HE8mfmiriamwide
HE8mfnarkistam
LHEenc
LHEcmr
LHEcmss
LHEcmtt
LHEclas
LHEshold
LHEshscr
LHEshstk
LHEfr
LHEcrml
LHEredis
nowarn
hebfont
produce a documentation driver file
produce the encoding definition for CodePage 1255 (HE8)
make Hebrew default font in HE8
make Hebrew sans-serif font in HE8
make Hebrew typewriter font in HE8
Hebrew font from the Omega project (by ???)
Hebrew sans-serif font (Culmus)
Hebrew serif font (Culmus)
Hebrew old serif font (Culmus)
Hebrew isans-serif font (Culmus)
Hebrew serif font (Culmus)
Hebrew handwriting font (Culmus)
Hebrew monospaced font
Hebrew sans-serif font (Culmus)
Hebrew Shalom (Courier) font (by IBM)
Hebrew NarkisTam (Helvetica) (by Zvi Narkis)
Hebrew Narkissim (Times) (by Zvi Narkis)
Hebrew David font (by ???)
Hebrew Frank-Ruehl font (by ??)
Hebrew Frank-Ruehl (thick) font (by ??)
Hebrew Frank-Ruehl (thin) font (by ??)
Hebrew Miriam font (by ???)
Hebrew Miriam (wide) font (by ???)
Hebrew Narkis Tam font (by ???)
produce the encoding definition for Local Hebrew Encoding (LHE)
make Hebrew default font in LHE
make Hebrew sans-serif font in LHE
make Hebrew typewriter font in LHE
make Hebrew classic font (by Joel M. Hoffman) in LHE
make Hebrew shalom old font (by Jonathan Brecher) in LHE make Hebrew shalom script font (by Jonathan Brecher) in LHE make Hebrew shalom stick font (by Jonathan Brecher) in LHE make Hebrew frank-ruehl font in LHE
make Hebrew carmel font (by Dr. Samy Zafrany) in LHE make Hebrew redis font (by Prof. Jacques J. Goldberg) in LHE option for font definition files, that used to produce "silent" font substitutions without giving warnings 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\}\}

## 62．3 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ 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

```
62.1 〈*LHEenc \(\rangle\)
62.2 DeclareFontEncoding\{LHE \(\}\) \{\} \{\}
\(62.3 \backslash\) DeclareFontSubstitution\{LHE \(\}\) \{cmr \(\}\{\mathrm{m}\}\{\mathrm{n}\}\)
62.4 \DeclareErrorFont \(\{\mathrm{LHE}\}\{\mathrm{cmr}\}\{\mathrm{m}\}\{\mathrm{n}\}\{10\}\)
62.5 (/LHEenc)
```

Then we define a few commands in the LHE encoding．

```
62.6 <*LHEenc>
62.7\ProvideTextCommand{\textcopyright}{LHE}{\textcircled{\@latin{c}}}
62.8\ProvideTextCommand{\textregistered}{LHE}{\textcircled{\scshape%
6 2 . 9 ~ \ @ l a t i n \{ r \} \} \}
62.10\ProvideTextCommand{\texttrademark}{LHE}{\textsuperscript{\@latin{TM}}}
62.11 </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 $\mathrm{IA}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ 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 glyphes 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．

```
62.12〈+LHEenc\rangle\lccode ''='`
```

Hebrew letters occupy the positions 96－122 in LHE encoding：

```
62.13〈*LHEenc\rangle
62.14\DeclareTextSymbol{\hebalef}{LHE}{96}
62.15\DeclareTextSymbol{\hebbet}{LHE}{97}
62.16 \DeclareTextSymbol{\hebgimel}{LHE}{98}
62.17\DeclareTextSymbol{\hebdalet}{LHE}{99}
62.18 \DeclareTextSymbol{\hebhe}{LHE}{100}
62.19 \DeclareTextSymbol{\hebvav}{LHE}{101}
62.20\DeclareTextSymbol{\hebzayin}{LHE}{102}
62.21 \DeclareTextSymbol{\hebhet}{LHE}{103}
```

```
62.22\DeclareTextSymbol{\hebtet}{LHE}{104}
62.23\DeclareTextSymbol{\hebyod}{LHE}{105}
62.24\DeclareTextSymbol{\hebfinalkaf}{LHE}{106}
62.25\DeclareTextSymbol{\hebkaf}{LHE}{107}
62.26\DeclareTextSymbol{\heblamed}{LHE}{108}
62.27\DeclareTextSymbol{\hebfinalmem}{LHE}{109}
62.28\DeclareTextSymbol{\hebmem}{LHE}{110}
62.29\DeclareTextSymbol{\hebfinalnun}{LHE}{111}
62.30\DeclareTextSymbol{\hebnun}{LHE}{112}
62.31 \DeclareTextSymbol{\hebsamekh}{LHE}{113}
62.32\DeclareTextSymbol{\hebayin}{LHE}{114}
62.33\DeclareTextSymbol{\hebfinalpe}{LHE}{115}
62.34 \DeclareTextSymbol{\hebpe}{LHE}{116}
62.35\DeclareTextSymbol{\hebfinaltsadi}{LHE}{117}
62.36 \DeclareTextSymbol{\hebtsadi}{LHE}{118}
62.37\DeclareTextSymbol{\hebqof}{LHE}{119}
62.38 \DeclareTextSymbol{\hebresh}{LHE}{120}
62.39 \DeclareTextSymbol{\hebshin}{LHE}{121}
62.40\DeclareTextSymbol{\hebtav}{LHE}{122}
62.41 〈/LHEenc>
```

Letter \hebsin is defined as a synonym of \hebshin:
$62.42\langle+$ LHEenc $\rangle \backslash$ let $\backslash$ hebsin= $\backslash$ hebshin

### 62.4 The font definition files (in LHE encoding)

### 62.4.1 Hebrew default font

It uses Jerusalem font for regular font, Old Jaffa font for italic shape and smallcaps, Dead Sea font for bold face, and Tel-Aviv for bold-italic

```
62.43 <*LHEcmr>
62.44 \DeclareFontFamily{LHE}{cmr}{\hyphenchar\font45}
62.45\DeclareFontShape{LHE}{cmr}{m}{n}
62.46 {<-> jerus10 }{}
62.47%%%%%%%% Italicized shape
62.48\DeclareFontShape{LHE}{cmr}{m}{it}
62.49 {<-> oldjaf10 }{}
62.50\DeclareFontShape{LHE}{cmr}{m}{sl}
62.51 {<-> oldjaf10 }{}
62.52\DeclareFontShape{LHE}{cmr}{m}{sc}
62.53 {<-> oldjaf10 }{}
62.54%%%%%%%% Bold extended series
62.55\DeclareFontShape{LHE}{cmr}{bx}{n}
62.56 {<-> deads10 }{}
62.57\DeclareFontShape{LHE}{cmr}{b}{n}
62.58 {<-> deads10 }{}
62.59%%%%%%% Bold extended (Italic) series
62.60\DeclareFontShape{LHE}{cmr}{bx}{sl}
62.61 {<-> telav10 }{}
62.62\DeclareFontShape{LHE}{cmr}{bx}{it}
```

```
62.63 {<-> telav10 }{}
```

62.64 〈／LHEcmr〉

## 62．4．2 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．

```
62.65 <*LHEcmss>
62.66\DeclareFontFamily{LHE}{cmss}{\hyphenchar\font45}
62.67 \DeclareFontShape{LHE}{cmss}{m}{n}
62.68 {<-> telav10 }{}
62.69%%%%%%%% Font/shape undefined, therefore substituted
62.70\DeclareFontShape{LHE}{cmss}{m}{sc}
62.71 <-nowarn\rangle {<->sub * cmss/m/n}{}
62.72 <+nowarn\rangle {<->ssub * cmss/m/n}{}
62.73%%%%%%%% Italicized shape
62.74\DeclareFontShape{LHE}{cmss}{m}{it}
62.75 {<-> oldjaf10 }{}
62.76%%%%%%%% Font/shape undefined, therefore substituted
62.77\DeclareFontShape{LHE}{cmss}{m}{sl}
62.78 <-nowarn\rangle {<->sub * cmss/m/it}{}
62.79 <+nowarn\rangle {<->ssub * cmss/m/it}{}
62.80 %%%%%%%% Bold extended series
62.81 \DeclareFontShape{LHE}{cmss}{bx}{n}
62.82 {<-> deads10 }{}
62.83%%%%%%%% Font/shape undefined, therefore substituted
62.84 \DeclareFontShape{LHE}{cmss}{b}{n}
62.85 <-nowarn\rangle {<->sub * cmss/bx/n}{}
62.86 <+nowarn\rangle {<->ssub * cmss/bx/n}{}
62.87%%%%%%%% Font/shape undefined, therefore substituted
62.88 \DeclareFontShape{LHE}{cmss}{bx}{sl}
62.89 <-nowarn\rangle {<->sub * cmss/bx/n}{}
62.90 <+nowarn\rangle {<->ssub * cmss/bx/n}{}
62.91%%%%%%%% Font/shape undefined, therefore substituted
62.92\DeclareFontShape{LHE}{cmss}{bx}{it}
62.93 <-nowarn\rangle {<->sub * cmss/bx/n}{}
62.94 〈+nowarn\rangle {<->ssub * cmss/bx/n}{}
62.95 </LHEcmss>
```


## 62．4．3 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．

```
62.96 <*LHEcmtt\rangle
62.97\DeclareFontFamily{LHE}{cmtt}{\hyphenchar \font\m@ne}
62.98\DeclareFontShape{LHE}{cmtt}{m}{n}
62.99 {<-> telav10 }{}
62.100%%%%%%%% Font/shape undefined, therefore substituted
62.101 \DeclareFontShape{LHE}{cmtt}{m}{sc}
```

```
62.102 <-nowarn\rangle {<->sub * cmtt/m/n}{}
62.103 \langle+nowarn\rangle {<->ssub * cmtt/m/n}{}
62.104 %%%%%%% Italicized shape
62.105\DeclareFontShape{LHE}{cmtt}{m}{it}
62.106 {<-> oldjaf10 }{}
62.107%%%%%%%% Font/shape undefined, therefore substituted
62.108 \DeclareFontShape{LHE}{cmtt}{m}{sl}
62.109 <-nowarn\rangle {<->sub * cmtt/m/it}{}
62.110 <+nowarn\rangle {<->ssub * cmtt/m/it}{}
62.111%%%%%%% Bold extended series
62.112 \DeclareFontShape{LHE}{cmtt}{bx}{n}
62.113 {<-> deads10 }{}
62.114%%%%%%%% Font/shape undefined, therefore substituted
62.115 \DeclareFontShape{LHE}{cmtt}{bx}{it}
62.116 <-nowarn\rangle {<->sub * cmtt/bx/n}{}
62.117 <+nowarn\rangle {<->ssub * cmtt/bx/n}{}
62.118 </LHEcmtt\rangle
```


### 62.4.4 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 $\mathrm{HT}_{\mathrm{EX}} 2_{\varepsilon}$, however one can use vowels and dots directly with Plain $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ macros. We are working on generic vowels and dots support for $\operatorname{LAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$.

```
62.119 <*LHEclas>
62.120 \DeclareFontFamily{LHE}{clas}{}
62.121 \DeclareFontShape{LHE}{clas}{m}{n}
62.122 {<-> s * [0.83345] hclassic }{}
62.123%%%%%%%% Font/shape undefined, therefore substituted
62.124 \DeclareFontShape{LHE}{clas}{m}{sc}
62.125 <-nowarn\rangle {<->sub * clas/m/n}{}
62.126 \langle+nowarn\rangle {<->ssub * clas/m/n}{}
62.127%%%%%%% Slanted shape
62.128 \DeclareFontShape{LHE}{clas}{m}{sl}
62.129 {<-> s * [0.69389] hcaption }{}
62.130%%%%%%%% Font/shape undefined, therefore substituted
62.131 \DeclareFontShape{LHE}{clas}{m}{it}
62.132 \langle-nowarn\rangle {<->sub * clas/m/sl}{}
```

```
62.133 <+nowarn\rangle {<->ssub * clas/m/sl}{}
62.134 </LHEclas>
```


## 62．4．5 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 $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ or $\mathrm{LAT}_{\mathrm{E}} \mathrm{X}$ ，with as well as without vowels．

Currently，only the Hebrew consonants（hebalef－hebtav）from these fonts are supported by $\mathrm{EAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ ，however one can use vowels and dots directly with Plain TEX macros．We are working on generic vowels and dots support for $\mathrm{LAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ ．

```
62.135 <*LHEshold\rangle
```

62.136 \DeclareFontFamily\{LHE\}\{shold\} \{\}
62.137 \DeclareFontShape\{LHE\}\{shold\}\{m\}\{n\}
62.138 \{<-> shold10 \}\{\}
62.139 〈/LHEshold〉
62.140 〈*LHEshscr〉
62.141 \DeclareFontFamily\{LHE\}\{shscr\}\{\}
62.142 \DeclareFontShape\{LHE\}\{shscr\} $\{\mathrm{m}\}\{\mathrm{n}\}$
$62.143 \quad\{\langle-\rangle$ shscr10 $\}\}$
62.144 〈/LHEshscr〉
62.145 〈*LHEshstk〉
62.146 \DeclareFontFamily\{LHE\}\{shstk\} \{\}
62.147 \DeclareFontShape\{LHE\}\{shstk\}\{m\}\{n\}
$62.148 \quad\{\langle->$ shstk10 $\}\}$
62.149 〈/LHEshstk〉

## 62．4．6 Hebrew frank－ruehl font

Frank Ruehl font was written in METAFONT and includes three shapes：regular， bold extaneded and slanted．

```
62.150 <*LHEfr)
62.151 \DeclareFontFamily{LHE}{fr}{}
62.152\DeclareFontShape{LHE}{fr}{m}{n}
```

```
62.153 {<-> fr }{}
62.154%%%%%%%% Font/shape undefined, therefore substituted
62.155\DeclareFontShape{LHE}{fr}{m}{sc}
62.156 <-nowarn\rangle {<->sub * fr/m/n}{}
62.157 <+nowarn\rangle {<->ssub * fr/m/n}{}
62.158%%%%%%%% Slanted shape
62.159 \DeclareFontShape{LHE}{fr}{m}{sl}
62.160 {<-> frsl }{}
62.161%%%%%%%% Font/shape undefined, therefore substituted
62.162\DeclareFontShape{LHE}{fr}{m}{it}
62.163 <-nowarn\rangle {<->sub * fr/m/sl}{}
62.164 <+nowarn\rangle {<->ssub * fr/m/sl}{}
62.165%%%%%%%% Bold extended series
62.166 \DeclareFontShape{LHE}{fr}{bx}{n}
62.167 {<-> frbx }{}
62.168%%%%%%%% Font/shape undefined, therefore substituted
62.169 \DeclareFontShape{LHE}{fr}{b}{n}
62.170 <-nowarn\rangle {<->sub * fr/bx/n}{}
62.171 <+nowarn\rangle {<->ssub * fr/bx/n}{}
62.172%%%%%%%% Font/shape undefined, therefore substituted
62.173\DeclareFontShape{LHE}{fr}{bx}{sl}
62.174 <-nowarn\rangle {<->sub * fr/bx/n}{}
62.175 <+nowarn\rangle {<->ssub * fr/bx/n}{}
62.176%%%%%%%% Font/shape undefined, therefore substituted
62.177 \DeclareFontShape{LHE}{fr}{bx}{it}
62.178 <-nowarn\rangle {<->sub * fr/bx/n}{}
62.179 <+nowarn\rangle {<->ssub * fr/bx/n}{}
62.180 </LHEfr>
```


### 62.4.7 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.

```
62.181 〈*LHEcrml\rangle
62.182\DeclareFontFamily{LHE}{crml}{}
62.183\DeclareFontShape{LHE}{crml}{m}{n}
62.184 {<-> crml10 }{}
62.185%%%%%%%% Font/shape undefined, therefore substituted
62.186 \DeclareFontShape{LHE}{crml}{m}{sc}
62.187 <-nowarn\rangle {<->sub * crml/m/n}{}
62.188 <+nowarn\rangle {<->ssub * crml/m/n}{}
62.189%%%%%%%% Slanted shape
62.190\DeclareFontShape{LHE}{crml}{m}{sl}
62.191 {<-> crmlsl10 }{}
62.192%%%%%%%% Font/shape undefined, therefore substituted
62.193 \DeclareFontShape{LHE}{crml}{m}{it}
62.194 <-nowarn\rangle {<->sub * crml/m/sl}{}
62.195 <+nowarn\rangle {<->ssub * crml/m/sl}{}
```

```
62.196 〈/LHEcrmI\
```


### 62.4.8 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 extanded shapes. This font contains a full set of Hebrew letters in a "sans-serif vectorized" style, and selected punctuation.

```
62.197 〈*LHEredis\rangle
62.198 \DeclareFontFamily{LHE}{redis}{}
62.199 \DeclareFontShape{LHE}{redis}{m}{n}{%
62.200 <5> <6> redis7
62.201 <7> <8> <9> <10> <12> gen * redis
62.202 <10.95> redis10
62.203 <14.4> redis12
62.204 <17.28> <20.74> <24.88> redis17}{}
62.205%%%%%%%% Font/shape undefined, therefore substituted
62.206 \DeclareFontShape{LHE}{redis}{m}{sc}
62.207 <-nowarn\rangle {<->sub * redis/m/n}{}
62.208 <+nowarn\rangle {<->ssub * redis/m/n}{}
62.209%%%%%%%% Slanted shape
62.210\DeclareFontShape{LHE}{redis}{m}{sl}{%
62.211 <5> <6> <7> rediss8
62.212 <8> <9> <10> <12> gen * rediss
62.213 <10.95> rediss10
62.214 <14.4> <17.28> <20.74> <24.88> rediss12}{}
62.215%%%%%%% Font/shape undefined, therefore substituted
62.216 \DeclareFontShape{LHE}{redis}{m}{it}
62.217 <-nowarn\rangle {<->sub * redis/m/sl}{}
62.218 <+nowarn\rangle {<->ssub * redis/m/sl}{}
62.219 %%%%%%%% Bold extended series
62.220\DeclareFontShape{LHE}{redis}{bx}{n}{%
62.221 <5> <6> <7> <8> <9> <10> <10.95> <12>
62.222 <14.4> <17.28> <20.74> <24.88> redisb10}{}
62.223%%%%%%%% Font/shape undefined, therefore substituted
62.224 \DeclareFontShape{LHE}{redis}{b}{n}
62.225 <-nowarn\rangle {<->sub * redis/bx/n}{}
62.226 <+nowarn\rangle {<->ssub * redis/bx/n}{}
62.227 %%%%%%%% Font/shape undefined, therefore substituted
62.228 \DeclareFontShape{LHE}{redis}{bx}{sl}
62.229 <-nowarn\rangle {<->sub * redis/bx/n}{}
62.230 <+nowarn\rangle {<->ssub * redis/bx/n}{}
62.231%%%%%%%% Font/shape undefined, therefore substituted
62.232 \DeclareFontShape{LHE}{redis}{bx}{it}
62.233 <-nowarn\rangle {<->sub * redis/bx/n}{}
62.234 <+nowarn\rangle {<->ssub * redis/bx/n}{}
62.235 </LHEredis>
```


### 62.5 The HE8encoding definition file

The Hebrew font encoding HE8 is based upon an extention by Microsoft to the ISO-8859-8 standard. This is an 8bit encoding. The extentions 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.

```
62.236 <*HE8enc>
62.237 \DeclareFontEncoding{HE8}{}{}
62.238\DeclareFontSubstitution{HE8}{cmr}{m}{n}
62.239 \DeclareErrorFont{HE8}{cmr}{m}{n}{10}
62.240 </HE8enc\rangle
```

Then we define a few commands in the HE8 encoding.

```
62.241 〈*HE8enc\rangle
62.242\ProvideTextCommand{\textcopyright}{HE8}{\textcircled{\@latin{c}}}
62.243 \ProvideTextCommand{\textregistered}{HE8}{\textcircled{\scshape%
6 2 . 2 4 4 ~ \ @ l a t i n \{ r \} \} \}
62.245\ProvideTextCommand{\texttrademark}{HE8}{\textsuperscript{\@latin{TM}}}
62.246 </HE8enc>
```


### 62.5.1 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 $\mathrm{HAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}$ 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 glyphes 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:

```
62.247 <*HE8enc\rangle
62.248 \ProvideTextCommand{\textcopyright}{HE8}{\textcircled{\@latin{c}}}
62.249 \ProvideTextCommand{\textregistered}{HE8}{\textcircled{\scshape%
6 2 . 2 5 0 ~ \ @ l a t i n \{ r \} \} \}
62.251 \ProvideTextCommand{\texttrademark}{HE8}{\textsuperscript{\@latin{TM}}}
62.252 </HE8enc>
The hebrew points:
```

```
62.253 <*HE8enc\rangle
```

62.253 <*HE8enc\rangle
62.254 \DeclareTextSymbol{\sheva}{HE8}{192}
62.255 \DeclareTextSymbol{\hatafsegol}{HE8}{193}
62.256 \DeclareTextSymbol{\hatafpatah}{HE8}{194}
62.257 \DeclareTextSymbol{\hatafqamats}{HE8}{195}
62.258\DeclareTextSymbol{\hiriq}{HE8}{196}

```
```

62.259 \DeclareTextSymbol{\tsere}{HE8}{197}
62.260 \DeclareTextSymbol{\segol}{HE8}{198}
62.261 \DeclareTextSymbol{\patah}{HE8}{199}
62.262 \DeclareTextSymbol{\qamats}{HE8}{200}
62.263 \DeclareTextSymbol{\holam}{HE8}{201}
62.264 \DeclareTextSymbol{\qubuts}{HE8}{203}
62.265 \DeclareTextSymbol{\dagesh}{HE8}{204}
62.266 \DeclareTextSymbol{\meteg}{HE8}{205}
62.267 \DeclareTextSymbol{\maqaf}{HE8}{206}
62.268 \DeclareTextSymbol{\rafe}{HE8}{207}
62.269 \DeclareTextSymbol{\paseq}{HE8}{208}
62.270 \DeclareTextSymbol{\shindot}{HE8}{209}
62.271 \DeclareTextSymbol{\sindot}{HE8}{210}
62.272 \DeclareTextSymbol{\sofpasuq}{HE8}{211}
62.273\DeclareTextSymbol{\doublevav}{HE8}{212}
62.274 \DeclareTextSymbol{\vavyod} {HE8}{213}
62.275\DeclareTextSymbol{\doubleyod}{HE8}{214}
62.276 </HE8enc)

```

Hebrew letters occupy the positions 224-250 in HE8 encoding [WHAT ABOUT OTHER MARKS]:
```

6 2 . 2 7 7 ~ \langle * H E 8 e n c )

```
\(62.278 \%\) \lccode'‘='، \% probably not needed (Tzafrir)
62.279 \DeclareTextSymbol\{\hebalef\}\{HE8\}\{224\}
62.280 \DeclareTextSymbol\{\hebbet\}\{HE8\}\{225\}
62.281 \DeclareTextSymbol\{\hebgimel\}\{HE8\}\{226\}
62.282 \DeclareTextSymbol\{\hebdalet\}\{HE8\}\{227\}
62.283 \DeclareTextSymbol\{ \(\backslash\) hebhe \(\}\) \{HE8\}\{228\}
62.284 \DeclareTextSymbol\{\hebvav\} \{HE8\} \{229\}
\(62.285 \backslash\) DeclareTextSymbol\{\hebzayin\}\{HE8\}\{230\}
62.286 \DeclareTextSymbol\{\hebhet\}\{HE8\}\{231\}
62.287 \DeclareTextSymbol\{\hebtet\} \{HE8\} \{232\}
62.288 \DeclareTextSymbol\{\hebyod\} \{HE8\} \{233\}
62.289 \DeclareTextSymbol\{\hebfinalkaf\}\{HE8\}\{234\}
62.290 \DeclareTextSymbol\{\hebkaf\}\{HE8\}\{235\}
62.291 \DeclareTextSymbol\{\heblamed\} \{HE8\}\{236\}
62.292 \DeclareTextSymbol\{\hebfinalmem\}\{HE8\}\{237\}
62.293 \DeclareTextSymbol\{\hebmem\} \{HE8\}\{238\}
\(62.294 \backslash\) DeclareTextSymbol\{\hebfinalnun\}\{HE8\}\{239\}
62.295 \DeclareTextSymbol\{\hebnun\} \{HE8\}\{240\}
62.296 \DeclareTextSymbol\{\hebsamekh\}\{HE8\}\{241\}
62.297 \DeclareTextSymbol\{\hebayin\}\{HE8\}\{242\}
62.298 \DeclareTextSymbol\{\hebfinalpe\}\{HE8\}\{243\}
62.299 \DeclareTextSymbol \{ \hebpe\} \{HE8\} \{244\}
\(62.300 \backslash\) DeclareTextSymbol\{\hebfinaltsadi\}\{HE8\}\{245\}
62.301 \DeclareTextSymbol\{\hebtsadi\}\{HE8\}\{246\}
62.302 \DeclareTextSymbol\{\hebqof \(\}\) \{HE8\} \{247\}
62.303 \DeclareTextSymbol\{\hebresh\}\{HE8\}\{248\}
62.304 \DeclareTextSymbol\{\hebshin\}\{HE8\}\{249\}
62.305 \DeclareTextSymbol\{\hebtav\} \{HE8\}\{250\}
```

62.306 </HE8enc>

```

Letter \hebsin is defined as a synonym of \hebshin:
```

62.307 \langle+HE8enc\rangle\let\hebsin=\hebshin

```

\subsection*{62.6 The font definition files (in HE8 encoding)}

\subsection*{62.6.1 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
```

62.308 <*HE8cmr>
62.309\DeclareFontFamily{HE8}{cmr}{\hyphenchar\font45}
62.310 \DeclareFontShape{HE8}{cmr}{m}{n}
62.311 {<-> david }{}
62.312%%%%%%%% Italicized shape
62.313 \DeclareFontShape{HE8}{cmr} {m}{it}
62.314 {<-> davidi }{}
62.315 \DeclareFontShape{HE8}{cmr}{m}{sl}
62.316 {<-> davidi }{}
62.317\DeclareFontShape{HE8}{cmr}{m}{sc}
62.318 {<-> david }{}
62.319%%%%%%%% Bold extended series
62.320 \DeclareFontShape{HE8}{cmr}{bx}{n}
62.321 {<-> davidb }{}
62.322 \DeclareFontShape{HE8}{cmr}{b}{n}
62.323 {<-> davidb }{}
62.324%%%%%%%% Bold extended (Italic) series
62.325 \DeclareFontShape{HE8}{cmr}{bx}{sl}
62.326 {<-> davidbi }{}
62.327 \DeclareFontShape{HE8}{cmr}{bx}{it}
62.328 {<-> davidbi }{}
62.329 〈/HE8cmr>

```

\subsection*{62.6.2 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 definitons above.
```

62.330 〈*HE8cmss>
62.331 \DeclareFontFamily{HE8}{cmss}{\hyphenchar\font45}
62.332\DeclareFontShape{HE8}{cmss}{m}{n}
62.333 {<-> nachlieli }{}
62.334%%%%%%% Italicized shape
62.335\DeclareFontShape{HE8}{cmss}{m}{it}
62.336 {<-> nachlieli }{}
62.337\DeclareFontShape{HE8}{cmss}{m}{sl}
62.338 {<-> nachlieli }{}
62.339 \DeclareFontShape{HE8}{cmss}{m}{sc}
62.340 {<-> nachlieli }{}

```
```

62.341 %%%%%%%% Bold extended series
62.342 \DeclareFontShape{HE8}{cmss}{bx}{n}
62.343 {<-> nachlieli }{}
62.344 \DeclareFontShape{HE8}{cmss}{b}{n}
62.345 {<-> nachlieli }{}
62.346 %%%%%%%% Bold extended (Italic) series
62.347 \DeclareFontShape{HE8}{cmss}{bx}{sl}
62.348 {<-> nachlieli }{}
62.349 \DeclareFontShape{HE8}{cmss}{bx}{it}
62.350 {<-> nachlieli }{}
62.351 〈/HE8cmss\rangle

```

\section*{62．6．3 Hebrew typewriter font}

Until we have a real sans－serif font in this distribution，this file will remain a copy of the roman fonts definitons above．
```

62.352 〈*HE8cmtt\rangle
62.353\DeclareFontFamily{HE8}{cmtt}{\hyphenchar\font45}
62.354 \DeclareFontShape{HE8}{cmtt}{m}{n}
62.355 {<-> miriam }{}
62.356%%%%%%%% Italicized shape
62.357 \DeclareFontShape{HE8}{cmtt}{m}{it}
62.358 {<-> miriam }{}
62.359 \DeclareFontShape{HE8}{cmtt}{m}{sl}
62.360 {<-> miriam }{}
62.361 \DeclareFontShape{HE8}{cmtt}{m}{sc}
62.362 {<-> miriam }{}
62.363%%%%%%% Bold extended series
62.364 \DeclareFontShape{HE8}{cmtt}{bx}{n}
62.365 {<-> miriam }{}
62.366 \DeclareFontShape{HE8}{cmtt}{b}{n}
62.367 {<-> miriam }{}
62.368%%%%%%%% Bold extended (Italic) series
62.369 \DeclareFontShape{HE8}{cmtt}{bx}{sl}
62.370 {<-> miriam }{}
62.371 \DeclareFontShape{HE8}{cmtt}{bx}{it}
62.372 {<-> miriam }{}
62.373 〈/HE8cmtt\rangle

```

\section*{62．6．4 8Bit OmegaHebrew font}

OmegaHebrew is a serif hebrew font created by the omega project［FILL IN CRED－
ITS］［FILL IN GENERAL SHAPE DESCRIPTION］shapes：［FILL IN］
\(62.374\langle * \mathrm{HE} 8\) OmegaHebrew \(\rangle\)
62．375 \def \OmegaHebrewscale\｛0．9\}
62.376 \DeclareFontFamily\｛HE8\}\{OmegaHebrew\}\{\hyphenchar \(\backslash\) font45\}
62.377 \DeclareFontShape\｛HE8\}\{OmegaHebrew\}\{m\}\{n\}\{<-> [\OmegaHebrewscale] OmegaHebrew \}\{\}
62.378 \％\endinput \％is it needed［tzafrir］
62.379 〈／HE8OmegaHebrew〉

\subsection*{62.6.5 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]
```

62.380 <*HE8aharoni\rangle
62.381 \def\Aharoniscale{1.0}
62.382 \DeclareFontFamily{HE8}{aharoni}{\hyphenchar\font45}
62.383\DeclareFontShape{HE8}{aharoni}{m}{n} {<-> [\Aharoniscale] aharoni}{}
62.384 \DeclareFontShape{HE8}{aharoni}{m}{it} {<-> [\Aharoniscale] aharonii}{}
62.385 \DeclareFontShape{HE8}{aharoni}{m}{sl} {<-> [\Aharoniscale] aharonii}{}
62.386 \DeclareFontShape{HE8}{aharoni}{b}{n} {<-> [\Aharoniscale] aharonib}{}
62.387\DeclareFontShape{HE8}{aharoni}{bx}{n} {<-> [\Aharoniscale] aharonib}{}
62.388\DeclareFontShape{HE8}{aharoni}{bx}{it} {<-> [\Aharoniscale] aharonibi}{}
6 2 . 3 8 9
62.390 %\endinput % is it needed [tzafrir]
62.391 </HE8aharoni\rangle

```

\subsection*{62.6.6 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]
```

62.392 \langle*HE8david\rangle
62.393\def\Davidscale{1.0}
62.394 \DeclareFontFamily{HE8}{david}{\hyphenchar\font45}
6 2 . 3 9 5
62.396\DeclareFontShape{HE8}{david}{m}{n} {<-> [\Davidscale] david}{}
62.397 \DeclareFontShape{HE8}{david}{m}{it} {<-> [\Davidscale] davidi}{}
62.398 \DeclareFontShape{HE8}{david}{m}{sl} {<-> [\Davidscale] davidi}{}
62.399 \DeclareFontShape{HE8}{david}{b}{n} {<-> [\Davidscale] davidb}{}
62.400 \DeclareFontShape{HE8}{david}{bx}{n} {<-> [\Davidscale] davidb}{}
62.401 \DeclareFontShape{HE8}{david}{bx}{it} {<-> [\Davidscale] davidbi}{}
6 2 . 4 0 2
6 2 . 4 0 3
62.404 %\endinput % is it needed [tzafrir]
62.405 </HE8david\rangle

```

\subsection*{62.6.7 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]
```

62.406 \langle*HE8drugulin
62.407 \def\Drugulinscale{1.0}
62.408\DeclareFontFamily{HE8}{drugulin}{\hyphenchar\font45}
62.409 \DeclareFontShape{HE8}{drugulin}{m}{n} {<-> [\Drugulinscale] drugulinb}{}
62.410 \DeclareFontShape{HE8}{drugulin}{m}{it} {<-> [\Drugulinscale] drugulinbi}{}
62.411 \DeclareFontShape{HE8}{drugulin}{m}{sl} {<-> [\Drugulinscale] drugulinbi}{}
62.412 \DeclareFontShape{HE8}{drugulin}{b}{n} {<-> [\Drugulinscale] drugulinb}{}
62.413\DeclareFontShape{HE8}{drugulin}{bx}{n} {<-> [\Drugulinscale] drugulinb}{}

```
```

62.414 \DeclareFontShape{HE8}{drugulin}{bx}{it} {<-> [\Drugulinscale] drugulinbi}{}
62.415 %\endinput % is it needed [tzafrir]
62.416 〈/HE8drugulin\

```

\section*{62．6．8 8Bit Ellinia font}

Ellinia is a sans－serif hebrew font created by the omega project［FILL IN CRED－ ITS］［FILL IN GENERAL SHAPE DESCRIPTION］shapes：［FILL IN］
```

62.417 <*HE8ellinia\rangle
62.418\def\Elliniascale{1.0}
62.419 \DeclareFontFamily{HE8}{ellinia}{\hyphenchar\font45}
62.420 \DeclareFontShape{HE8}{ellinia}{m}{n} {<-> [\Elliniascale] ellinia}{}
62.421 \DeclareFontShape{HE8}{ellinia}{m}{it} {<-> [\Elliniascale] elliniai}{}
62.422 \DeclareFontShape{HE8}{ellinia}{m}{sl} {<-> [\Elliniascale] elliniai}{}
62.423 \DeclareFontShape{HE8}{ellinia}{b}{n} {<-> [\Elliniascale] elliniab}{}
62.424 \DeclareFontShape{HE8}{ellinia}{bx}{n} {<-> [\Elliniascale] elliniab}{}
62.425 \DeclareFontShape{HE8}{ellinia}{bx}{it} {<-> [\Elliniascale] elliniabi}{}
62.426 %\endinput % is it needed [tzafrir]
62.427 〈/HE8ellinia\

```

\section*{62．6．9 8Bit FrankRuehl font}

FrankRuehl is a serif hebrew font created by the omega project［FILL IN CRED－
ITS］［FILL IN GENERAL SHAPE DESCRIPTION］shapes：［FILL IN］
```

62.428 <*HE8frankruehl\rangle
62.429 \def\FrankRuehlscale{1.0}
62.430 \DeclareFontFamily{HE8}{frank}{\hyphenchar\font45}
62.431 \DeclareFontShape{HE8}{frank}{m}{n} {<-> [\FrankRuehlscale] frank}{}
62.432\DeclareFontShape{HE8}{frank}{m}{it} {<-> [\FrankRuehlscale] franki}{}
62.433\DeclareFontShape{HE8}{frank}{m}{sl} {<-> [\FrankRuehlscale] franki}{}
62.434\DeclareFontShape{HE8}{frank}{b}{n} {<-> [\FrankRuehlscale] frankb}{}
62.435\DeclareFontShape{HE8}{frank}{bx}{n} {<-> [\FrankRuehlscale] frankb}{}
62.436 \DeclareFontShape{HE8}{frank}{bx}{it} {<-> [\FrankRuehlscale] frankbi}{}
62.437%\endinput % is it needed [tzafrir]
62.438 </HE8frankruehl\rangle

```

62．6．10 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］
```

62.439 〈*HE8yad\rangle
62.440 \def\KtavYadscale{1.0}
62.441 \DeclareFontFamily{HE8}{yad}{\hyphenchar\font45}
62.442 \DeclareFontShape{HE8}{yad}{m}{n} {<-> [\KtavYadscale] yadi}{}
62.443 \DeclareFontShape{HE8}{yad}{m}{it} {<-> [\KtavYadscale] yadi}{}
62.444 \DeclareFontShape{HE8}{yad}{m}{sl} {<-> [\KtavYadscale] yadi}{}
62.445 \DeclareFontShape{HE8}{yad}{b}{n} {<-> [\KtavYadscale] yadbi}{}
62.446 \DeclareFontShape{HE8}{yad}{bx}{n} {<-> [\KtavYadscale] yadbi}{}

```
```

62.447 \DeclareFontShape{HE8}{yad}{bx}{it} {<-> [\KtavYadscale] yadbi}{}
62.448 %\endinput % is it needed [tzafrir]
62.449 </HE8yad

```

\section*{62．6．11 8Bit MiriamMono font}

MiriamMono is a serif hebrew font created by the omega project［FILL IN CRED－ ITS］［FILL IN GENERAL SHAPE DESCRIPTION］shapes：［FILL IN］
\(62.450\langle * \mathrm{HE} 8\) miriam \(\rangle\)
62.451 \def \(\backslash\) MiriamMonoscale\{1.0\}
62.452 \DeclareFontFamily\{HE8\}\{miriam\}\{\hyphenchar \(\backslash\) font45\}
62.453 \DeclareFontShape\{HE8\}\{miriam\}\{m\}\{n\} \{<-> [\MiriamMonoscale] miriam\}\{\}
\(62.454 \backslash\) DeclareFontShape \(\{H E 8\}\{\) miriam \(\}\{m\}\{i t\} \quad\{<->\) [\MiriamMonoscale] miriami\} \(\}\)
62.455 \DeclareFontShape\{HE8\}\{miriam\}\{m\}\{sl\} \{<-> [\MiriamMonoscale] miriami\}\{\}
\(62.456 \backslash\) DeclareFontShape \(\{\mathrm{HE} 8\}\{\) miriam \(\}\) \{b\}\{n\} \(\{\langle-\rangle\) [ n MiriamMonoscale] miriamb\}\{\}
62.457 \DeclareFontShape\{HE8\}\{miriam\} \{bx\}\{n\} \{<-> [\MiriamMonoscale] miriamb\}\{\}
62.458 \DeclareFontShape\{HE8\}\{miriam\}\{bx\}\{it\} \{<-> [\MiriamMonoscale] miriambi\}\{\}
62.459
62.460 \% \endinput \% is it needed [tzafrir]
62.461 〈/HE8miriam〉

\section*{62．6．12 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］
```

62.462 \langle*HE8nachlieli\rangle
62.463\def\Nachlieliscale{1.0}
62.464 \DeclareFontFamily{HE8}{nachlieli}{\hyphenchar\font45}
62.465\DeclareFontShape{HE8}{nachlieli}{m}{n} {<-> [\Nachlieliscale] nachlieli}{}
62.466 \DeclareFontShape{HE8}{nachlieli}{m}{it} {<-> [\Nachlieliscale] nachlielii}{}
62.467 \DeclareFontShape{HE8}{nachlieli}{m}{sl} {<-> [\Nachlieliscale] nachlielii}{}
62.468 \DeclareFontShape{HE8}{nachlieli}{b}{n} {<-> [\Nachlieliscale] nachlielib}{}
62.469 \DeclareFontShape{HE8}{nachlieli}{bx}{n} {<-> [\Nachlieliscale] nachlielib}{}
62.470 \DeclareFontShape{HE8}{nachlieli}{bx}{it} {<-> [\Nachlieliscale] nachlielibi}{}
62.471 %\endinput % is it needed [tzafrir]
62.472 〈/HE8nachlieli\rangle

```

\section*{62．6．13 Hebrew font switching commands}

The hebfont package defines a number of high－level commands（all starting with \text．．similar to the standard \(\mathrm{AT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}\) font－change commands，for example \textbf）that have one argument and typeset this argument in the requested 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 \(\mathrm{HT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon\) document：
```

\sethebrew
\textclas{Hebrew text printed with Classic fonts}

```
\begin{tabular}{lll} 
Command & Corresponds to & Font family \\
\textjm\{..\} & \rmfamily & Jerusalem font \\
\textds\{..\} & \bfseries & Dead Sea font \\
\textoj\{..\} & \begin{tabular}{l} 
\itshape \\
\slshape
\end{tabular} & Old Jaffa font \\
& \emph & \\
\textta\{..\} & \sffamily & Tel-Aviv font \\
& \ttfamily & \\
\textcrml\{..\} & \fontfamily\{crml\} & Carmel fonts \\
\textfr\{..\} & \fontfamily\{fr\} & Frank-Ruehl fonts \\
\textredis\{..\} & \fontfamily\{redis\} & Redis fonts \\
\textclas\{..\} & \fontfamily\{redis\} & Classic fonts \\
\textshold\{..\} & \fontfamily\{shold\} & Shalom Old Style font \\
\textshscr\{..\} & \fontfamily\{shscr\} & Shalom Script font \\
\textshstk\{..\} & \fontfamily\{shstk\} & Shalom Stick font
\end{tabular}

Table 33: 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.
or to use Hebrew with Classic fonts locally:
```

\R{\textclas{Hebrew text printed with Classic fonts}}

```

We declare \(\operatorname{LAT}_{E} \mathrm{X} 2_{\varepsilon}\) font commands, e.g. \textjm\{...\} for all available fonts. Table 33 shows the meanings of all these new high-level commands.
\textjm Switches to Jerusalem font which is default regular Hebrew font ("roman" family). Commands \(\backslash\) textrm\{...\} and old-style \(\{\backslash\) rm \(\ldots\}\) will produce the same result.
62.473 〈*hebfont〉
62.474 \def \(\backslash i v r i t e x @ t m p\{H E 8\}\)
62.475 \ifx\ivritex@tmp\HeblatexEncoding \%
62.476 \% compatibility with hebfonts:
62.477 \DeclareTextFontCommand\{\textjm\}\{\rmfamily \(\backslash\) selectfont\}
62.478 \DeclareTextFontCommand\{\textds\}\{\bfseries \(\backslash\) selectfont\}
62.479 \DeclareTextFontCommand\{\textoj\}\{\itshape\selectfont\}
62.480 \DeclareTextFontCommand\{\textta\}\{\sffamily \(\backslash\) selectfont\}
62.481
\(62.482 \%\) an attempt to give some replacements to the original hebfonts:
62.483 \%
62.484 \DeclareTextFontCommand\{\textcrml\}\{\fontfamily\{david\}\selectfont\}
62.485 \DeclareTextFontCommand\{\textfr\}\{\fontfamily\{frank\}\selectfont\}
62.486 \DeclareTextFontCommand\{\textredis\}\{\fontfamily\{aharoni\}\selectfont\}
```

62.487 \DeclareTextFontCommand{\textclas}{\fontfamily{drugulin}\selectfont}
62.488 \DeclareTextFontCommand{\textshold}{\fontfamily{frank}\selectfont}
62.489 \DeclareTextFontCommand{\textshscr}{\fontfamily{yad}\selectfont}
62.490 \DeclareTextFontCommand{\textshstk}{\fontfamily{aharoni}\selectfont}
62.491 % note that redis is larger than shstk
6 2 . 4 9 2
6 2 . 4 9 3
62.494 \DeclareTextFontCommand{\textaha}{\fontfamily{aharoni}\selectfont}
62.495 \DeclareTextFontCommand{\textdav}{\fontfamily{david}\selectfont}
62.496 \DeclareTextFontCommand{\textdru}{\fontfamily{drugulin}\selectfont}
62.497 \DeclareTextFontCommand{\textel} {\fontfamily{ellinia}\selectfont}
6 2 . 4 9 8 ~ \% ~ \ t e x t f r ~ i s ~ a l r e a d y ~ d e c l a r e d ~ a b o v e
62.499 \DeclareTextFontCommand{\textmir}{<br>fontfamily{miriam}\selectfont}
62.500 \DeclareTextFontCommand{\textna} {\fontfamily{nachlieli}\selectfont}
62.501 % is this necessary:
62.502 \DeclareTextFontCommand{\textyad} {\fontfamily{yad}\selectfont}
6 2 . 5 0 3
62.504 \else%
62.505\DeclareTextFontCommand{\textjm}{\rmfamily\selectfont}

```
\textds Switches to Dead Sea font which is default bold font in Hebrew. Commands \textbf\{...\} and old-style \(\{\backslash \mathrm{bf} \ldots\}\) will produce the same result. \(62.506 \backslash\) DeclareTextFontCommand\{\textds\}\{\bfseries \(\backslash\) selectfont \(\}\)
\textoj Switches to Old Jaffa font which is default italic font in Hebrew. Commands \textit\{...\}, \textsl\{...\}, \emph\{...\} and old-style \(\{\backslash i t \ldots\}\) or \(\{\backslash e m \ldots\}\) will produce the same result.
62.507 \DeclareTextFontCommand\{\textoj\}\{\itshape \(\backslash\) selectfont \(\}\)
\textta Switches to Tel-Aviv font which is default sans-serif font in Hebrew. Commands \(\backslash\) textsf\{...\}, \texttt\{...\} and old-style \(\{\backslash\) sf \(\ldots\}\) or \(\{\backslash t t \ldots\}\) will produce the same result (because sans-serif is used as typewriter font when in Hebrew mode).
62.508 \DeclareTextFontCommand\{\textta\}\{\sffamily\selectfont\}
\textcrml Switches to Carmel font. Regular and slanted variants of carmel font will be used.. 62.509 \DeclareTextFontCommand\{\textcrml\}\{\fontfamily\{crml\}\selectfont\}
\textfr Switches to Frank-Ruehl font family. Regular, bold and slanted frank ruehl fonts will be used.
\(62.510 \backslash\) DeclareTextFontCommand\{\textfr\}\{\fontfamily\{fr\}\selectfont\}
\textredis Switches to Redis font family. Regular, bold and slanted redis fonts of various sizes will be used.
62.511 \DeclareTextFontCommand\{\textredis\}\{\fontfamily\{redis\}\selectfont\}
\textclas Switches to Classic font family. The normal font will be hclassic and slanted hcaption.
\(62.512 \backslash\) DeclareTextFontCommand\{\textclas\}\{\fontfamily\{clas\}\selectfont\}
\begin{tabular}{lll} 
Old font command & Font name & Comment \\
\(\{\backslash j \mathrm{~m} \ldots\}\) & Jerusalem & default regular (roman) font \\
\(\{\backslash\) ds \(\ldots\}\) & Dead Sea & default bold font \\
\(\{\backslash \mathrm{oj} \ldots\}\) & Old Jaffa & \begin{tabular}{l} 
default italic and slanted font \\
used also to emphasize text
\end{tabular} \\
\(\{\backslash\) ta \(\ldots\}\) & Tel-Aviv & \begin{tabular}{l} 
default sans-serif and typewriter font
\end{tabular}
\end{tabular}

Table 34: Hebrew old font-change commands for compatibility mode
\textshold Switches to Shalom Old Style font.
\(62.513 \backslash\) DeclareTextFontCommand\{\textshold\}\{\fontfamily\{shold\}\selectfont\}
\textshscr Switches to Shalom Script font.
62.514 \DeclareTextFontCommand\{\textshscr\}\{\fontfamily\{shscr\}\selectfont\}
\textshstk Switches to Shalom Stick font.
```

62.515\DeclareTextFontCommand{\textshstk}{\fontfamily{shstk}\selectfont}

```
62.516 \fi

Finally, for backward compatibility with \(\mathrm{IAT}_{\mathrm{E}} 2.09\). four old font commands, e.g. \(\{\backslash j m \ldots\}\) are defined too (see Table 34).
```

62.517 \if@compatibility

```
62.518 \DeclareOldFontCommand\{\jm\}\{\normalfont \(\backslash\) rmfamily \(\backslash\) selectfont \(\} \%\)
\(\{\backslash @\) nomath \(\backslash j m\}\)
62.520 \DeclareOldFontCommand \(\{\backslash\) ds \(\}\{\backslash\) normalfont \(\backslash\) bfseries \(\backslash\) selectfont \(\} \%\)
\(62.521 \quad\{\backslash @\) nomath \(\backslash\) ds \(\}\)
62.522 \DeclareOldFontCommand\{\oj\}\{\normalfont \(\backslash\) itshape \(\backslash\) selectfont \(\} \%\)
\(62.523 \quad\{\backslash @ n o m a t h \backslash o j\}\)
62.524 \DeclareOldFontCommand\{\ta\}\{\normalfont \(\backslash\) sffamily \(\backslash\) selectfont \(\} \%\)
\(62.525 \quad\{\backslash @\) nomath \(\backslash\) ta\}
62.526 \fi
62.527 〈/hebfont〉

\section*{63 Hebrew in \(\mathrm{E}_{\mathrm{E}} \mathrm{XX} 2.09\) compatibility mode}
\documentstyle command in the preamble of \(\mathrm{IAT}_{\mathrm{E}} \mathrm{X}\) document indicates that it is a \(\mathrm{EAT}_{\mathrm{E}} \mathrm{X} 2.09\) 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 hebcal can be included after the abovenamed Hebrew style option，for example：
\documentstyle［12pt，hebrew＿p，hebcal］\｛report\}.
Any Hebrew document which compiled under \(\mathrm{IAT}_{\mathrm{E}} 2.09\) should compile under compatibility mode，unless it uses low－level commands such as \tenrm．

\section*{63．1 The DOcSTRIP modules}

The following modules are used in the implementation to direct DOcstrip in generating the external files：
\begin{tabular}{ll} 
newcode & produce hebrew＿newcode．sty \\
pccode & produce hebrew＿p．sty \\
oldcode & produce hebrew＿oldcode．sty
\end{tabular}

\section*{63．2 Obsolete style files}

For each of the Hebrew \(\mathrm{EAT}_{\mathrm{E}} \mathrm{X}\) 2．09 Hebrew styles，we produce a file which uses correct input encoding and calls babel with Hebrew and English language op－ tions．This means that any styles which say \input hebrew＿newcode．sty or \documentstyle［．．．hebrew＿newcode．．．］\｛．．．\} should still work.
```

63.1 〈*newcode | pccode | oldcode\rangle
63.2 \NeedsTeXFormat{LaTeX2e}
63.3 </newcode | pccode | oldcode\rangle
63.4〈*newcode\rangle
63.5\@obsoletefile{hebrew.sty}{hebrew_newcode.sty}
63.6 \RequirePackage [8859-8] {inputenc}
63.7 〈/newcode\
63.8〈*pccode\rangle
63.9\@obsoletefile{hebrew.sty}{hebrew_p.sty}
63.10\RequirePackage[cp862] {inputenc}
63.11 \/pccode>
63.12 <*oldcode〉
63.13 \@obsoletefile{hebrew.sty}{hebrew_oldcode.sty}
63.14\RequirePackage[si960] {inputenc}
63.15 </oldcode\
63.16 <*newcode | pccode | oldcode\rangle
63.17\RequirePackage[english,hebrew] {babel}
63.18 </newcode | pccode | oldcode\

```

\section*{64 The Bahasa language}

The file bahasa.dtx \({ }^{74}\) 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 pronounciation, this file can be used for both languages.

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.
```

64.1 <*code>
64.2\LdfInit{bahasa}\captionsbahasa

```

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 \1@bahasa to see whether we have to do something here.
```

64.3\ifx\l@bahasa\@undefined
64.4 \@nopatterns{Bahasa}
64.5 \adddialect\l@bahasa0\fi

```

The next step consists of defining commands to switch to (and from) the Bahasa language.
\captionsbahasa The macro \captionsbahasa defines all strings used in the four standard documentclasses provided with \(\mathrm{IAT}_{\mathrm{E}} \mathrm{X}\).
```

64.6 \addto\captionsbahasa{%
64.7 \def\prefacename{Pendahuluan}%
64.8 \def\refname{Pustaka}%
64.9 \def\abstractname{Ringkasan}% (sometime it's called 'intisari'
64.10 % or 'ikhtisar')
64.11 \def\bibname{Bibliografi}%
64.12\def\chaptername{Bab}%
64.13\def\appendixname{Lampiran}%
64.14\def\contentsname{Daftar Isi}%
64.15\def\listfigurename{Daftar Gambar}%
64.16 \def\listtablename{Daftar Tabel}%
64.17 \def\indexname{Indeks}%
64.18<br>def\figurename{Gambar}%
64.19 \def\tablename{Tabel}%
64.20 \def\partname{Bagian}%
64.21% Subject: Subyek
64.22 % From: Dari
64.23\def\enclname{Lampiran}%
64.24 \def\ccname{cc}%
64.25\def\headtoname{Kepada}%
64.26 \def\pagename{Halaman}%
64.27% Notes (Endnotes): Catatan

```

\footnotetext{
\({ }^{74}\) The file described in this section has version number v1.0i and was last revised on 2005/03/29
}
```

64.28\def\seename{lihat}%
64.29 \def\alsoname{lihat juga}%
64.30 \def\proofname{Bukti}%
64.31 \def\glossaryname{Daftar Istilah}%
64.32 }

```
\datebahasa The macro \datebahasa redefines the command \today to produce Bahasa dates.
```

64.33\def\datebahasa{%
64.34\def\today{\number\day~\ifcase\month\or
64.35 Januari\or Pebruari\or Maret\or April\or Mei\or Juni\or
64.36 Juli\or Agustus\or September\or Oktober\or Nopember\or Desember\fi
64.37 \space \number\year}}

```
\extrasbahasa The macro \extrasbahasa will perform all the extra definitions needed for the \noextrasbahasa 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.
```

64.38 \addto\extrasbahasa{}
64.39 \addto\noextrasbahasa{}

```
\bahasahyphenmins The bahasa hyphenation patterns should be used with \lefthyphenmin set to 2 and \righthyphenmin set to 2.
64.40 \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 \(\backslash\) begin\{document\} and resetting the category code of @ to its original value.
\(64.41 \backslash l d f @ f i n i s h\{b a h a s a\}\)
64.42 〈/code〉

\section*{65 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 \(T_{E} X\)-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 localhyphen.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 acheive 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 ini \(\mathrm{T}_{\mathrm{E}} \mathrm{X}\) sees, we need to set some category codes just to be able to change the definition of \input
```

65.1 〈*bplain | blplain\rangle
65.2 \catcode'\{=1 % left brace is begin-group character
65.3\catcode'\}=2 % right brace is end-group character
65.4 \catcode'\#=6 % hash mark is macro parameter character

```

Now let's see if a file called hyphen.cfg can be found somewhere on \(\mathrm{T}_{\mathrm{E}}\) 's input path by trying to open it for reading...
65.5 \openin 0 hyphen.cfg

If the file wasn't found the following test turns out true.
65.6 \ifeof0
65.7 \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).
65.8 \let \(\backslash a \backslash i n p u t\)

Then \input is defined to forget about its argument and load hyphen. cfg instead.
\begin{tabular}{cc}
65.9 & \def \input \#1 \(\{\%\) \\
65.10 & \(\backslash l e t \backslash i n p u t \backslash a\) \\
65.11 & \(\backslash a\) hyphen.cfg
\end{tabular}

Once that's done the original meaning of \input can be restored and the definition of \(\backslash \mathrm{a}\) can be forgotten.
```

65.12 \let\a\undefined
65.13 }

```
```

65.14 \fi
65.15 </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．
65.16 〈bplain〉\a plain．tex
65.17 〈blplain〉 \a lplain．tex

Finally we change the contents of \(\backslash\) fmtname to indicate that this is not the plain format，but a format based on plain with the babel package preloaded．
65.18 〈bplain〉\def \fmtname\｛babel－plain\}
65.19 〈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．

\section*{66 Support for formats based on PLAINTEX}

The following code duplicates or emulates parts of \(\mathrm{IATEX}_{\mathrm{E}} 2_{\varepsilon}\) that are needed for babel．
```

66.1 〈*code>
66.2\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．
66．3 \def \＠empty\｛\}
We need to define \loadlocalcfg for plain users as the \(\mathrm{LAT}_{\mathrm{E}} \mathrm{X}\) definition uses \InputIfFileExists．
66.4 \def \(\backslash\) loadlocalcfg\＃1\｛\％
66.5 \openin0\＃1．cfg
66.6 \ifeof0
66.7 \closein0
66.8 \else
66.9 \closein0
\(66.10 \quad\{\backslash i m m e d i a t e \backslash w r i t e 16\{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\} \%\)
66.11 \immediate\write16\｛＊Local config file \＃1．cfg used\}\%
66.12 \immediate\write16\｛＊\}\%
66.13 \}
66.14 \input \＃1．cfg\relax
66.15 \fi

We have to execute \＠endofldf in this case
66.16 \＠endofldf
66.17 \}

We want to add a message to the message \(\mathrm{IAT}_{\mathrm{E}} 2.09\) 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 \(\mathrm{IA}_{\mathrm{E}} \mathrm{X}\) fills the register.

There are some problems with this approach though.
- When someone wants to use several hyphenation patterns with SLIEX the above scheme won't work. The reason is that \(\mathrm{SUT}_{\mathrm{E}} \mathrm{X}\) overwrites the contents of the \everyjob register with its own message.
- Plain \(\mathrm{T}_{\mathrm{E}} \mathrm{X}\) 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.
```

66.18 \let\orig@dump=\dump
66.19 \def\dump{%

```

To make sure that \(\mathrm{IAT}_{\mathrm{E}} 2.09\) 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.
```

66.20 \ifx\@ztryfc\@undefined
66.21 \else
66.22 \toks0=\expandafter{\@preamblecmds}
66.23<br>edef\@preamblecmds{\noexpand\@begindocumenthook\the\toks0}
66.24 \def\@begindocumenthook{}
66.25 \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.
```

66.26 \everyjob\expandafter{\the\everyjob%
66.27 \immediate\write16{\the\toks8 loaded.}}%

```

Then everything is restored to the old situation and the format is dumped.
```

66.28 \let\dump\orig@dump\let\orig@dump\@undefined\dump\}
66.29 \expandafter $\backslash e n d i n p u t$
66.30 \fi

```

The rest of this file is not processed by iniTEX but during the normal document run. A number of \(\mathrm{IAT}_{\mathrm{E}} \mathrm{X}\) macro's that are needed later on.
```

66.31 \long\def\@firstofone\#1{\#1}
66.32\long\def\@firstoftwo\#1\#2{\#1}

```
```

66.33\long\def\@secondoftwo\#1\#2{\#2}
66.34 \def\@ifstar\#1{\@ifnextchar *{\@firstoftwo{\#1}}}
66.35 \def\@star@or@long\#1{%
66.36 \@ifstar
66.37 {\let\l@ngrel@x\relax\#1}%
66.38 {\let\l@ngrel@x\long\#1}}
66.39\let\l@ngrel@x\relax
66.40 \def\@car\#1\#2\@nil{\#1}
66.41 \def\@cdr\#1\#2\@nil{\#2}
66.42\let\@typeset@protect\relax
66.43\long\def\@gobble\#1{}
66.44 \edef\@backslashchar{\expandafter\@gobble\string<br>}
66.45 \def\strip@prefix\#1>{}
66.46\def\g@addto@macro\#1\#2{{%
66.47 \toks@\expandafter{\#1\#2}%
66.48 \xdef\#1{\the\toks@}}}
66.49 \def\@namedef\#1{\expandafter\def\csname \#1\endcsname}
66.50 \def\@ifundefined\#1{%
66.51 \expandafter\ifx\csname\#1\endcsname\relax
66.52 \expandafter\@firstoftwo
66.53 \else
66.54 \expandafter\@secondoftwo
66.55 \fi}

```

LATEX \(2 \varepsilon\) has the command \@onlypreamble which adds commands to a list of commands that are no longer needed after \begin\{document\}. }
```

66.56\ifx\@preamblecmds\@undefined
66.57 \def\@preamblecmds{}
66.58 \fi
66.59 \def\@onlypreamble\#1{%
66.60 \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
66.61 \@preamblecmds\do\#1}}
66.62 \@onlypreamble\@onlypreamble

```

Mimick \(L^{A} T_{E X}\) 's \(\backslash\) AtBeginDocument; for this to work the user needs to add \(\backslash\) begindocument to his file.
\(66.63 \backslash\) def \(\backslash\) begindocument \(\{\%\)
66.64 \@begindocumenthook
66.65 \global\let\@begindocumenthook\@undefined
66.66 \def \do\#\#1\{\global\let \#\#1\@undefined\}\%
66.67 \@preamblecmds
66.68 \global\let\do\noexpand
66.69 \}
66.70 \ifx\@begindocumenthook\@undefined
66.71 \def \@begindocumenthook\{\}
66.72 \fi
66.73 \@onlypreamble\@begindocumenthook
\(66.74 \backslash\) def \(\backslash\) AtBeginDocument \(\{\backslash\) g@addto@macro\@begindocumenthook\}

We also have to mimick LATEX's \(\backslash\) AtEndOfPackage. Our replacement macro is much simpler; it stores its argument in \@endofldf.
```

66.75\def\AtEndOfPackage\#1{\g@addto@macro\@endofldf{\#1}}
66.76 \@onlypreamble\AtEndOfPackage
66.77 \def\@endofldf{}
66.78 \@onlypreamble\@endofldf

```
\({ }^{\mathrm{LA}} \mathrm{T}_{\mathrm{E}} \mathrm{X}\) needs to be able to switch off writing to its auxiliary files; plain doesn't have them by default.
```

66.79\ifx\if@filesw\@undefined
66.80 \expandafter\let\csname if@filesw\expandafter\endcsname
66.81 \csname iffalse\endcsname
66.82\fi

```

Mimick \(\mathrm{LA}_{\mathrm{E}} \mathrm{X}\) 's commands to define control sequences.
```

66.83\def\newcommand{\@star@or@long\new@command}
66.84 \def\new@command\#1{%
66.85 \@testopt{\@newcommand\#1}0}
66.86 \def\@newcommand\#1[\#2] {%
66.87 \@ifnextchar [{\@xargdef\#1[\#2]}%
66.88 {\@argdef\#1[\#2]}}
66.89 \long\def\@argdef\#1[\#2]\#3{%
66.90 \@yargdef\#1\@ne{\#2}{\#3}}
66.91 \long\def\@xargdef\#1[\#2] [\#3] \#4{%
66.92 \expandafter\def\expandafter\#1\expandafter{%
66.93 \expandafter\@protected@testopt\expandafter \#1%
66.94 \csname\string\#1\expandafter\endcsname{\#3}}%
66.95 \expandafter\@yargdef \csname\string\#1\endcsname
66.96 \tw@{\#2}{\#4}}
66.97\long\def\@yargdef\#1\#2\#3{%
66.98 \@tempcnta\#3\relax
66.99 \advance \@tempcnta \@ne
66.100 \let\@hash@\relax
66.101 \edef\reserved@a{\ifx\#2\tw@ [\@hash@1]\fi}%
66.102 \@tempcntb \#2%
66.103<br>@whilenum\@tempentb <\@tempenta
66.104 \do{%
66.105 \edef\reserved@a{\reserved@a\@hash@\the\@tempentb}%
66.106 \advance\@tempcntb \@ne}%
66.107 \let\@hash@\#\#%
66.108 \l@ngrel@x\expandafter\def\expandafter\#1\reserved@a}
66.109 \let\providecommand\newcommand
66.110\def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
66.111 \def\declare@robustcommand\#1{%
66.112 \edef\reserved@a{\string\#1}%
66.113 \def\reserved@b{\#1}%
66.114<br>edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
66.115 \edef\#1{%
66.116 \ifx\reserved@a\reserved@b
66.117 \noexpand\x@protect

```
```

66.118 \noexpand\#1%
66.119 \fi
66.120 \noexpand\protect
66.121 \expandafter\noexpand\csname
66.122 \expandafter\@gobble\string\#1 \endcsname
66.123 }%
66.124 \expandafter\new@command\csname
6 6 . 1 2 5
66.126 }
66.127 \def\x@protect\#1{%
66.128 \ifx\protect\@typeset@protect\else
66.129 \@x@protect\#1%
66.130 \fi
6 6 . 1 3 1 \}
66.132\def\@x@protect\#1\fi\#2\#3{%
66.133 \fi\protect\#1%
66.134 }

```

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 \(\backslash \mathrm{bbl@t}\) empa.
```

66.135\def\bbl@tmpa{\csname newif\endcsname\ifin@}
66.136 \ifx\in@\@undefined
66.137 \def\in@\#1\#2{%
66.138 \def\in@@\#\#1\#1\#\#2\#\#3\in@@{%
66.139 \ifx\in@\#\#2\in@false\else\in@true\fi}%
66.140 \in@@\#2\#1\in@\in@@}
66.141 \else
66.142 \let\bbl@tmpa\@empty
66.143 \fi
66.144 \bbl@tmpa

```
\(\mathrm{LAT}_{\mathrm{E}} \mathrm{X}\) 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 \(\mathrm{T}_{\mathrm{E}} \mathrm{X}\) 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).
```

66.145\def\@ifpackagewith\#1\#2\#3\#4{%
66.146 \#3}

```

The \({ }^{A} T_{E} \mathrm{X}\) macro \@if1@aded checks whether a file was loaded. This functionality is not needed for plain \(\mathrm{T}_{\mathrm{E}} \mathrm{X}\) but we need the macro to be defined as a no-op.
66.147 \def \@ifl@aded\#1\#2\#3\#4\{\}

For the following code we need to make sure that the commands \(\backslash\) newcommand and \providecommand exist with some sensible definition. They are not fully equivalent to their \(\operatorname{LAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}\) versions; just enough to make things work in plain \(\mathrm{T}_{\mathrm{E}} \mathrm{Xenvironments}\).
```

66.148\ifx\@tempcnta\@undefined
66.149 \csname newcount\endcsname\@tempcnta\relax
66.150 \fi
66.151 \ifx\@tempcntb\@undefined
66.152 \csname newcount\endcsname\@tempcntb\relax
66.153\fi

```

To prevent wasting two counters in \(\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2.09\) (because counters with the same name are allocated later by it) we reset the counter that holds the next free counter ( \(\backslash\) count10).
66.154 \ifx \(\backslash\) bye \@undefined
66.155 \advance\count10 by \(-2 \backslash r e l a x\)
66.156 \fi
66.157 \ifx\@ifnextchar\@undefined
66.158 \def \@ifnextchar\#1\#2\#3\{\%
66.159 \let \(\backslash\) reserved@d=\#1\%
66.160 \def \(\backslash\) reserved@a\{\#2\}\def \(\backslash\) reserved@b\{\#3\}\%
66.161 \futurelet \@let@token\@ifnch\}
66.162 \def \@ifnch\{\%
66.163 \ifx\@let@token\@sptoken
\(66.164 \backslash\) let \(\backslash r e s e r v e d @ c \backslash @ x i f n c h\)
66.165 \else
66.166 \ifx\@let@token\reserved@d
66.167 \let\reserved@c\reserved@a
66.168 \else
66.169 \let \(\backslash r e s e r v e d @ c \backslash r e s e r v e d @ b ~\)
66.170 \fi
66.171 \fi
66.172 \reserved@c\}
66.173 \def \:\{\let\@sptoken= \} \: \% this makes \@sptoken a space token
\(66.174 \backslash \operatorname{def} \backslash:\{\backslash @ x i f n c h\}\) \expandafter\def \(\backslash:\) \{\futurelet \(\backslash\) @let@token \@ifnch\}
66.175 \fi
66.176 \def \@testopt\#1\#2\{\%
66.177 \@ifnextchar[\{\#1\}\{\#1[\#2]\}\}
66.178 \def \@protected@testopt\#1\{\%\%
66.179 \ifx\protect\@typeset@protect
66.180 \expandafter \@testopt
66.181 \else
66.182 \@x@protect\#1\%
66.183 \fi\}
66.184 \long \(\backslash\) def \@whilenum\#1\do \#2\{\ifnum \#1\relax \#2\relax\@iwhilenum\{\#1\relax
66.185 \#2\relax\}\fi\}
\(66.186 \backslash\) long \def \@iwhilenum\#1\{\ifnum \#1\expandafter \(\backslash\) @iwhilenum
\(66.187 \backslash e l s e \backslash e x p a n d a f t e r \backslash @ g o b b l e \backslash f i\{\# 1\}\}\)

Code from ltoutenc.dtx, adapted for use in the plain \(\mathrm{T}_{\mathrm{E}} \mathrm{X}\) environment.
66.188 \def \(\backslash\) DeclareTextCommand\{\%
66.189 \@dec@text@cmd \(\backslash\) providecommand
66.190 \}
66.191 \def \(\backslash\) ProvideTextCommand\{\%
```

66.192 \@dec@text@cmd\providecommand
66.193 }
66.194 \def \DeclareTextSymbol\#1\#2\#3{%
66.195 \@dec@text@cmd\chardef\#1{\#2}\#3\relax
66.196 }
66.197 \def\@dec@text@cmd\#1\#2\#3{%
66.198 \expandafter\def\expandafter\#2%
66.199 \expandafter{%
66.200 \csname\#3-cmd\expandafter\endcsname
66.201 \expandafter\#2%
66.202 \csname\#3\string\#2\endcsname
66.203 }%
66.204% \let\@ifdefinable\@rc@ifdefinable
66.205 \expandafter\#1\csname\#3\string\#2\endcsname
66.206 }
66.207\def\@current@cmd\#1{%
66.208 \ifx\protect\@typeset@protect\else
66.209 \noexpand\#1\expandafter\@gobble
66.210 \fi
66.211}
66.212 \def\@changed@cmd\#1\#2{%
66.213 \ifx\protect\@typeset@protect
66.214 \expandafter\ifx\csname\cf@encoding\string\#1\endcsname\relax
66.215 \expandafter\ifx\csname ?\string\#1\endcsname\relax
66.216 \expandafter\def\csname ?\string\#1\endcsname{%
\@changed@x@err{\#1}%
}%
66.218 }%
66.219 \fi
66.220 \global\expandafter\let
66.221 \csname\cf@encoding \string\#1\expandafter\endcsname
66.222 \csname ?\string\#1\endcsname
66.223 \fi
66.224 \csname\cf@encoding\string\#1%
66.225 \expandafter\endcsname
66.226 \else
66.227 \noexpand\#1%
66.228 \fi
6 6 . 2 2 9 \}
66.230 \def\@changed@x@err\#1{%
66.231 \errhelp{Your command will be ignored, type <return> to proceed}%
66.232 \errmessage{Command \protect\#1 undefined in encoding \cf@encoding}}
66.233 \def\DeclareTextCommandDefault\#1{%
66.234 \DeclareTextCommand\#1?%
66.235 }
66.236 \def\ProvideTextCommandDefault\#1{%
66.237 \ProvideTextCommand\#1?%
66.238 }
66.239 \expandafter\let\csname OT1-cmd\endcsname\@current@cmd
66.240 \expandafter\let\csname?-cmd\endcsname\@changed@cmd
66.241 \def\DeclareTextAccent\#1\#2\#3{%

```
```

66.242 \DeclareTextCommand\#1{\#2}[1]{\accent\#3 \#\#1}
66.243 }
66.244 \def\DeclareTextCompositeCommand\#1\#2\#3\#4{%
66.245 \expandafter\let\expandafter\reserved@a\csname\#2\string\#1\endcsname
66.246 \edef \reserved@b{\string\#\#1}%
66.247 \edef\reserved@c{%
66.248 \expandafter\@strip@args\meaning\reserved@a:-\@strip@args}%
66.249 \ifx\reserved@b\reserved@c
66.250 \expandafter\expandafter\expandafter\ifx
66.251 \expandafter\@car\reserved@a\relax\relax\@nil
66.252 \@text@composite
66.253 \else
66.254 \edef\reserved@b\#\#1{%
66.255 \def\expandafter\noexpand
66.256 \csname\#2\string\#1\endcsname\#\#\#\#1{%
66.257 \noexpand\@text@composite
66.258 \expandafter\noexpand\csname\#2\string\#1\endcsname
66.259 \#\#\#\#1\noexpand\@empty\noexpand\@text@composite
66.260 {\#\#1}%
66.261 }%
66.262 }%
66.263 \expandafter\reserved@b\expandafter{\reserved@a{\#\#1}}%
66.264 \fi
66.265 \expandafter\def\csname\expandafter\string\csname
66.266 \#2\endcsname\string\#1-\string\#3\endcsname{\#4}
66.267 \else
66.268 \errhelp{Your command will be ignored, type <return> to proceed}%
66.269 \errmessage{\string\DeclareTextCompositeCommand\space used on
inappropriate command \protect\#1}
\i
66.272 }
66.273\def\@text@composite\#1\#2\#3\@text@composite{%
66.274 \expandafter\@text@composite@x
66.275 \csname\string\#1-\string\#2\endcsname
66.276 }
66.277 \def\@text@composite@x\#1\#2{%
66.278 \ifx\#1\relax
66.279 \#2%
66.280 \else
66.281 \#1%
66.282 \fi
66.283 }
66.284 %
66.285\def\@strip@args\#1:\#2-\#3\@strip@args{\#2}
66.286 \def\DeclareTextComposite\#1\#2\#3\#4{%
66.287<br>def\reserved@a{\DeclareTextCompositeCommand\#1{\#2}{\#3}}%
66.288 \bgroup
66.289 \lccode`\@=\#4%
66.290 \lowercase{%
66.291 \egroup

```
```

66.292 \reserved@a @%
6 6 . 2 9 3
}%
66.294 }
66.295 %
66.296 \def\UseTextSymbol\#1\#2{%
66.297% \let\@curr@enc\cf@encoding
66.298% \@use@text@encoding{\#1}%
66.299 \#2%
66.300 % \@use@text@encoding\@curr@enc
6 6 . 3 0 1 \}
66.302 \def\UseTextAccent\#1\#2\#3{%
66.303 % \let\@curr@enc\cf@encoding
66.304% \@use@text@encoding{\#1}%
66.305% \#2{\@use@text@encoding\@curr@enc\selectfont\#3}%
66.306 % \@use@text@encoding\@curr@enc
66.307 }
66.308 \def\@use@text@encoding\#1{%
66.309 % \edef\f@encoding{\#1}%
66.310% \xdef\font@name{%
66.311%<br>csname\curr@fontshape/\f@size\endcsname
66.312 % }%
66.313% \pickup@font
66.314% \font@name
66.315 % \@@enc@update
66.316}
66.317\def\DeclareTextSymbolDefault\#1\#2{%
66.318 \DeclareTextCommandDefault\#1{\UseTextSymbol{\#2}\#1}%
6 6 . 3 1 9 \}
66.320 \def\DeclareTextAccentDefault\#1\#2{%
66.321 \DeclareTextCommandDefault\#1{\UseTextAccent{\#2}\#1}%
6 6 . 3 2 2 ~ \}
66.323\def\cf@encoding{OT1}

```

Currently we only use the \(\mathrm{IAT}_{\mathrm{E}} \mathrm{X} 2_{\varepsilon}\) method for accents for those that are known to be made active in some language definition file.
```

66.324 \DeclareTextAccent{\"}{0T1}{127}
66.325 \DeclareTextAccent{\'}{0T1}{19}
66.326 \DeclareTextAccent{\^}{0T1}{94}
66.327 \DeclareTextAccent{\'}{0T1}{18}
66.328 \DeclareTextAccent{\~}{0T1}{126}

```

The following control sequences are used in babel.def but are not defined for plain \(\mathrm{T}_{\mathrm{E}} \mathrm{X}\).
66.329 \DeclareTextSymbol\{\textquotedblleft\}\{0T1\}\{92\}
66.330 \DeclareTextSymbol\{\textquotedblright\}\{0T1\}\{'\"\}
66.331 \DeclareTextSymbol\{\textquoteleft\}\{OT1\}\{‘\‘\}
66.332 \DeclareTextSymbol\{\textquoteright\}\{0T1\}\{'\'\}
66.333 \DeclareTextSymbol\{\i\}\{0T1\}\{16\}
66.334 \DeclareTextSymbol\{\ss\}\{0T1\}\{25\}

For a couple of languages we need the \(\mathrm{IAT}_{\mathrm{E}} \mathrm{X}\)-control sequence \(\backslash\) scriptsize to be available. Because plain \(T_{E} X\) doesn't have such a sofisticated font mechanism as LeTEX has, we just \let it to \sevenrm.
66.335 \ifx\scriptsize\@undefined
66.336 \let\scriptsize\sevenrm
66.337 \fi
66.338 (/code〉

\section*{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.
 adddialect \(=\) \subitem \(*+\backslash\) adddiađect+, \usage\{11\} loadlocalcfg= \(\backslash\) subitem \(*+\backslash\) loadlocalcfg+, addlanguage \(=\) \subitem *enlvaddinanguggetherkargagegel 1\(\}\) otherlanguage, addto \(=\) \subitem \(*+\backslash\) addto + , \usage\{14\} language \(=\) \subitem language + , \usage\{12

allowhyphens \(=\) subitem \(*+\backslash\) allowhyphens + , \usage\{14\}extraslang \(=\) subitem \(*+\backslash\) noextraslang + , Apageref \(=\quad\) subitem \(*+\backslash\) Afpagiemefingułgeage\{2563\}ubitem \(*+\backslash\) foreignlanguage + , \usage\{5\}
Aref \(=\quad\) subitem \(*+\backslash\) Aref fulluasage \(\{2566\) tem \(*+\backslash\) full + onturage \(\{390\}\) subitem \(*+\backslash\) ontoday + , \usage \(\{2\)
\(\mathrm{Az}=\) \subitem \(*+\backslash \mathrm{Az}^{+}\), \usage\{255\} otherlanguage \({ }^{*}=\) otherlanguage \(*\)


\(\mathrm{Azr}=\) \subitem \(*+\backslash\) Azr + , gxoded 255\(\}\) \subitem \(*+\backslash\) greektext+ (envirousagent \(\{15\}_{5}\)




frenchspacing \(=\) subitem frenchspacing + , \usage\{14 \({ }^{\text {selectlanguage }}=\) \subitem \(*+\backslash\) selectlanguag
 shorthandon \(=\quad \backslash\) subitem \(*+\backslash\) shorthandon + ,




 environments:hyphenrules=>hyppisknnules\\,subitem finish+, \usage\{12\}

5 quit= \subitem quit+, \usadertałnds=
\subitem *+\useshorthands```


[^0]:    During the development ideas from Nico Poppelier, Piet van Oostrum and many others have been used. Bernd Raichle has provided many helpful suggestions.

[^1]:    ${ }^{1}$ This mechanism was introduced by Bernd Raichle.

[^2]:    ${ }^{2}$ The ' $b$ ' is added to the name to distinguish the file from Partls' file.

[^3]:    ${ }^{3}$ The file described in this section is called babel.dtx, has version number v3.8g and was last revised on $2005 / 05 / 21$.

[^4]:    ${ }^{4}$ This is because different operating systems sometimes use very different file-naming conventions.

[^5]:    ${ }^{5}$ This code is based on code presented in TUGboat vol. 12, no2, June 1991 in "An expansion Power Lemma" by Sonja Maus.

[^6]:    ${ }^{6}$ \originalTeX has to be expandable，i．e．you shouldn＇t let it to \relax．

[^7]:    ${ }^{7} \mathrm{~T}_{\mathrm{E}} \mathrm{X}$ begins and ends a word for hyphenation at a glue node．The penalty prevents a linebreak at this glue node．

[^8]:    ${ }^{8}$ The file described in this section has version number v 3.8 g , and was last revised on 2005/05/21.

[^9]:    ${ }^{9} \mathrm{~A}$ fact that was pointed out to me by Nico Poppelier and was already used in Piet van Oostrum's document style option nl .

[^10]:    ${ }^{10}$ The file described in this section has version number ? and was last revised on ?. A contribution was made by Ruiz-Altaba Marti (ruizaltb@cernvm.cern.ch). Code from the file esperant.sty by Jörg Knappen (knappen@vkpmzd.kph.uni-mainz.de) was included.

[^11]:    ${ }^{11}$ The file described in this section has version number v1.6 and was last revised on 2005/03/30

[^12]:    ${ }^{12}$ The file described in this section has version number v3.8i, and was last revised on 2005/03/30.
    ${ }^{13}$ contributed by Stoffel Lombard (lombc@b31pc87.up.ac.za)

[^13]:    ${ }^{14}$ The file described in this section has version number v 3.3 o and was last revised on 2005/03/30.

[^14]:    ${ }^{15}$ The file described in this section has version number v 2.6 k and was last revised on 2004/02/19.
    ${ }^{16}$ This file is a re-implementation of Hubert Partl's german.sty version 2.5b, see [4].

[^15]:    ${ }^{19}$ The file described in this section has version number v1.0d and was last revised on 2005/03/31.

[^16]:    ${ }^{20}$ The file described in this section has version number v 1.0 h and was last revised on 2005/03/30. A contribution was made by Marion Gunn.

[^17]:    ${ }^{21}$ The file described in this section has version number v 1.0 g and was last revised on 2005/03/31. A contribution was made by Fraser Grant (FRASER@CERNVM).

[^18]:    ${ }^{22}$ The file described in this section has version number v1.31 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.

[^19]:    ${ }^{23}$ Apostolos Syropoulos wishes to thank him for his patience, collaboration, cooments and suggestions.

[^20]:    ${ }^{24}$ The file described in this section has version number ? and was last revised on ?.
    ${ }^{25}$ \selectlanguage\{francais\} and \selectlanguage\{frenchb\} are kept for backward compatibility but should no longer be used.

[^21]:    ${ }^{26}$ The file described in this section has version number v 1.2 q and was last revised on 2005/02/05. The original author is Maurizio Codogno, (mau@beatles.cselt.stet.it). It has been largely revised by Johannes Braams and Claudio Beccari

[^22]:    30.46 \initiate@active@char\{"\}\%
    30.47 \addto\extrasitalian\{\bbl@activate\{"\}\languageshorthands\{italian\}\}\%

[^23]:    ${ }^{27}$ The file described in this section has version number v2.0f and was last revised on 2005/03/30. The original author is Claudio Beccari with contributions by Krzysztof Konrad Żelechowski, (kkz@alfa.mimuw.edu.pl)

[^24]:    ${ }^{28}$ Most of these names were kindly suggested by Raffaella Tabacco.

[^25]:    ${ }^{29}$ This word does not exist in "regular" Latin, and it is used just as an example.

[^26]:    ${ }^{30}$ The file described in this section has version number v 1.2 p and was last revised on 2005/03/31. Contributions were made by Jose Pedro Ramalhete (JRAMALHE@CERNVM or Jose-Pedro_Ramalhete@MACMAIL) and Arnaldo Viegas de Lima arnaldo@VNET.IBM.COM.

[^27]:    ${ }^{31}$ The file described in this section has version number v 4.2 b and was last revised on $2004 / 02 / 20$. The original author from v4.0 on is Javier Bezos. Previous versions were by Julio Sánchez.

[^28]:    ${ }^{32}$ The file described in this section has version number $v 2.2 \mathrm{p}$ and was last revised on 2005/03/29.
    ${ }^{33}$ Please 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.

[^29]:    ${ }^{34}$ This seems to be the common practice. See for example: E. Coromina, El 9 Nou: Manual de redacció i estil, Ed. Eumo, Vic, 1993

[^30]:    ${ }^{35} \mathrm{~A}$ 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.

[^31]:    ${ }^{36}$ The macro names $\backslash 11$ and $\backslash L L$ were not taken because of the fact that $\backslash 11$ is already used in mathematical mode.
    ${ }^{37}$ G. Valiente and R. Fuster, Typesetting Catalan Texts with $\mathrm{T}_{\mathrm{E}}$, TUGboat $\mathbf{1 4}(3), 1993$
    ${ }^{38}$ G. Valiente, Modern Catalan Typographical Conventions, TUGboat 16(3), 1995.

[^32]:    ${ }^{39}$ This macro has been borrowed from francais.dtx

[^33]:    ${ }^{40}$ The file described in this section has version number v1．2l and was last revised on 2005／03／30．

[^34]:    ${ }^{41} \mathrm{~A}$ 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.
    ${ }^{42}$ The 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.

[^35]:    ${ }^{43}$ The 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).
    ${ }^{44}$ One 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

[^36]:    ${ }^{45}$ The 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.

[^37]:    ${ }^{46}$ The file described in this section has version number v1.2l and was last revised on 2005/03/31.

[^38]:    ${ }^{47}$ The file described in this section has version number $v 1.3 \mathrm{p}$ and was last revised on 2005/03/30. A contribution was made by Henning Larsen (larsen@cernvm.cern.ch)

[^39]:    ${ }^{48}$ The file described in this section has version number ? and was last revised on ?.

[^40]:    ${ }^{49}$ The file described in this section has version number v 2.0 h 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 shorthand definitions were provided by Rune Kleveland (runekl@math.uio.no).

[^41]:    ${ }^{50}$ The 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).

[^42]:    ${ }^{51}$ The file described in this section has version number v 1.0 c and was last revised on 2004/02/20. It was written by Regnor Jernsletten, (Regnor.Jernsletten@sami.uit.no) or (Regnor.Jernsletten@eunet.no).

[^43]:    ${ }^{52}$ The file described in this section has version number v 1.3 p and was last revised on 2005/03/30. A contribution was made by Mikko KANERVA (KANERVA@CERNVM) and Keranen Reino (KERANEN@CERNVM).

[^44]:    ${ }^{53}$ That 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).
    ${ }^{54}$ It 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).

[^45]:    ${ }^{55}$ The file described in this section has version number v 1.0 h and was last revised on 2005/03/30. The original author is Enn Saar, (saar@aai.ee).

[^46]:    ${ }^{56}$ The file described in this section has version number v1.3l and was last revised on 2005/03/29 A contribution was made by Alan Paić (paica@cernvm.cern.ch).

[^47]:    ${ }^{57}$ The file described in this section has version number v 1.3 k and was last revised on 2005/03/29. Contributions were made by Milos Lokajicek (LOKAJICK@CERNVM)

[^48]:    ${ }^{58}$ The file described in this section has version number v1.2l and was last revised on 2005/03/31.

[^49]:    ${ }^{59}$ The 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).

[^50]:    ${ }^{60}$ The file described in this section has version number v1.3a and was last revised on 2005/03/31. It was written by Jana Chlebikova (chlebik@euromath.dk) and modified by Tobias Schlemmer (Tobias.Schlemmer@web.de).

[^51]:    ${ }^{62}$ The 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 1.0 b (for $\mathrm{LAT}_{\mathrm{E}} \mathrm{X} 2.09$ ), russian. sty version v2.5c (for $\operatorname{LAT}_{\mathrm{E}} \mathrm{X} 2 \varepsilon$ ) and francais.sty version 4.5 c and germanb.sty version 2.5 c were added.

[^52]:    54.302 \declare@shorthand\{russian\}\{"-\}\{\%

[^53]:    ${ }^{63}$ The 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).

[^54]:    ${ }^{64}$ The file described in this section has version number ?. This file was derived from the russianb.dtx version 1.1 g .

[^55]:    56.309 \% \declare@shorthand\{ukrainian\}\{",\}\{\nobreak\hskip.2em\ignorespaces\}

[^56]:    ${ }^{65}$ The file described in this section has version number v1.0f and was last revised on 2005/03/31. It was written by Eduard Werner (edi@kaihh.hanse.de).

[^57]:    ${ }^{66}$ The file described in this section has version number v1.0i and was last revised on 2005/03/31. It was written by Eduard Werner (edi@kaihh.hanse.de).

[^58]:    ${ }^{67}$ The file described in this section has version number v 1.2 m and was last revised on 2005/03/31.
    ${ }^{68}$ Mustafa 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

[^59]:    ${ }^{69}$ The Hebrew language support files described in this section have version number v2.3h and were last revised on 2005/03/30.

[^60]:    60.22 \addto \captionshebrew\{\%
    $60.23 \backslash$ def \prefacename\{\@ensure@R\{\hebmem\hebbet $\backslash$ hebvav $\backslash$ hebalef $\}\} \%$
    $60.24 \backslash$ def $\backslash r e f n a m e\{\backslash @ e n s u r e @ R\{\backslash$ hebresh $\backslash$ hebshin $\backslash$ hebyod $\backslash$ hebmem $\backslash$ hebtav $\backslash$ \%
    $60.25 \backslash$ hebmem $\backslash h e b q o f \backslash h e b v a v \backslash h e b r e s h \backslash h e b v a v \backslash h e b t a v\}\} \%$
    $60.26 \backslash d e f \backslash a b s t r a c t n a m e\{\backslash @ e n s u r e @ R\{\backslash h e b t a v \backslash h e b q o f \backslash h e b t s a d i \backslash h e b y o d \backslash h e b r e s h\}\} \%$
    $60.27 \backslash$ def $\backslash$ bibname $\{\backslash @ e n s u r e @ R\{\backslash$ hebbet $\backslash$ hebyod $\backslash$ hebbet $\backslash$ heblamed $\backslash$ hebyod $\backslash$ hebvav\%
    $60.28 \backslash$ hebgimel \hebresh $\backslash$ hebpe $\backslash$ hebyod $\backslash$ hebhe\}\}\%

[^61]:    ${ }^{70}$ The following description of hebcal package is based on the comments included with original source by the author, Michail Rozman.
    ${ }^{71}$ Please direct any comments, bug reports, questions, etc. about the package to this address.

[^62]:    ${ }^{72}$ The files described in this section have version number v 1.1 b and were last revised on 2004/02/20.

[^63]:    ${ }^{73}$ The files described in this section have version number v1．2c and were last revised on 2005／05／20．

