Hebrew language support from the babel system

Boris Lavva Udi Fogiel

Printed August 22, 2023

Contents

1 The Hebrew language

The file `hebrew.dtx`[1] provides the following packages and files for Hebrew language support:

[1] The Hebrew language support files described in this section have version number v2.4a and were last revised on 2023/08/22.

`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 LaTeX 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 the files were moved to the hebrew-fonts package):

`hebinp.dtx` provides Hebrew input encodings, such as ISO 8859-8, MS Windows codepage 1255 or IBM PC codepage 862.

`hebrew.fdd` contains Hebrew font encodings, related font definition files and hebfont package that provides Hebrew font switching commands.

LaTeX 2.09 compatibility files are included with `heb209.dtx` and gives possibility to compile existing LaTeX 2.09 Hebrew documents with small (if any) changes (see Section 2 on page 45 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

1.1 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.

1.2 The DOCSTRIP modules

The following modules are used in the implementation to direct DOCSTRIP in generating external files:

driver	produce a documentation driver file
hebrew	produce Hebrew language support file
rightleft	create right-to-left support file
calendar	create Hebrew calendar package

A typical DOCSTRIP command file would then have entries like:

```
\generateFile{hebrew.ldf}{t}{\from{hebrew.dtx}{hebrew}}
```

1.3 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.

```
⟨*hebrew⟩
\LdfInit{hebrew}{captionshebrew}
```

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

```
\ifx\l@hebrew\@undefined
  \@nopatterns{Hebrew}%
  \adddialect\l@hebrew0
\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 LaTeX at end of this package.

Also note that if you want to use the encoding `HE8` , you should define the following in your document, *before loading babel*:

```
\def\HeblatexEncoding{HE8}
\def\HeblatexEncodingFile{he8enc}
```

```
% \changes{hebrew~2.4}{2023/08/16}{Make NHE8 the default font encoding}
% \changes{hebrew~2.4a}{2023/08/16}{Better handeling for font encoding}
\@ifundefined{HeblatexEncoding}
  {
\@ifl@aded{def}{nhe8enc}
{
  \providecommand*\HeblatexEncoding{NHE8}
  \providecommand*\HeblatexEncodingFile{nhe8enc}
}
```

```
{
  \@ifl@aded{def}{he8enc}
   {
     \providecommand*\HeblatexEncoding{HE8}
     \providecommand*\HeblatexEncodingFile{he8enc}
   }
   {
     \@ifl@aded{def}{lheenc}
      {
        \providecommand*\HeblatexEncoding{LHE}
        \providecommand*\HeblatexEncodingFile{lheenc}
      }
      {
        \providecommand*\HeblatexEncoding{NHE8}
      \providecommand*\HeblatexEncodingFile{nhe8enc}

      }
   }
}
  }{}
\@ifl@aded{def}{\HeblatexEncodingFile}{}
{\input{\HeblatexEncodingFile.def}}

\newcommand{\heblatex@set@encoding}[2]{
}

\edef\hebrewencoding{\HeblatexEncoding}
\def\heb@NHEenc{NHE8}
\ifx\heb@NHEenc\hebrewencoding
\def\@brackets#1#2#3{#1#2#3}
\fi
```

The next step consists of defining commands to switch to (and from) the Hebrew language.

`\hebrewhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`. They are set to 2.

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

`\captionshebrew` The macro `\captionshebrew` replaces all captions used in the four standard document classes provided with LaTeX 2ε with their Hebrew equivalents.

```
\addto\captionshebrew{%
  \def\prefacename{\@ensure@R{\hebmem\hebbet\hebvav\hebalef}}%
  \def\refname{\@ensure@R{\hebresh\hebshin\hebyod\hebmem\hebtav\ %
    \hebmem\hebqof\hebvav\hebresh\hebvav\hebtav}}%
  \def\abstractname{\@ensure@R{\hebtav\hebqof\hebtsadi\hebyod\hebresh}}%
  \def\bibname{\@ensure@R{\hebbet\hebyod\hebbet\heblamed\hebyod\hebvav%
    \hebgimel\hebresh\hebpe\hebyod\hebhe}}%
  \def\chaptername{\@ensure@R{\hebpe\hebresh\hebqof}}%
  \def\appendixname{\@ensure@R{\hebnun\hebsamekh\hebpe\hebhet}}%
  \def\contentsname{\@ensure@R{%
    \hebtav\hebvav\hebkaf\hebfinalnun\ %
    \hebayin\hebnun\hebyod\hebyod\hebnun\hebyod\hebfinalmem}}%
  \def\listfigurename{\@ensure@R{%
```

```
    \hebresh\hebshin\hebyod\hebmem\hebtav\ %
    \hebalef\hebyod\hebvav\hebresh\hebyod\hebfinalmem}}%
  \def\listtablename{\@ensure@R{%
    \hebresh\hebshin\hebyod\hebmem\hebtav\
    \hebtet\hebbet\heblamed\hebalef\hebvav\hebtav}}%
  \def\indexname{\@ensure@R{\hebmem\hebpe\hebtav\hebhet}}%
  \def\figurename{\@ensure@R{\hebalef\hebyod\hebvav\hebresh}}%
  \def\tablename{\@ensure@R{\hebtet\hebbet\heblamed\hebhe}}%
  \def\partname{\@ensure@R{\hebhet\heblamed\hebqof}}%
  \def\enclname{\@ensure@R{\hebresh\hebtsadi"\hebbet}}%
  \def\ccname{\@ensure@R{\hebhe\hebayin\hebtav\hebqof\hebyod%
    \hebfinalmem}}%
  \def\headtoname{\@ensure@R{\hebalef\heblamed}}%
  \def\pagename{\@ensure@R{\hebayin\hebmem\hebvav\hebdalet}}%
  \def\psname{\@ensure@R{\hebnun.\hebbet.}}%
  \def\seename{\@ensure@R{\hebresh\hebalef\hebhe}}%
  \def\alsoname{\@ensure@R{\hebresh\hebalef\hebhe \hebgimel%
    \hebmemesof}}%
  \def\proofname{\@ensure@R{\hebhe\hebvav\hebkaf\hebhet\hebhe}}
  \def\glossaryname{\@ensure@L{Glossary}}% <-- Needs translation
}
```

`\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

```
\@ifclassloaded{seminar}{%
  \def\slidelabel{\bf \if@rl\R{\hebshin\hebqof\hebfinalpe{} \theslide}%
                      \else\L{Slide \theslide}%
                      \fi}%
}{}
```

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.

```
\def\hebmonth#1{%
  \ifcase#1\or \hebyod\hebnun\hebvav\hebalef\hebresh\or %
     \hebpe\hebbet\hebresh\hebvav\hebalef\hebresh\or %
     \hebmem\hebresh\hebfinaltsadi\or %
     \hebalef\hebpe\hebresh\hebyod\heblamed\or %
     \hebmem\hebalef\hebyod\or \hebyod\hebvav\hebnun\hebyod\or %
     \hebyod\hebvav\heblamed\hebyod\or %
     \hebalef\hebvav\hebgimel\hebvav\hebsamekh\hebtet\or %
     \hebsamekh\hebpe\hebtet\hebmem\hebbet\hebresh\or %
     \hebalef\hebvav\hebqof\hebtet\hebvav\hebbet\hebresh\or %
     \hebnun\hebvav\hebbet\hebmem\hebbet\hebresh\or %
     \hebdalet\hebtsadi\hebmem\hebbet\hebresh\fi}
```

`\hebdate` The macro `\hebdate{`*day*`}{`*month*`}{`*year*`}` translates a given Gregorian date to Hebrew.

```
\def\hebdate#1#2#3{%
  \beginR\beginL\number#1\endL\ \hebbet\hebmonth{#2}
         \beginL\number#3\endL\endR}
```

\hebday The macro \hebday will replace \today command when in Hebrew mode.

```
\def\hebday{\hebdate{\day}{\month}{\year}}
```

\datehebrew The macro \datehebrew redefines the command \today to produce Gregorian dates in Hebrew. It uses the macro \hebday.

```
\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 TEX's restricted horizontal mode, the \L{*latin text*} and \R{*hebrew 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.

```
\addto\extrashebrew{%
  \tohebrew%
  \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.

```
\addto\noextrashebrew{%
  \fromhebrew%
  \ifhmode\ifinner\else\beginL\fi\fi}
```

Generally, we can switch to- and from- Hebrew by means of standard babel-defined commands, for example,

```
\selectlanguage{hebrew}
```

or

```
\begin{otherlanguage}{hebrew}
    some Hebrew text
\end{otherlanguage}
```

Now we define two additional commands that offer the possibility to switch to and from Hebrew language. These commands are backward compatible with the previous versions of hebrew.sty.

\sethebrew The command \sethebrew will switch from the current font encoding to the 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.

```
\def\sethebrew{\setrllanguage{hebrew}}
\def\unsethebrew{\unsetrllanguage{hebrew}}
```

\hebrewtext \nohebrewtext The following two commands are *obsolete* and work only in LaTeX2.09 compatibility mode. They are synonyms of `\sethebrew` and `\unsethebrew` defined above.

```
\if@compatibility
  \let\hebrewtext=\sethebrew
  \let\nohebrewtext=\unsethebrew
\fi
```

\tohebrew \fromhebrew These two commands change only the current font encoding to- and from- Hebrew encoding. Their implementation uses `\@torl{`*language name*`}` and `\@fromrl` macros defined in `rlbabel.def` file. Both commands may be useful *only* for package and class writers, not for regular users.

```
\def\tohebrew{\@torl{hebrew}}%
\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.

```
\def\@@hebrew#1{\beginR{{\tohebrew#1}}\endR}
\def\@hebrew{\protect\@@hebrew}
```

1.3.1 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 \if@gim@final Hebrew numbers can be written in various styles: with or without apostrophes, and with the letters kaf, mem, nun, pe, tsadi as either final or initial forms when they are the last letters in the sequence. We provide two flags to set the style options.

```
\newif\if@gim@apost  % whether we print apostrophes
\newif\if@gim@final  % whether we use final or initial letters
```

\hebrewnumeral \Hebrewnumeral \Hebrewnumeralfinal The commands that print a Hebrew number must specify the style locally: relying on a global style option could cause a counter to print in an inconsistent manner—for instance, page numbers might appear in different styles if the global style option changed mid-way through a document. The commands only allow three of the four possible flag combinations (I do not know of a use that requires the combination of final letters and no apostrophes –RA).

Each command sets the style flags and calls `\@hebrew@numeral`. Double braces are used in order to protect the values of `\@tempcnta` and `\@tempcntb`, which are changed by this call; they also keep the flag assignments local (this is not important because the global values are never used).

```
\newcommand*{\hebrewnumeral}[1]      % no apostrophe, no final letters
 {{\@gim@finalfalse\@gim@apostfalse\@hebrew@numeral{#1}}}
\newcommand*{\Hebrewnumeral}[1]      % apostrophe, no final letters
 {{\@gim@finalfalse\@gim@aposttrue\@hebrew@numeral{#1}}}
\newcommand*{\Hebrewnumeralfinal}[1] % apostrophe, final letters
 {{\@gim@finaltrue\@gim@aposttrue\@hebrew@numeral{#1}}}
```

\alph
\@alph
\Alph
\@Alph
\Alphfinal
\@Alphfinal

Counter-printing commands are based on the above commands. The natural name for the counter-printing commands is \alph, because Hebrew numerals are the only way to represent numbers with Hebrew letters (kaf always means 20, never 11). Hebrew has no uppercase letters, hence no need for the familiar meaning of \Alph; we therefore define \alph to print counters as Hebrew numerals without apostrophes, and \Alph to print with apostrophes. A third form, \Alphfinal, is provided to print with apostrophes and final letters, as is required for Hebrew year designators. The commands \alph and \Alph are defined in latex.ltx, and we only need to redefine the internal commands \@alph and \@Alph; for \Alphfinal we need to provide both a wrapper and an internal command. The counter printing commands are made semi-robust: without the \protect, commands like \theenumii break (I'm not quite clear on why this happens, –RA); at the same time, we cannot make the commands too robust (e.g. with \DeclareRobustCommand) because this would enter the command name rather than its value into files like .aux, .toc etc. The old meanings of meaning of \@alph and \@Alph are saved upon entering Hebrew mode and restored upon exiting it.

```
\addto\extrashebrew{%
  \let\saved@alph=\@alph%
  \let\saved@Alph=\@Alph%
  \renewcommand*{\@alph}[1]{\protect\hebrewnumeral{\number#1}}%
  \renewcommand*{\@Alph}[1]{\protect\Hebrewnumeral{\number#1}}%
  \def\Alphfinal#1{\expandafter\@Alphfinal\csname c@#1\endcsname}%
  \providecommand*{\@Alphfinal}[1]{\protect\Hebrewnumeralfinal{\number#1}}}
\addto\noextrashebrew{%
  \let\@alph=\saved@alph%
  \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 LaTeX labels need to be redefined for appropriate cross-referencing, see below. LaTeX default class files specify \Alph for the fourth-level enumerate level, this should probably be changed. Also, the way labels get flushed left by default looks inappropriate for Hebrew numerals, so we should redefine \labelenumii as well as \labelenumiv (presently not implemented).

\theenumii
\theenumiv
\label

Cross-references to counter labels need to be printed according to the language environment in which a label was issued, not the environment in which it is called: 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 IvriTeX list. We therefore redefine \theenumii and \theenumiv, so that an explicit language instruction gets written to the .aux file.

```
\renewcommand{\theenumii}
   {\if@rl\protect\hebrewnumeral{\number\c@enumii}%
    \else\protect\L{\protect\@@alph{\number\c@enumii}}\fi}
\renewcommand{\theenumiv}
   {\if@rl\protect\Hebrewnumeral{\number\c@enumiv}%
    \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`$\cdots$`\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.

```
\ifx\heb@NHEenc\HeblatexEncoding
\def\label#1{\@bsphack
  \if@rl
    \protected@write\@auxout{}%
         {\string\newlabel{#1}{{\beginR\@currentlabel\endR}{\thepage}}}%
  \else
    \protected@write\@auxout{}%
         {\string\newlabel{#1}{{\beginL\@currentlabel\endL}{\thepage}}}%
  \fi
  \@esphack}
\else
\def\label#1{\@bsphack
  \if@rl
    \def\p@enumiii{\p@enumii)\theenumii(}%
    \protected@write\@auxout{}%
         {\string\newlabel{#1}{{\beginR\@currentlabel\endR}{\thepage}}}%
  \else
    \def\p@enumiii{\p@enumii(\theenumii)}%
    \protected@write\@auxout{}%
         {\string\newlabel{#1}{{\beginL\@currentlabel\endL}{\thepage}}}%
  \fi
  \@esphack}
\fi
```

NOTE: it appears that the definition of `\label` is language-independent and thus belongs in rlbabel.def, but this is not the case. The decision to typeset label numerals in the font of the surrounding language is reasonable for Hebrew, because mixed-font (1b) and (2dalet) are somewhat acceptable. The same may not be acceptable for Arabic, whose numeral glyphs are radically different from those in the Latin fonts. The decision about the direction may also be different for Arabic, which is more right-to-left oriented than Hebrew (two examples: dates like 15/6/2003 are written left-to-right in Hebrew but right-to-left in Arabic; equations like $1 + 2 = 3$ are written left-to-right in Hebrew but right-to-left in Arabic elementary school textbooks using Arabic numeral glyphs). My personal hunch is that a label like (1b) in an Arabic text would be typeset left-to-right if the '1' is a Western glyph, but right-to-left if the '1' is an Arabic glyph. But this is just a guess, I'd have to ask Arab typesetters to find the correct answer. –RA.

`\appendix` The following code provides for the proper printing of appendix numbers in tables of contents. Section and chapter headings are normally bilingual: regardless of the text language, the author supplies each section/chapter with two headings—one for the Hebrew table of contents and one for the Latin table of contents. It makes sense that the label should be a Latin letter in the Latin table of contents and a Hebrew letter in the Hebrew table of contents. The definition is similar to that of `\theenumii` and `\theenumiv` above, but additional `\protect` commands ensure that the entire condition is written 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 LaTeX [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.

```
\@ifclassloaded{letter}{}{%
\@ifclassloaded{slides}{}{%
  \let\@@appendix=\appendix%
  \@ifclassloaded{article}{%
    \renewcommand\appendix{\@@appendix%
      \renewcommand\thesection
        {\protect\if@rl\protect\Hebrewnumeral{\number\c@section}%
         \protect\else\@@Alph\c@section\protect\fi}}
   {\renewcommand\appendix{\@@appendix%
      \renewcommand\thechapter
        {\protect\if@rl\protect\Hebrewnumeral{\number\c@chapter}%
         \protect\else\@@Alph\c@chapter\protect\fi}}}}}
```

QUESTION: is this also the appropriate way to refer to an appendix in the text, or should we retain the original label the same way we did with `enumerate` labels? ANOTHER QUESTION: are similar redefinitions needed for other counters that generate texts in bilingual lists like `.lof/.fol` and `.lot/.tol`? –RA.

`\@hebrew@numeral` The command `\@hebrew@numeral` prints a Hebrew number. The groups of thousands, millions, billions are separated by apostrophes and typeset without apostrophes or final letters; the remainder (under 1000) is typeset conventionally, with the selected styles for apostrophes and final letters. The function calls on `\gim@no@mil`

to typeset each three-digit block. The algorithm is recursive, but the maximum recursion depth is 4 because TEX only allows numbers up to $2^{31} - 1 = 2{,}147{,}483{,}647$. The typesetting routine is wrapped in `\@hebrew` in order to ensure that numbers are always typeset in Hebrew mode.

Initialize: `\@tempcnta` holds the value, `\@tempcntb` is used for calculations.

```
\newcommand*{\@hebrew@numeral}[1]
{\@hebrew{\@tempcnta=#1\@tempcntb=#1\relax
 \divide\@tempcntb by 1000
```

If we're under 1000, call `\gim@nomil`

```
 \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.

```
 \else{\@gim@apostfalse\@gim@finalfalse\@hebrew@numeral\@tempcntb}'%
      \multiply\@tempcntb by 1000\relax
      \advance\@tempcnta by -\@tempcntb\relax
      \gim@nomil\@tempcnta\relax
 \fi
}}
```

NOTE: is it the case that 15,000 and 16,000 are written as yod-he and yod-vav, rather than tet-vav and tet-zayin? This vaguely rings a bell, but I'm not certain. If this is the case, then the current behavior is incorrect and should be changed. –RA.

`\gim@nomil` The command `\gim@nomil` typesets an integer between 0 and 999 (for 0 it typesets nothing). The code has been modified from the old `hebcal.sty` (appropriate credits—Boris Lavva and Michail Rozman ?). `\@tempcnta` holds the total value that remains to be typeset. At each stage we find the highest valued letter that is less than or equal to `\@tempcnta`, and call on `\gim@print` to subtract this value and print the letter.

Initialize: `\@tempcnta` holds the value, there is no previous letter.

```
\newcommand*{\gim@nomil}[1]{\@tempcnta=#1\@gim@prevfalse
```

Find the hundreds digit.

```
 \@tempcntb=\@tempcnta\divide\@tempcntb by 100\relax % hundreds digit
 \ifcase\@tempcntb                    % print nothing if no hundreds
    \or\gim@print{100}{\hebqof}%
    \or\gim@print{200}{\hebresh}%
    \or\gim@print{300}{\hebshin}%
    \or\gim@print{400}{\hebtav}%
    \or\hebtav\@gim@prevtrue\gim@print{500}{\hebqof}%
    \or\hebtav\@gim@prevtrue\gim@print{600}{\hebresh}%
    \or\hebtav\@gim@prevtrue\gim@print{700}{\hebshin}%
    \or\hebtav\@gim@prevtrue\gim@print{800}{\hebtav}%
    \or\hebtav\@gim@prevtrue\hebtav\gim@print{900}{\hebqof}%
 \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.

```
 \@tempcntb=\@tempcnta\divide\@tempcntb by 10\relax      % tens digit
 \ifcase\@tempcntb                        % print nothing if no tens
```

```
    \or                                 % number between 10 and 19
          \ifnum\@tempcnta = 16 \gim@print {9}{\hebtet}% tet-zayin
       \else\ifnum\@tempcnta = 15 \gim@print {9}{\hebtet}% tet-vav
       \else                      \gim@print{10}{\hebyod}%
          \fi % \@tempcnta = 15
          \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.

```
    \or\gim@print{20}{\if@gim@final\hebfinalkaf\else\hebkaf\fi}%
    \or\gim@print{30}{\heblamed}%
    \or\gim@print{40}{\if@gim@final\hebfinalmem\else\hebmem\fi}%
    \or\gim@print{50}{\if@gim@final\hebfinalnun\else\hebnun\fi}%
    \or\gim@print{60}{\hebsamekh}%
    \or\gim@print{70}{\hebayin}%
    \or\gim@print{80}{\if@gim@final\hebfinalpe\else\hebpe\fi}%
    \or\gim@print{90}{\if@gim@final\hebfinaltsadi\else\hebtsadi\fi}%
  \fi
```

Print the ones digit.

```
  \ifcase\@tempcnta                     % print nothing if no ones
    \or\gim@print{1}{\hebalef}%
    \or\gim@print{2}{\hebbet}%
    \or\gim@print{3}{\hebgimel}%
    \or\gim@print{4}{\hebdalet}%
    \or\gim@print{5}{\hebhe}%
    \or\gim@print{6}{\hebvav}%
    \or\gim@print{7}{\hebzayin}%
    \or\gim@print{8}{\hebhet}%
    \or\gim@print{9}{\hebtet}%
  \fi
}
```

`\gim@print` `\if@gim@prev` The actual printing routine typesets a digit with the appropriate apostrophes: 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.

```
\newif\if@gim@prev % flag if a previous letter has been typeset
```

For each letter, we first subtract its value from the total. Then,

- if the result is zero then this is the last letter; we check the flag to see if this is the only letter and print it with the appropriate apostrophe;
- if the result is not zero then there remain additional letters to be typeset; we print without an apostrophe and set the 'previous letter' flag.

`\@tempcnta` holds the total value that remains to be typeset. We first deduct the letter's value from `\@tempcnta`, so `\@tempcnta` is zero if and only if this is the last letter.

```
\newcommand*{\gim@print}[2]{%   #2 is a letter, #1 is its value.
  \advance\@tempcnta by -#1\relax% deduct the value from the remainder
```

If this is the last letter, we print with the appropriate apostrophe (depending on the style option): if there is a preceding letter, print `"x` if the style calls for apostrophes, `x` if it doesn't; otherwise, this is the only letter: print `x'` if the style calls for apostrophes, `x` if it doesn't.

```
\ifnum\@tempcnta=0% if this is the last letter
   \if@gim@prev\if@gim@apost"\fi#2%
   \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.

```
\else{\@gim@finalfalse#2}\@gim@prevtrue\fi}
```

`\hebr` `\gim` The older Hebrew counter commands `\hebr` and `\gim` are retained in order to keep older documents from breaking. They are set to be equivalent to `\alph`, and their use is deprecated. Note that `\hebr` gives different results than it had in the past—it now typesets 11 as yod-alef rather than kaf.

```
\let\hebr=\alph
\let\gim=\alph
```

For backward compatibility with 'older' hebrew.sty packages, we define Hebrew equivalents of some useful LaTeX commands. Note, however, that 8-bit macros defined in Hebrew are no longer supported.

```
\def\hebcopy{\protect\R{\hebhe\hebayin\hebtav\hebqof}}
\def\hebincl{\protect\R{\hebresh\hebtsadi"\hebbet}}
\def\hebpage{\protect\R{\hebayin\hebmem\hebvav\hebdalet}}
\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.

```
\def\hadgesh#1{\leavevmode\setbox0=\hbox{#1}%
  \kern-.025em\copy0\kern-\wd0
  \kern.05em\copy0\kern-\wd0
  \kern-.025em\raise.0433em\box0 }
```

`\piska` and `\piskapiska` sometimes used in 'older' hebrew sources, and should not be used in LaTeX 2_ε.

```
\if@compatibility
  \def\piska#1{\item{#1}\hangindent=-\hangindent}
  \def\piskapiska#1{\itemitem{#1}\hangindent=-\hangindent}
\fi
```

The following commands are simply synonyms for the standard ones, provided with LaTeX 2_ε.

```
\let\makafgadol=\textendash
\let\makafanak=\textemdash
\let\geresh=\textquoteright
\let\opengeresh=\textquoteright
\let\closegeresh=\textquoteleft
\let\openquote=\textquotedblright
\let\closequote=\textquotedblleft
\let\leftquotation=\textquotedblright
\let\rightquotation=\textquotedblleft
```

We need to ensure that Hebrew is used as the default right-to-left language at `\begin{document}`. The mechanism of defining the `\@rllanguagename` is the same as in babel's `\languagename`: the last right-to-left language in the `\usepackage{babel}` line is set as the default right-to-left language at document beginning.

For example, the following code:

```
\usepackage[russian,hebrew,arabic,greek,english]{babel}
```

will set the Arabic language as the default right-to-left language and the English language as the default language. As a result, the commands `\L{}` and `\embox{}` will use English and `\R{}` and `\hmbox{}` will use Arabic by default. These defaults can be changed with the next `\sethebrew` or `\selectlanguage{`*language name*`}` command.

```
\AtBeginDocument{\def\@rllanguagename{hebrew}}
```

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

```
\ldf@finish{hebrew}
⟨/hebrew⟩
```

1.4 Right to left support

This file rlbabel.def defines necessary bidirectional macro support for LaTeX 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 TeX extensions, namely TeX primitives `\beginL`, `\endL` and `\beginR`, `\endR`, currently implemented only in ε-TeX and in TeX--XeT.

If ε-TeX is used, we should switch it to the *enhanced* mode:

```
⟨*rightleft⟩
\ifx\TeXXeTstate\undefined\else%
    \TeXXeTstate=1
\fi
```

Note, that ε-TeX's format file should be created for *extended* mode. Mode can be checked by running ε-TeX on some TeX file, for example:

```
This is e-TeX, Version 3.14159-1.1 (Web2c 7.0)
entering extended mode
```

The second line should be `entering extended mode`.

We check if user uses Right-to-Left enabled engine instead of regular Knuth's TeX:

```
\ifx\beginL\@undefined%
    \newlinechar`\^^J
    \typeout{^^JTo avoid this error message,^^J%
      run TeX--XeT or e-TeX engine instead of regular TeX.^^J}
    \errmessage{Right-to-Left Support Error: use TeX--XeT or e-TeX
      engine}%
\fi
```

1.4.1 Switching from LR to RL mode and back

`\@torl` and `\@fromrl` are called each time the horizontal direction changes. They do all that is necessary besides changing the direction. Currently their task is to change the encoding information and mode (condition `\if@rl`). They should not normally be called by users: user-level macros, such as `\sethebrew` and `\unsethebrew`, as well as babel's `\selectlanguage` are defined in language-definition files and should be used to change default language (and direction).

Local direction changing commands (for small pieces of text): `\L{}`, `\R{}`, `\embox{}` and `\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.

```
\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.

```
\AtBeginDocument{% Here we set the main document direction
  \newif\if@rlmain%
  \if@rl% e.g: if the options to babel were [english,hebrew]
    \@rlmaintrue%
  \else%  e.g: if the options to babel were [hebrew,english]
    \@rlmainfalse%
  \fi%
}
```

`\@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`.

```
\DeclareRobustCommand{\@torl}[1]{%
  \if@rl\else%
     \let\lr@encodingdefault=\encodingdefault%
  \fi%
  \def\@rllanguagename{#1}%
  \def\encodingdefault{\csname#1encoding\endcsname}%
  \fontencoding{\encodingdefault}%
  \selectfont%
  \@rltrue}
```

`\@fromrl` Opposite to `\@torl`, switches current direction to Left-to-Right: restores saved Left-to-Right encoding (`\lr@encodingdefault`) and changes direction.

```
\DeclareRobustCommand{\@fromrl}{%
```

```
\if@rl%
   \let\encodingdefault=\lr@encodingdefault%
\fi%
\fontencoding{\encodingdefault}%
\selectfont%
\@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`.

```
\expandafter\def\csname selectlanguage \endcsname#1{%
  \edef\languagename{%
    \ifnum\escapechar=\expandafter`\string#1\@empty
    \else \string#1\@empty\fi}%
  \@@selectlanguage{\languagename}}
```

`\@@selectlanguage` This new internal macro redefines a final part of the standard babel's `\selectlanguage` implementation.

Standard LaTeX provides us with 3 tables: Table of Contents (`.toc`), List of Figures (`.lof`), and List of Tables (`.lot`). In multi-lingual texts mixing Left-to-Right languages with Right-to-Left ones, the use of various directions in one table results in very ugly output. Therefore, these 3 standard tables will be used now only for Left-to-Right languages, and we will add 3 Right-to-Left tables (their extensions are simply reversed ones): RL Table of Contents (`.cot`), RL List of Figures (`.fol`), and RL List of Tables (`.lof`).

```
\def\@@selectlanguage#1{%
  \select@language{#1}%
  \if@filesw
     \protected@write\@auxout{}{\string\select@language{#1}}%
     \if@rl%
       \addtocontents{cot}{\xstring\select@language{#1}}%
       \addtocontents{fol}{\xstring\select@language{#1}}%
       \addtocontents{tol}{\xstring\select@language{#1}}%
     \else%
       \addtocontents{toc}{\xstring\select@language{#1}}%
       \addtocontents{lof}{\xstring\select@language{#1}}%
       \addtocontents{lot}{\xstring\select@language{#1}}%
     \fi%
  \fi}
```

`\setrllanguage` `\unsetrllanguage` The `\setrllanguage` and `\unsetrllanguage` pair of macros is proved to very useful in bilingual texts, for example, in Hebrew-English texts. The language-specific commands, for example, `\sethebrew` and `\unsethebrew` use these macros as basis.

Implementation saves and restores other language in `\other@languagename` variable, and uses internal macro `\@@selectlanguage`, defined above, to switch between languages.

```
\let\other@languagename=\languagename
\DeclareRobustCommand{\setrllanguage}[1]{%
   \if@rl\else%
     \let\other@languagename=\languagename%
   \fi%
```

```
      \def\languagename{#1}%
      \@@selectlanguage{\languagename}}

\DeclareRobustCommand{\unsetrllanguage}[1]{%
   \if@rl%
     \let\languagename=\other@languagename%
   \fi
   \@@selectlanguage{\languagename}}
```

\L \R \HeblatexRedefineL

Macros for changing direction, originally taken from TUGboat. Usage: \L{*Left to Right text*} and \R{*Right to Left text*}. Numbers should also be enclosed in \L{}, as in \L{123}.

Note, that these macros do not receive language name as parameter. Instead, the saved \@rllanguagename will be used. We assume that each Right-to-Left language defines \to*languagename* and \from*languagename* macros in language definition file, for example, for Hebrew: \tohebrew and \fromhebrew macros in hebrew.ldf file.

The macros \L and \R include 'protect' to to make them robust and allow use, for example, in tables.

Due to the fact that some packages have different definitions for \L the macro \HeblatexRedefineL is provided to overide them. This may be required with hyperref, for instance.

```
\let\next=\
\def\HeblatexRedefineL{%
  \def\L{\protect\pL}%
}
\HeblatexRedefineL
\def\pL{\protect\afterassignment\moreL \let\next= }
\def\moreL{\bracetext \aftergroup\endL \beginL\csname
  from\@rllanguagename\endcsname}

\def\R{\protect\pR}
\def\pR{\protect\afterassignment\moreR \let\next= }
\def\moreR{\bracetext \aftergroup\endR \beginR\csname
  to\@rllanguagename\endcsname}
\def\bracetext{\ifcat\next{\else\ifcat\next}\fi
  \errmessage{Missing left brace has been substituted}\fi \bgroup}
\everydisplay{\if@rl\aftergroup\beginR\fi }
```

\@ensure@R \@ensure@L

Two small internal macros, a-la \ensuremath

```
\def\@ensure@R#1{\if@rl#1\else\R{#1}\fi}
\def\@ensure@L#1{\if@rl\L{#1}\else#1\fi}
```

Take care of Right-to-Left indentation in every paragraph. The old approach had conflicts with amsthm, so I addapted the code from the bidi package by Vafa Khalighi.

```
\def\heb@rl@everypar{{\setbox\z@\lastbox\if@rl\beginR\else\beginL\fi\ifvoid\z@\else\usebox\z
\let\heb@o@everypar=\everypar
\newtoks\heb@n@everypar
\heb@n@everypar\expandafter{\the\heb@o@everypar}
\heb@o@everypar{\heb@rl@everypar\the\heb@n@everypar}
\let\everypar=\heb@n@everypar
```

\hmbox \embox Useful vbox commands. All text in math formulas is best enclosed in these: LR text in \embox and RL text in \hmbox. \mbox{} is useless for both cases, since it typesets in Left-to-Right even for Right-to-Left languages (additions by Yaniv Bargury).

```
\newcommand{\hmbox}[1]{\mbox{\R{#1}}}
\newcommand{\embox}[1]{\mbox{\L{#1}}}
```

\@brackets When in Right-to-Left mode, brackets should be swapped. This macro receives 3 parameters: left bracket, content, right bracket. Brackets can be square brackets, braces, or parentheses.

```
\def\@brackets#1#2#3{\protect\if@rl #3#2#1\protect\else
  #1#2#3\protect\fi}
```

\@number \@latin \@number preserves numbers direction from Left to Right. \@latin in addition switches current encoding to the latin.

```
\def\@@number#1{\ifmmode\else\beginL\fi#1\ifmmode\else\endL\fi}
\def\@@latin#1{\@@number{{\@fromrl#1}}}
\def\@number{\protect\@@number}
\def\@latin{\protect\@@latin}
```

1.4.2 Counters

To make counter references work in Right to Left text, we need to surround their original definitions with an \@number{...} or \@latin{...}. Note, that language-specific counters, such as \hebr or \gim are provided with language definition file.

We start with saving the original definitions:

```
\let\@@arabic=\@arabic
\let\@@roman=\@roman
\let\@@Roman=\@Roman
\let\@@alph=\@alph
\let\@@Alph=\@Alph
```

\@arabic \@roman \@Roman Arabic and roman numbers should be from Left to Right. In addition, roman numerals, both lower- and upper-case should be in latin encoding.

```
\def\@arabic#1{\@number{\@@arabic#1}}
\def\@roman#1{\@latin{\@@roman#1}}
\def\@Roman#1{\@latin{\@@Roman#1}}
```

\arabicnorl This macro preserves the original definition of \arabic (overrides the overriding of \@arabic)

```
\def\arabicnorl#1{\expandafter\@@arabic\csname c@#1\endcsname}
```

\make@lr In Right to Left documents all counters defined in the standard document classes *article*, *report* and *book* provided with LaTeX 2_ε, such as \thesection, \thefigure, \theequation should be typed as numbers from left to right. To ensure direction, we use the following \make@lr{*counter*} macro:

```
\def\make@lr#1{\begingroup
    \toks@=\expandafter{#1}%
    \edef\x{\endgroup
  \def\noexpand#1{\noexpand\@number{\the\toks@}}}%
  \x}
```

```
\@ifclassloaded{letter}{}{%
  \@ifclassloaded{slides}{}{%
    \make@lr\thesection
    \make@lr\thesubsection
    \make@lr\thesubsubsection
    \make@lr\theparagraph
    \make@lr\thesubparagraph
    \make@lr\thefigure
    \make@lr\thetable
  }
  \make@lr\theequation
}
```

1.4.3 Preserving logos

Preserve TEX, LATEX and LATEX 2ε logos.

\TeX

```
\let\@@TeX\TeX
\def\TeX{\@latin{\@@TeX}}
```

\LaTeX

```
\let\@@LaTeX\LaTeX
\def\LaTeX{\@latin{\@@LaTeX}}
```

\LaTeXe

```
\let\@@LaTeXe\LaTeXe
\def\LaTeXe{\@latin{\@@LaTeXe}}
```

1.4.4 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. LATEX 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.

```
\def\list#1#2{%
  \ifnum \@listdepth >5\relax
    \@toodeep
  \else
    \global\advance\@listdepth\@ne
  \fi
  \rightmargin\z@
  \listparindent\z@
  \itemindent\z@
  \csname @list\romannumeral\the\@listdepth\endcsname
  \def\@itemlabel{#1}%
  \let\makelabel\@mklab
  \@nmbrlistfalse
  #2\relax
  \@trivlist
```

```
  \parskip\parsep
  \parindent\listparindent
  \advance\linewidth -\rightmargin
  \advance\linewidth -\leftmargin
```

The only change in the macro is the `\if@rl` case:

```
  \if@rl
    \advance\@totalleftmargin \rightmargin
  \else
    \advance\@totalleftmargin \leftmargin
  \fi
  \parshape \@ne \@totalleftmargin \linewidth
  \ignorespaces}
```

`\labelenumii` `\p@enumiii` The `\labelenumii` and `\p@enumiii` commands use *parentheses*. They are revised to work Right-to-Left with the help of `\@brackets` macro defined above.

```
\def\labelenumii{\@brackets(\theenumii)}
\def\p@enumiii{\p@enumii\@brackets(\theenumii)}
```

1.4.5 Tables of moving stuff

Tables of moving arguments: table of contents (`toc`), list of figures (`lof`) and list of tables (`lot`) are handles here. These three default LaTeX tables will be used now exclusively for Left to Right stuff.

Three additional Right-to-Left tables: RL table of contents (`cot`), RL list of figures (`fol`), and RL list of tables (`tol`) are added. These three tables will be used exclusively for Right to Left stuff.

`\@tableofcontents` `\@listoffigures` `\@listoftables` We define 3 new macros similar to the standard LaTeX tables, but with one parameter — table file extension. These macros will help us to define our additional tables below.

```
\@ifclassloaded{letter}{}{% other
\@ifclassloaded{slides}{}{% other
  \@ifclassloaded{article}{% article
    \newcommand\@tableofcontents[1]{%
      \section*{\contentsname\@mkboth%
        {\MakeUppercase\contentsname}%
        {\MakeUppercase\contentsname}}%
      \@starttoc{#1}}
    \newcommand\@listoffigures[1]{%
      \section*{\listfigurename\@mkboth%
        {\MakeUppercase\listfigurename}%
        {\MakeUppercase\listfigurename}}%
      \@starttoc{#1}}
    \newcommand\@listoftables[1]{%
      \section*{\listtablename\@mkboth%
        {\MakeUppercase\listtablename}%
        {\MakeUppercase\listtablename}}%
      \@starttoc{#1}}}%
  {% else report or book
    \newcommand\@tableofcontents[1]{%
      \@restonecolfalse\if@twocolumn\@restonecoltrue\onecolumn%
      \fi\chapter*{\contentsname\@mkboth%
```

```
1.444         {\MakeUppercase\contentsname}%
1.445         {\MakeUppercase\contentsname}}%
1.446       \@starttoc{#1}\if@restonecol\twocolumn\fi}
1.447     \newcommand\@listoffigures[1]{%
1.448       \@restonecolfalse\if@twocolumn\@restonecoltrue\onecolumn%
1.449       \fi\chapter*{\listfigurename\@mkboth%
1.450         {\MakeUppercase\listfigurename}%
1.451         {\MakeUppercase\listfigurename}}%
1.452       \@starttoc{#1}\if@restonecol\twocolumn\fi}
1.453     \newcommand\@listoftables[1]{%
1.454       \if@twocolumn\@restonecoltrue\onecolumn\else\@restonecolfalse\fi%
1.455       \chapter*{\listtablename\@mkboth%
1.456         {\MakeUppercase\listtablename}%
1.457         {\MakeUppercase\listtablename}}%
1.458       \@starttoc{#1}\if@restonecol\twocolumn\fi}}%
```

`\lrtableofcontents` `\lrlistoffigures` `\lrlistoftables` Left-to-Right tables are called now `\lr`*xxx* and defined with the aid of three macros defined above (extensions `toc`, `lof`, and `lot`).

```
1.459   \newcommand\lrtableofcontents{\@tableofcontents{toc}}%
1.460   \newcommand\lrlistoffigures{\@listoffigures{lof}}%
1.461   \newcommand\lrlistoftables{\@listoftables{lot}}%
```

`\rltableofcontents` `\rllistoffigures` `\rllistoftables` Right-to-Left tables will be called `\rl`*xxx* and defined with the aid of three macros defined above (extensions `cot`, `fol`, and `tol`).

```
1.462   \newcommand\rltableofcontents{\@tableofcontents{cot}}%
1.463   \newcommand\rllistoffigures{\@listoffigures{fol}}%
1.464   \newcommand\rllistoftables{\@listoftables{tol}}%
```

`\tableofcontents` `\listoffigures` `\listoftables` Let `\`*xxx* be `\rl`*xxx* if the current direction is Right-to-Left and `\lr`*xxx* if it is Left-to-Right.

```
1.465   \renewcommand\tableofcontents{\if@rl\rltableofcontents%
1.466                                  \else\lrtableofcontents\fi}
1.467   \renewcommand\listoffigures{\if@rl\rllistoffigures%
1.468                                \else\lrlistoffigures\fi}
1.469   \renewcommand\listoftables{\if@rl\rllistoftables%
1.470                              \else\lrlistoftables\fi}}
```

`\@dottedtocline` The following makes problems when making a Right-to-Left tables, since it uses `\leftskip` and `\rightskip` which are both mode dependent.

```
1.471 \def\@dottedtocline#1#2#3#4#5{%
1.472   \ifnum #1>\c@tocdepth \else
1.473     \vskip \z@ \@plus.2\p@
1.474     {\if@rl\rightskip\else\leftskip\fi #2\relax
1.475       \if@rl\leftskip\else\rightskip\fi \@tocrmarg \parfillskip
1.476       -\if@rl\leftskip\else\rightskip\fi
1.477      \parindent #2\relax\@afterindenttrue
1.478      \interlinepenalty\@M
1.479      \leavevmode
1.480      \@tempdima #3\relax
1.481      \advance\if@rl\rightskip\else\leftskip\fi \@tempdima
1.482      \null\nobreak\hskip -\if@rl\rightskip\else\leftskip\fi
1.483      {#4}\nobreak
1.484      \leaders\hbox{$\m@th
```

```
        \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
        mu$}\hfill
     \nobreak
     \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor \beginL#5\endL}%
     \par}%
  \fi}
```

`\l@part` This standard macro was redefined for table of contents since it uses `\rightskip` which is mode dependent.

```
\@ifclassloaded{letter}{}{% other
\@ifclassloaded{slides}{}{% other
\renewcommand*\l@part[2]{%
  \ifnum \c@tocdepth >-2\relax
    \addpenalty{-\@highpenalty}%
    \addvspace{2.25em \@plus\p@}%
    \begingroup
      \setlength\@tempdima{3em}%
      \parindent \z@ \if@rl\leftskip\else\rightskip\fi \@pnumwidth
      \parfillskip -\@pnumwidth
      {\leavevmode
       \large \bfseries #1\hfil \hb@xt@\@pnumwidth{\hss#2}}\par
       \nobreak
         \global\@nobreaktrue
         \everypar{\global\@nobreakfalse\everypar{}}%
    \endgroup
  \fi}}
```

`\@part` Part is redefined to support new Right-to-Left table of contents (`cot`) as well as the Left-to-Right one (`toc`).

```
\@ifclassloaded{article}{% article class
  \def\@part[#1]#2{%
    \ifnum \c@secnumdepth >\m@ne
      \refstepcounter{part}%
      \addcontentsline{toc}{part}{\thepart\hspace{1em}#1}%
      \addcontentsline{cot}{part}{\thepart\hspace{1em}#1}%
    \else
      \addcontentsline{toc}{part}{#1}%
      \addcontentsline{cot}{part}{#1}%
    \fi
    {\parindent \z@ \raggedright
     \interlinepenalty \@M
     \normalfont
     \ifnum \c@secnumdepth >\m@ne
       \Large\bfseries \partname~\thepart
       \par\nobreak
     \fi
     \huge \bfseries #2%
     \markboth{}{}\par}%
    \nobreak
    \vskip 3ex
    \@afterheading}%
}{% report and book classes
  \def\@part[#1]#2{%
    \ifnum \c@secnumdepth >-2\relax
```

```
      \refstepcounter{part}%
      \addcontentsline{toc}{part}{\thepart\hspace{1em}#1}%
      \addcontentsline{cot}{part}{\thepart\hspace{1em}#1}%
    \else
      \addcontentsline{toc}{part}{#1}%
      \addcontentsline{cot}{part}{#1}%
    \fi
    \markboth{}{}%
    {\centering
     \interlinepenalty \@M
     \normalfont
     \ifnum \c@secnumdepth >-2\relax
       \huge\bfseries \partname~\thepart
       \par
       \vskip 20\p@
     \fi
     \Huge \bfseries #2\par}%
     \@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.

```
\def\@sect#1#2#3#4#5#6[#7]#8{%
  \ifnum #2>\c@secnumdepth
    \let\@svsec\@empty
  \else
    \refstepcounter{#1}%
    \protected@edef\@svsec{\@seccntformat{#1}\relax}%
  \fi
  \@tempskipa #5\relax
  \ifdim \@tempskipa>\z@
    \begingroup
      #6{%
        \@hangfrom{\hskip #3\relax\@svsec}%
          \interlinepenalty \@M #8\@@par}%
    \endgroup
    \csname #1mark\endcsname{#7}%
    \addcontentsline{toc}{#1}{%
      \ifnum #2>\c@secnumdepth \else
        \protect\numberline{\csname the#1\endcsname}%
      \fi
      #7}%
    \addcontentsline{cot}{#1}{%
      \ifnum #2>\c@secnumdepth \else
        \protect\numberline{\csname the#1\endcsname}%
      \fi
      #7}%
  \else
    \def\@svsechd{%
      #6{\hskip #3\relax
      \@svsec #8}%
      \csname #1mark\endcsname{#7}%
      \addcontentsline{toc}{#1}{%
        \ifnum #2>\c@secnumdepth \else
          \protect\numberline{\csname the#1\endcsname}%
```

```
        \fi
        #7}%
      \addcontentsline{cot}{#1}{%
        \ifnum #2>\c@secnumdepth \else
          \protect\numberline{\csname the#1\endcsname}%
        \fi
        #7}}%
  \fi
  \@xsect{#5}}
```

`\@caption` Caption was redefined from the latex.ltx file. It is changed to support Left-to-Right list of figures and list of tables (`lof` and `lot`) as well as new Right-to-Left lists (`fol` and `tol`) simultaneously.

```
\long\def\@caption#1[#2]#3{%
  \par
  \addcontentsline{\csname ext@#1\endcsname}{#1}%
    {\protect\numberline{\csname the#1\endcsname}%
    {\ignorespaces #2}}%
  \def\@fignm{figure}
  \ifx#1\@fignm\addcontentsline{fol}{#1}%
     {\protect\numberline{\csname the#1\endcsname}%
     {\ignorespaces #2}}\fi%
  \def\@tblnm{table}
  \ifx#1\@tblnm\addcontentsline{tol}{#1}%
     {\protect\numberline{\csname the#1\endcsname}%
     {\ignorespaces #2}}\fi%
  \begingroup
    \@parboxrestore
    \if@minipage
      \@setminipage
    \fi
    \normalsize
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
  \endgroup}
```

`\l@chapter` This standard macro was redefined for table of contents since it uses `\rightskip` which is mode dependent.

```
\@ifclassloaded{letter}{}{%
\@ifclassloaded{slides}{}{%
  \@ifclassloaded{article}{}{%
  \@ifundefined{l@chapter}{}{%
    \renewcommand*\l@chapter[2]{%
      \ifnum \c@tocdepth >\m@ne
      \addpenalty{-\@highpenalty}%
      \vskip 1.0em \@plus\p@
      \setlength\@tempdima{1.5em}%
      \begingroup
         \parindent \z@ \if@rl\leftskip\else\rightskip\fi \@pnumwidth
         \parfillskip -\@pnumwidth
         \leavevmode \bfseries
         \advance\if@rl\rightskip\else\leftskip\fi\@tempdima
         \hskip -\if@rl\rightskip\else\leftskip\fi
         #1\nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss#2}\par
         \penalty\@highpenalty
```

```
1.631       \endgroup
1.632       \fi}}}}}
```

\l@section
\l@subsection
\l@subsubsection
\l@paragraph
\l@subparagraph

The toc entry for section did not work in article style. Also it does not print dots, which is funny when most of your work is divided into sections.

It was revised to use \@dottedtocline as in report.sty (by Yaniv Bargury) and was updated later for all kinds of sections (by Boris Lavva).

```
1.633 \@ifclassloaded{article}{%
1.634 \renewcommand*\l@section{\@dottedtocline{1}{1.5em}{2.3em}}
1.635 \renewcommand*\l@subsection{\@dottedtocline{2}{3.8em}{3.2em}}
1.636 \renewcommand*\l@subsubsection{\@dottedtocline{3}{7.0em}{4.1em}}
1.637 \renewcommand*\l@paragraph{\@dottedtocline{4}{10em}{5em}}
1.638 \renewcommand*\l@subparagraph{\@dottedtocline{5}{12em}{6em}}}{}
```

1.4.6 Two-column mode

This is the support of twocolumn option for the standard LaTeX 2_ε classes. The following code was originally borrowed from the ArabTeX package, file latexext.sty, copyright by Klaus Lagally, Institut fuer Informatik, Universitaet Stuttgart. It was updated for this package by Boris Lavva.

\@outputdblcol
\set@outputdblcol
rl@outputdblcol

First column is \@leftcolumn will be shown at the right side, Second column is \@outputbox will be shown at the left side.

\set@outputdblcol IS CURRENTLY DISABLED. TODO: REMOVE IT [tzafrir]

```
1.639 \let\@@outputdblcol\@outputdblcol
1.640 %\def\set@outputdblcol{%
1.641 %  \if@rl\renewcommand{\@outputdblcol}{\rl@outputdblcol}%
1.642 %  \else\renewcommand{\@outputdblcol}{\@@outputdblcol}\fi}
1.643 \renewcommand{\@outputdblcol}{%
1.644   \if@rlmain%
1.645     \rl@outputdblcol%
1.646   \else%
1.647     \@@outputdblcol%
1.648   \fi%
1.649 }
1.650 \newcommand{\rl@outputdblcol}{%
1.651   \if@firstcolumn
1.652     \global \@firstcolumnfalse
1.653     \global \setbox\@leftcolumn \box\@outputbox
1.654   \else
1.655     \global \@firstcolumntrue
1.656     \setbox\@outputbox \vbox {\hb@xt@\textwidth {%
1.657                               \hskip\columnwidth%
1.658                               \hfil\vrule\@width\columnseprule\hfil
1.659                               \hb@xt@\columnwidth {%
1.660                                 \box\@leftcolumn \hss}%
1.661                               \hb@xt@\columnwidth {%
1.662                                 \hskip-\textwidth%
1.663                                 \box\@outputbox \hss}%
1.664                               \hskip\columnsep%
1.665                               \hskip\columnwidth}}%
1.666     \@combinedblfloats
```

```
    \@outputpage
    \begingroup
      \@dblfloatplacement
      \@startdblcolumn
      \@whilesw\if@fcolmade \fi
        {\@outputpage
         \@startdblcolumn}%
    \endgroup
\fi}
```

1.4.7 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.

```
\let\@@footnoterule=\footnoterule
\def\footnoterule{\if@rl\hb@xt@\hsize{\hss\vbox{\@@footnoterule}}%
                  \else\@@footnoterule\fi}
```

1.4.8 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 ps@myheadings headeven headodd First, we will support the standard letter class:

```
\@ifclassloaded{letter}{%
  \def\headodd{\protect\if@rl\beginR\fi\headtoname{}
               \ignorespaces\toname
               \hfil \@date
               \hfil \pagename{} \thepage\protect\if@rl\endR\fi}
  \if@twoside
     \def\ps@headings{%
         \let\@oddfoot\@empty\let\@evenfoot\@empty
         \def\@oddhead{\select@language{\headlanguage}\headodd}
         \let\@evenhead\@oddhead}
  \else
     \def\ps@headings{%
         \let\@oddfoot\@empty
         \def\@oddhead{\select@language{\headlanguage}\headodd}}
  \fi
  \def\headfirst{\protect\if@rl\beginR\fi\fromlocation \hfill %
                 \telephonenum\protect\if@rl\endR\fi}
  \def\ps@firstpage{%
     \let\@oddhead\@empty
     \def\@oddfoot{\raisebox{-45\p@}[\z@]{%
        \hb@xt@\textwidth{\hspace*{100\p@}%
          \ifcase \@ptsize\relax
             \normalsize
```

```
          \or
             \small
          \or
             \footnotesize
          \fi
        \select@language{\headlanguage}\headfirst}}\hss}}
%
  \renewcommand{\opening}[1]{%
     \let\headlanguage=\languagename%
     \ifx\@empty\fromaddress%
        \thispagestyle{firstpage}%
        {\raggedleft\@date\par}%
     \else  % home address
        \thispagestyle{empty}%
        {\raggedleft
        \if@rl\begin{tabular}{@{\beginR\csname%
          to\@rllanguagename\endcsname}r@{\endR}}\ignorespaces
           \fromaddress \\*[2\parskip]%
           \@date \end{tabular}\par%
        \else\begin{tabular}{l}\ignorespaces
           \fromaddress \\*[2\parskip]%
           \@date \end{tabular}\par%
        \fi}%
     \fi
     \vspace{2\parskip}%
     {\raggedright \toname \\ \toaddress \par}%
     \vspace{2\parskip}%
     #1\par\nobreak}
}
```

Then, the article, report and book document classes are supported. Note, that in one-sided mode `\markright` was changed to `\markboth`.

```
{% article, report, book
  \def\headeven{\protect\if@rl\beginR\thepage\hfil\rightmark\endR
                \protect\else\thepage\hfil{\slshape\leftmark}
                \protect\fi}
  \def\headodd{\protect\if@rl\beginR\leftmark\hfil\thepage\endR
               \protect\else{\slshape\rightmark}\hfil\thepage
               \protect\fi}
  \@ifclassloaded{article}{% article
    \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\sectionmark##1{%
           \markboth {\MakeUppercase{%
               \ifnum \c@secnumdepth >\z@
                  \thesection\quad
               \fi
               ##1}}{}}%
         \def\subsectionmark##1{%
           \markright{%
```

```
            \ifnum \c@secnumdepth >\@ne
               \thesubsection\quad
            \fi
        ##1}}}
   \else          % one-sided
      \def\ps@headings{%
        \let\@oddfoot\@empty
        \def\@oddhead{\headodd}%
        \let\@mkboth\markboth
        \def\sectionmark##1{%
          \markboth{\MakeUppercase{%
              \ifnum \c@secnumdepth >\m@ne
                 \thesection\quad
              \fi
              ##1}}{\MakeUppercase{%
              \ifnum \c@secnumdepth >\m@ne
                 \thesection\quad
              \fi
              ##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}}}}
    \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\chaptermark\@gobble
      \let\sectionmark\@gobble
  }}}
```

1.4.9 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 TEXto 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 TEXto something like:
`[postscript:make color cyan]some LTR text[postscript:make color black]`
However, with the bidirectionality support we get:
`\textcolor{cyan}{\hebalef\hebbet}`
Translated to:
`[postscript:make color black]{bet}{alef}[postscript:make color cyan]`
While we want:
`[postscript:make color cyan]{bet}{alef}[postscript:make color black]`
The following code will probably work at least with code that stays in the same line:

`@textcolor`

```
\AtBeginDocument{%
  %I assume that \@textcolor is only defined by the package color
  \ifx\@textcolor\@undefined\else%
    % If that macro was defined before the beginning of the document,
    % that is: the package was loaded: redefine it with bidi support
    \def\@textcolor#1#2#3{%
      \if@rl%
        \beginL\protect\leavevmode{\color#1{#2}\beginR#3\endR}\endL%
      \else%
        \protect\leavevmode{\color#1{#2}#3}%
      \fi%
    }%
  \fi%
}
% \end{macrocode}
```

```
% \end{macro}
% \begin{macro}{\thetrueSlideCounter}
%    This macro probably needs to be overriden for when using |prosper|,
%    (waiting for feedback. Tzafrir)
%    \begin{macrocode}
\@ifclassloaded{prosper}{%
  \def\thetrueSlideCounter{\arabicnorl{trueSlideCounter}}
}{}
```

1.4.10 Miscellaneous internal LaTeX macros

\raggedright \raggedleft

\raggedright was changed from latex.ltx file to support Right-to-Left mode, because of the bug in its implementation.

```
\def\raggedright{%
  \let\\\@centercr
  \leftskip\z@skip\rightskip\@flushglue
  \parindent\z@\parfillskip\z@skip}
```

Swap meanings of \raggedright and \raggedleft in Right-to-Left mode.

```
\let\@@raggedleft=\raggedleft
\let\@@raggedright=\raggedright
\renewcommand\raggedleft{\if@rl\@@raggedright%
                           \else\@@raggedleft\fi}
\renewcommand\raggedright{\if@rl\@@raggedleft%
                            \else\@@raggedright\fi}
```

\author

\author is inserted with tabular environment, and will be used in restricted horizontal mode. Therefore we have to add explicit direction change command when in Right-to-Left mode.

```
\let\@@author=\author
\renewcommand{\author}[1]{\@@author{\if@rl\beginR #1\endR\else #1\fi}}
```

\MakeUppercase \MakeLowercase

There are no uppercase and lowercase letters in most Right-to-Left languages, therefore we should redefine \MakeUppercase and \MakeLowercase LaTeX 2_ε commands.

```
\let\@@MakeUppercase=\MakeUppercase
\def\MakeUppercase#1{\if@rl#1\else\@@MakeUppercase{#1}\fi}
\let\@@MakeLowercase=\MakeLowercase
\def\MakeLowercase#1{\if@rl#1\else\@@MakeLowercase{#1}\fi}
```

\underline

We should explicitly use \L and \R commands in \underlined text.

```
\DeclareRobustCommand\underline[1]{%
  \relax
  \ifmmode\@@underline{#1}%
  \else
\if@rl $\@@underline{\hbox{\beginR#1\endR}}\m@th$\relax
\else
$\@@underline{\hbox{#1}}\m@th$\relax\fi\fi}
```

\undertext was added for LaTeX2.09 compatibility mode.

```
\if@compatibility
   \let\undertext=\underline
\fi
```

\@xnthm
\@opargbegintheorem
The following has been inserted to correct the appearance of the number in \newtheorem to reorder theorem number components. A similar correction in the definition of \@opargbegintheorem was added too.

```
\def\@xnthm#1#2[#3]{%
  \expandafter\@ifdefinable\csname #1\endcsname
  {\@definecounter{#1}\@addtoreset{#1}{#3}%
    \expandafter\xdef\csname the#1\endcsname{\noexpand\@number
      {\expandafter\noexpand\csname the#3\endcsname \@thmcountersep
        \@thmcounter{#1}}}%
    \global\@namedef{#1}{\@thm{#1}{#2}}%
    \global\@namedef{end#1}{\@endtheorem}}}
%
\def\@opargbegintheorem#1#2#3{%
  \trivlist
      \item[\hskip \labelsep{\bfseries #1\ #2\
          \@brackets({#3})}]\itshape}
```

\@chapter
\@schapter
The following was added for pretty printing of the chapter numbers, for supporting Right-to-Left tables (cot, fol, and tol), to save \headlanguage for use in running headers, and to start two-column mode depending on chapter's main language.

```
\@ifclassloaded{article}{}{%
  % For pretty priniting
  \def\@@chapapp{Chapter}
  \def\@@thechapter{\@@arabic\c@chapter}
  \def\@chapter[#1]#2{%
    \let\headlanguage=\languagename%
    %\set@outputdblcol%
    \ifnum \c@secnumdepth >\m@ne
       \refstepcounter{chapter}%
       \typeout{\@@chapapp\space\@@thechapter.}%
       \addcontentsline{toc}{chapter}%
       {\protect\numberline{\thechapter}#1}
       \addcontentsline{cot}{chapter}%
       {\protect\numberline{\thechapter}#1}
    \else
       \addcontentsline{toc}{chapter}{#1}%
       \addcontentsline{cot}{chapter}{#1}%
    \fi
    \chaptermark{#1}
    \addtocontents{lof}{\protect\addvspace{10\p@}}%
    \addtocontents{fol}{\protect\addvspace{10\p@}}%
    \addtocontents{lot}{\protect\addvspace{10\p@}}%
    \addtocontents{tol}{\protect\addvspace{10\p@}}%
    \if@twocolumn
       \@topnewpage[\@makechapterhead{#2}]%
    \else
       \@makechapterhead{#2}%
       \@afterheading
    \fi}
  %
  \def\@schapter#1{%
    \let\headlanguage=\languagename%
    %\set@outputdblcol%
    \if@twocolumn
```

```
      \@topnewpage[\@makeschapterhead{#1}]%
    \else
      \@makeschapterhead{#1}%
      \@afterheading
    \fi}}
```

`\appendix` Changed mainly for pretty printing of appendix numbers, and to start two-column mode with the right language (if needed).

```
\@ifclassloaded{letter}{}{% other
\@ifclassloaded{slides}{}{% other
  \@ifclassloaded{article}{% article
    \renewcommand\appendix{\par
      \setcounter{section}{0}%
      \setcounter{subsection}{0}%
      \renewcommand\thesection{\@Alph\c@section}}
  }{% report and book
    \renewcommand\appendix{\par
      %\set@outputdblcol%
      \setcounter{chapter}{0}%
      \setcounter{section}{0}%
      \renewcommand\@chapapp{\appendixname}%
      % For pretty priniting
      \def\@@chapapp{Appendix}%
      \def\@@thechapter{\@@Alph\c@chapter}
      \renewcommand\thechapter{\@Alph\c@chapter}}}}}
```

1.4.11 Bibliography and citations

`\@cite` `\@biblabel` `\@lbibitem` Citations are produced by the macro `\@cite{`*LABEL*`}{`*NOTE*`}`. Both the citation label and the note is typeset in the current direction. We have to use `\@brackets` 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.

```
\def\@cite#1#2{\@brackets[{#1\if@tempswa , #2\fi}]}
\def\@biblabel#1{\@brackets[{#1}]}
\def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
      {\let\protect\noexpand
       \immediate
       \if@rl\write\@auxout{\string\bibcite{#2}{\R{#1}}}%
       \else\write\@auxout{\string\bibcite{#2}{\L{#1}}}\fi%
      }\fi\ignorespaces}
```

`thebibliography` (*env.*) Use `\rightmargin` instead of `\leftmargin` when in RL mode.

```
\@ifclassloaded{letter}{}{% other
\@ifclassloaded{slides}{}{% other
\@ifclassloaded{article}{%
  \renewenvironment{thebibliography}[1]
  {\section*{\refname\@mkboth%
      {\MakeUppercase\refname}%
      {\MakeUppercase\refname}}%
    \list{\@biblabel{\@arabic\c@enumiv}}%
    {\settowidth\labelwidth{\@biblabel{#1}}%
      \if@rl\leftmargin\else\rightmargin\fi\labelwidth
```

```
      \advance\if@rl\leftmargin\else\rightmargin\fi\labelsep
      \@openbib@code
      \usecounter{enumiv}%
      \let\p@enumiv\@empty
      \renewcommand\theenumiv{\@arabic\c@enumiv}}%
    \sloppy
    \clubpenalty4000
    \@clubpenalty \clubpenalty
    \widowpenalty4000%
    \sfcode`\.\@m}
  {\def\@noitemerr
    {\@latex@warning{Empty `thebibliography' environment}}%
     \endlist}}%
{\renewenvironment{thebibliography}[1]{%
    \chapter*{\bibname\@mkboth%
      {\MakeUppercase\bibname}%
      {\MakeUppercase\bibname}}%
    \list{\@biblabel{\@arabic\c@enumiv}}%
    {\settowidth\labelwidth{\@biblabel{#1}}%
      \if@rl\leftmargin\else\rightmargin\fi\labelwidth
      \advance\if@rl\leftmargin\else\rightmargin\fi\labelsep
      \@openbib@code
      \usecounter{enumiv}%
      \let\p@enumiv\@empty
      \renewcommand\theenumiv{\@arabic\c@enumiv}}%
    \sloppy
    \clubpenalty4000
    \@clubpenalty \clubpenalty
    \widowpenalty4000%
    \sfcode`\.\@m}
  {\def\@noitemerr
    {\@latex@warning{Empty `thebibliography' environment}}%
     \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.

```
\def\@verbatim{%
  \let\do\@makeother \dospecials%
  \obeylines \verbatim@font \@noligs}
```

`\@makecaption` Captions are set always centered. This allows us to use bilingual captions, for example: `\caption{\R{RLtext} \\ \L{LRtext}}`, which will be formatted as:

Right to left caption here (RLtext)
Left to right caption here (LRtext)

See also `\bcaption` command below.

```
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip%
  \begin{center}%
    #1: #2%
  \end{center} \par%
  \vskip\belowcaptionskip}
```

1.4.12 Additional bidirectional commands

- Section headings are typeset with the default global direction.
- Text in section headings in the reverse language *do not* have to be protected for the reflection command, as in: `\protect\L{`*Latin Text*`}`, because `\L` and `\R` are robust now.
- Table of contents, list of figures and list of tables should be typeset with the `\tableofcontents`, `\listoffigures` and `\listoftables` commands respectively.
- The above tables will be typeset in the main direction (and language) in effect where the above commands are placed.
- Only 2 tables of each kind are supported: one for Right-to-Left and another for Left-to-Right directions.

How to include line to both tables? One has to use bidirectional sectioning commands as following:

1. Use the `\b`*xxx* version of the sectioning commands in the text instead of the `\`*xxx* version (*xxx* is one of: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `caption`).
2. Syntax of the `\b`*xxx* command is `\b`*xxx*`{`*RL text*`}{`*LR text*`}`. Both arguments are typeset in proper direction by default (no need to change direction for the text inside).
3. The section header inside the document will be typeset in the global direction in effect at the time. i.e. The `{`*RL text*`}` will be typeset if Right-to-Left mode is in effect and `{`*LR text*`}` otherwise.

`\bpart`

```
1.1002 \newcommand{\bpart}[2]{\part{\protect\if@rl%
1.1003     #1 \protect\else #2 \protect\fi}}
```

`\bchapter`

```
1.1004 \newcommand{\bchapter}[2]{\chapter{\protect\if@rl%
1.1005     #1 \protect\else #2 \protect\fi}}
```

`\bsection`

```
1.1006 \newcommand{\bsection}[2]{\section{\protect\if@rl%
1.1007     #1 \protect\else #2 \protect\fi}}
```

`\bsubsection`

```
1.1008 \newcommand{\bsubsection}[2]{\subsection{\protect\if@rl%
1.1009     #1 \protect\else #2 \protect\fi}}
```

`\bsubsubsection`

```
1.1010 \newcommand{\bsubsubsection}[2]{\subsubsection{\protect\if@rl%
1.1011     #1 \protect\else #2 \protect\fi}}
```

\bcaption

```
\newcommand{\bcaption}[2]{%
  \caption[\protect\if@rl \R{#1}\protect\else \L{#2}\protect\fi]{%
    \if@rl\R{#1}\protect\\ \L{#2}
    \else\L{#2}\protect\\ \R{#1}\fi}}
```

The following definition is a modified version of `\bchapter`, meant as a bilingual twin for `\chapter*` and `\section*` (added by Irina Abramovici).

\bchapternn

```
\newcommand{\bchapternn}[2]{\chapter*{\protect\if@rl%
    #1 \protect\else #2 \protect\fi}}
```

\bsectionnn

```
\newcommand{\bsectionnn}[2]{\section*{\protect\if@rl%
    #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.

```
\AtEndOfPackage{\rlAtEndOfPackage}
%
\def\rlAtEndOfPackage{%
  \global\let\headlanguage=\languagename%\set@outputdblcol%
}
```

⟨/rightleft⟩

1.5 Hebrew calendar

The original version of the package hebcal.sty[2] for TEX and LATEX2.09, entitled "TEX & LATEX macros for computing Hebrew date from Gregorian one" was created by Michail Rozman, misha@iop.tartu.ew.su[3]

Released:	Tammuz 12, 5751–June 24, 1991	
Corrected:	Shebat 10, 5752–January 15, 1992	by Rama Porrat
Corrected:	Adar II 5, 5752–March 10, 1992	by Misha
Corrected:	Tebeth, 5756–January 1996	Dan Haran (haran@math.tau.ac.il)

The package was adjusted for babel and LATEX 2ε 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:

\Hebrewtoday Computes today's Hebrew date and prints it. If we are presently in Hebrew mode, the date will be printed in Hebrew, otherwise — in English (like Shebat 10, 5752).

\Hebrewdate Computes the Hebrew date from the given Gregorian date and prints it. If we

[2]The following description of hebcal package is based on the comments included with original source by the author, Michail Rozman.

[3]Please direct any comments, bug reports, questions, etc. about the package to this address.

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}
```

`full` The package option `full` sets the flag `\@full@hebrew@year`, which causes years from the current millenium to be printed with the thousands digit (he-tav-shin-samekh-gimel). Without this option, thousands are not printed for the current millenium. NOTE: should this be a command option rather than a package option? –RA.

1.5.1 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^{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.

1.5.2 Registers, Commands, Formatting Macros

The command `\Hebrewtoday` produces today's date for Hebrew calendar. It is similar to the standard LaTeX 2_ε command `\today`. In addition three numerical registers `\Hebrewday`, `\Hebrewmonth` and `\Hebrewyear` are set. For setting this registers without producing of date string command `\Hebrewsetreg` can be used.

The command `\Hebrewdate{`*Gday*`}{`*Gmonth*`}{`*Gyear*`}` produces Hebrew calendar date corresponding to Gregorian date `Gday.Gmonth.Gyear`. Three numerical registers `\Hebrewday`, `\Hebrewmonth` and `\Hebrewyear` are set.

For converting arbitrary Gregorian date `Gday.Gmonth.Gyear` to Hebrew date `Hday.Hmonth.Hyear` without producing date string the command:

`\HebrewFromGregorian{`*Gday*`}{`*Gmonth*`}{ `*Gyear*`}{`*Hday*`}{`*Hmonth*`}{`*Hyear*`}`

can be used.

1.1026 ⟨∗calendar⟩
1.1027 `\newif\if@full@hebrew@year`
1.1028 `\@full@hebrew@yearfalse`
1.1029 `\DeclareOption{full}{\@full@hebrew@yeartrue}`
1.1030 `\ProcessOptions`
1.1031 `\newcount\Hebrewday  \newcount\Hebrewmonth  \newcount\Hebrewyear`

`\Hebrewdate` Hebrew calendar date corresponding to Gregorian date `Gday.Gmonth.Gyear`. If Hebrew (right-to-left) fonts & macros are not loaded, we have to use English format.

```
\def\Hebrewdate#1#2#3{%
    \HebrewFromGregorian{#1}{#2}{#3}
                         {\Hebrewday}{\Hebrewmonth}{\Hebrewyear}%
    \ifundefined{if@rl}%
       \FormatForEnglish{\Hebrewday}{\Hebrewmonth}{\Hebrewyear}%
    \else%
       \FormatDate{\Hebrewday}{\Hebrewmonth}{\Hebrewyear}%
    \fi}
```

`\Hebrewtoday` Today's date in Hebrew calendar.

```
\def\Hebrewtoday{\Hebrewdate{\day}{\month}{\year}}
\let\hebrewtoday=\Hebrewtoday
```

`\Hebrewsetreg` Set registers: today's date in hebrew calendar.

```
\def\Hebrewsetreg{%
    \HebrewFromGregorian{\day}{\month}{\year}
                         {\Hebrewday}{\Hebrewmonth}{\Hebrewyear}}
```

`\FormatDate` Prints a Hebrew calendar date `Hebrewday.Hebrewmonth.Hebrewyear`.

```
\def\FormatDate#1#2#3{%
        \if@rl%
            \FormatForHebrew{#1}{#2}{#3}%
        \else%
            \FormatForEnglish{#1}{#2}{#3}
        \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.

```
\def\HebrewYearName#1{{%
   \@tempcnta=#1\divide\@tempcnta by 1000\multiply\@tempcnta by 1000
   \ifnum#1=\@tempcnta\relax % divisible by 1000: disambiguate
     \Hebrewnumeralfinal{#1}\ )\heblamed\hebpe"\hebgimel(%
   \else % not divisible by 1000
     \ifnum#1<1000\relax     % first millennium: disambiguate
       \Hebrewnumeralfinal{#1}\ )\heblamed\hebpe"\hebgimel(%
     \else
       \ifnum#1<5000
         \Hebrewnumeralfinal{#1}%
       \else
         \ifnum#1<6000 % current millenium, print without thousands
           \@tempcnta=#1\relax
           \if@full@hebrew@year\else\advance\@tempcnta by -5000\fi
           \Hebrewnumeralfinal{\@tempcnta}%
```

```
          \else % #1>6000
            \Hebrewnumeralfinal{#1}%
          \fi
        \fi
      \fi
    \fi}}
```

`\HebrewMonthName` The macro `\HebrewMonthName{`*month*`}{`*year*`}` returns the name of month in the 'year'.

```
\def\HebrewMonthName#1#2{%
    \ifnum #1 = 7 %
    \CheckLeapHebrewYear{#2}%
        \if@HebrewLeap \hebalef\hebdalet\hebresh\ \hebbet'%
           \else \hebalef\hebdalet\hebresh%
        \fi%
    \else%
        \ifcase#1%
           % nothing for 0
           \or\hebtav\hebshin\hebresh\hebyod%
           \or\hebhet\hebshin\hebvav\hebfinalnun%
           \or\hebkaf\hebsamekh\heblamed\hebvav%
           \or\hebtet\hebbet\hebtav%
           \or\hebshin\hebbet\hebtet%
           \or\hebalef\hebdalet\hebresh\ \hebalef'%
           \or\hebalef\hebdalet\hebresh\ \hebbet'%
           \or\hebnun\hebyod\hebsamekh\hebfinalnun%
           \or\hebalef\hebyod\hebyod\hebresh%
           \or\hebsamekh\hebyod\hebvav\hebfinalnun%
           \or\hebtav\hebmem\hebvav\hebzayin%
           \or\hebalef\hebbet%
           \or\hebalef\heblamed\hebvav\heblamed%
        \fi%
    \fi}
```

`\HebrewDayName` Name of day in Hebrew letters (gimatria).

```
\def\HebrewDayName#1{\Hebrewnumeral{#1}}
```

`\FormatForHebrew` The macro `\FormatForHebrew{`*hday*`}{`*hmonth* `}{`*hyear*`}` returns the formatted Hebrew date in Hebrew language.

```
\def\FormatForHebrew#1#2#3{%
  \HebrewDayName{#1}~\hebbet\HebrewMonthName{#2}{#3},~%
  \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.

```
\def\HebrewMonthNameInEnglish#1#2{%
    \ifnum #1 = 7%
    \CheckLeapHebrewYear{#2}%
        \if@HebrewLeap Adar II\else Adar\fi%
    \else%
        \ifcase #1%
```

```
            % nothing for 0
            \or Tishrei%
            \or Heshvan%
            \or Kislev%
            \or Tebeth%
            \or Shebat%
            \or Adar I%
            \or Adar II%
            \or Nisan%
            \or Iyar%
            \or Sivan%
            \or Tammuz%
            \or Av%
            \or Elul%
        \fi
    \fi}
```

`\FormatForEnglish` The macro `\FormatForEnglish{`*hday*`}{`*hmonth* `}{`*hyear*`}` is similar to `\FormatForHebrew` macro described above and returns the formatted Hebrew date in English.

```
\def\FormatForEnglish#1#2#3{%
    \HebrewMonthNameInEnglish{#2}{#3} \number#1,\ \number#3}
```

1.5.3 Auxiliary Macros

```
\newcount\@common
```

`\Remainder` `\Remainder{`a`}{`b`}{`c`}` calculates $c = a\%b == a - b \times \frac{a}{b}$

```
\def\Remainder#1#2#3{%
    #3 = #1%                    %  c = a
    \divide #3 by #2%           %  c = a/b
    \multiply #3 by -#2%        %  c = -b(a/b)
    \advance #3 by #1}%         %  c = a - b(a/b)

\newif\if@Divisible
```

`\CheckIfDivisible` `\CheckIfDivisible{`a`}{`b`}` sets `\@Divisibletrue` if $a\%b == 0$

```
\def\CheckIfDivisible#1#2{%
    {%
      \countdef\tmp = 0% \tmp == \count0 - temporary variable
      \Remainder{#1}{#2}{\tmp}%
      \ifnum \tmp = 0%
          \global\@Divisibletrue%
      \else%
          \global\@Divisiblefalse%
      \fi}}
```

`\ifundefined` From the TEXbook, ex. 7.7:

`\ifundefined{`*command*`}<true text>\else<false text>\fi`

```
\def\ifundefined#1{\expandafter\ifx\csname#1\endcsname\relax}
```

1.5.4 Gregorian Part

```
\newif\if@GregorianLeap
```

`\IfGregorianLeap` Conditional which is true if Gregorian 'year' is a leap year: $((year\%4 == 0) \wedge (year\%100 \neq 0)) \vee (year\%400 == 0)$

```
\def\IfGregorianLeap#1{%
    \CheckIfDivisible{#1}{4}%
    \if@Divisible%
        \CheckIfDivisible{#1}{100}%
        \if@Divisible%
            \CheckIfDivisible{#1}{400}%
            \if@Divisible%
                \@GregorianLeaptrue%
            \else%
                \@GregorianLeapfalse%
            \fi%
        \else%
            \@GregorianLeaptrue%
        \fi%
    \else%
        \@GregorianLeapfalse%
    \fi%
    \if@GregorianLeap}
```

`\GregorianDaysInPriorMonths` The macro `\GregorianDaysInPriorMonths{`*month*`}{`*year*`}{`*days*`}` calculates the number of days in months prior to 'month' in the 'year'.

```
\def\GregorianDaysInPriorMonths#1#2#3{%
    {%
        #3 = \ifcase #1%
                0 \or%              % no month number 0
                0 \or%
               31 \or%
               59 \or%
               90 \or%
              120 \or%
              151 \or%
              181 \or%
              212 \or%
              243 \or%
              273 \or%
              304 \or%
              334%
        \fi%
        \IfGregorianLeap{#2}%
            \ifnum #1 > 2%        % if month after February
                \advance #3 by 1% % add leap day
            \fi%
        \fi%
        \global\@common = #3}%
    #3 = \@common}
```

`\GregorianDaysInPriorYears` The macro `\GregorianDaysInPriorYears{`*year*`}{`*days*`}` calculates the number of days in years prior to the 'year'.

```
\def\GregorianDaysInPriorYears#1#2{%
     {%
         \countdef\tmpc = 4%      % \tmpc==\count4
         \countdef\tmpb = 2%      % \tmpb==\count2
         \tmpb = #1%              %
         \advance \tmpb by -1%    %
         \tmpc = \tmpb%           % \tmpc = \tmpb = year-1
         \multiply \tmpc by 365%  % Days in prior years =
         #2 = \tmpc%              % = 365*(year-1) ...
         \tmpc = \tmpb%           %
         \divide \tmpc by 4%      % \tmpc = (year-1)/4
         \advance #2 by \tmpc%    % ... plus Julian leap days ...
         \tmpc = \tmpb%           %
         \divide \tmpc by 100%    % \tmpc = (year-1)/100
         \advance #2 by -\tmpc%   % ... minus century years ...
         \tmpc = \tmpb%           %
         \divide \tmpc by 400%    % \tmpc = (year-1)/400
         \advance #2 by \tmpc%    % ... plus 4-century years.
         \global\@common = #2}%
    #2 = \@common}
```

`\AbsoluteFromGregorian` The macro `\AbsoluteFromGregorian{`*day*`}{`*month*`}{`*year*`}{`*absdate*`}` calculates the absolute date (days since 01.01.0001) from Gregorian date `day.month.year`.

```
\def\AbsoluteFromGregorian#1#2#3#4{%
    {%
        \countdef\tmpd = 0%       % \tmpd==\count0
        #4 = #1%                  % days so far this month
        \GregorianDaysInPriorMonths{#2}{#3}{\tmpd}%
        \advance #4 by \tmpd%     % add days in prior months
        \GregorianDaysInPriorYears{#3}{\tmpd}%
        \advance #4 by \tmpd%     % add days in prior years
        \global\@common = #4}%
    #4 = \@common}
```

1.5.5 Hebrew Part

```
\newif\if@HebrewLeap
```

`\CheckLeapHebrewYear` Set `\@HebrewLeaptrue` if Hebrew 'year' is a leap year, i.e. if $(1+7\times year)\%19 < 7$ then *true* else *false*

```
\def\CheckLeapHebrewYear#1{%
    {%
        \countdef\tmpa = 0%       % \tmpa==\count0
        \countdef\tmpb = 1%       % \tmpb==\count1
%
        \tmpa = #1%
        \multiply \tmpa by 7%
        \advance \tmpa by 1%
        \Remainder{\tmpa}{19}{\tmpb}%
        \ifnum \tmpb < 7%         % \tmpb = (7*year+1)%19
            \global\@HebrewLeaptrue%
        \else%
            \global\@HebrewLeapfalse%
        \fi}}
```

\HebrewElapsedMonths The macro \HebrewElapsedMonths{*year*}{*months*} determines the number of months elapsed from the Sunday prior to the start of the Hebrew calendar to the mean conjunction of Tishri of Hebrew 'year'.

```
\def\HebrewElapsedMonths#1#2{%
    {%
        \countdef\tmpa = 0%       % \tmpa==\count0
        \countdef\tmpb = 1%       % \tmpb==\count1
        \countdef\tmpc = 2%       % \tmpc==\count2
%
        \tmpa = #1%               %
        \advance \tmpa by -1%     %
        #2 = \tmpa%               % #2 = \tmpa = year-1
        \divide #2 by 19%         % Number of complete Meton cycles
        \multiply #2 by 235%      % #2 = 235*((year-1)/19)
%
        \Remainder{\tmpa}{19}{\tmpb}% \tmpa = years%19-years this cycle
        \tmpc = \tmpb%            %
        \multiply \tmpb by 12%    %
        \advance #2 by \tmpb%     % add regular months this cycle
%
        \multiply \tmpc by 7%     %
        \advance \tmpc by 1%      %
        \divide \tmpc by 19%      % \tmpc = (1+7*((year-1)%19))/19 -
%                                 %  number of leap months this cycle
        \advance #2 by \tmpc%     %  add leap months
%
        \global\@common = #2}%
    #2 = \@common}
```

\HebrewElapsedDays The macro \HebrewElapsedDays{*year*}{*days*} determines the number of days elapsed from the Sunday prior to the start of the Hebrew calendar to the mean conjunction of Tishri of Hebrew 'year'.

```
\def\HebrewElapsedDays#1#2{%
    {%
        \countdef\tmpa = 0%       % \tmpa==\count0
        \countdef\tmpb = 1%       % \tmpb==\count1
        \countdef\tmpc = 2%       % \tmpc==\count2
%
        \HebrewElapsedMonths{#1}{#2}%
        \tmpa = #2%               %
        \multiply \tmpa by 13753% %
        \advance \tmpa by 5604%   % \tmpa=MonthsElapsed*13758 + 5604
        \Remainder{\tmpa}{25920}{\tmpc}% \tmpc == ConjunctionParts
        \divide \tmpa by 25920%
%
        \multiply #2 by 29%
        \advance #2 by 1%
        \advance #2 by \tmpa%     %  #2 = 1 + MonthsElapsed*29 +
%                                 %          PartsElapsed/25920
        \Remainder{#2}{7}{\tmpa}% %  \tmpa == DayOfWeek
        \ifnum \tmpc < 19440%
            \ifnum \tmpc < 9924%
            \else%                % New moon at 9 h. 204 p. or later
                \ifnum \tmpa = 2% % on Tuesday ...
```

```
                    \CheckLeapHebrewYear{#1}% of a common year
                    \if@HebrewLeap%
                    \else%
                        \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%
                    \CheckLeapHebrewYear{#1}% at the end of leap year
                    \if@HebrewLeap%
                        \advance #2 by 1%
                    \fi%
                \fi%
            \fi%
        \else%
            \advance #2 by 1%       %  new moon at or after midday
        \fi%
%
        \Remainder{#2}{7}{\tmpa}%  %  \tmpa == DayOfWeek
        \ifnum \tmpa = 0%          %  if Sunday ...
            \advance #2 by 1%
        \else%                     %
            \ifnum \tmpa = 3%      %  Wednesday ...
                \advance #2 by 1%
            \else%
                \ifnum \tmpa = 5%  %  or Friday
                     \advance #2 by 1%
                \fi%
            \fi%
        \fi%
        \global\@common = #2}%
    #2 = \@common}
```

`\DaysInHebrewYear` The macro `\DaysInHebrewYear{`*year*`}{`*days*`}` calculates the number of days in Hebrew 'year'.

```
\def\DaysInHebrewYear#1#2{%
    {%
        \countdef\tmpe = 12%   % \tmpe==\count12
%
        \HebrewElapsedDays{#1}{\tmpe}%
        \advance #1 by 1%
        \HebrewElapsedDays{#1}{#2}%
        \advance #2 by -\tmpe%
        \global\@common = #2}%
    #2 = \@common}
```

`\HebrewDaysInPriorMonths` The macro `\HebrewDaysInPriorMonths{`*month*`}{`*year*`}{`*days*`}` calculates the number of days in months prior to 'month' in the 'year'.

```
\def\HebrewDaysInPriorMonths#1#2#3{%
    {%
        \countdef\tmpf= 14%    % \tmpf==\count14
```

```
%
        #3 = \ifcase #1%       % Days in prior month of regular year
               0 \or%          % no month number 0
               0 \or%          % Tishri
              30 \or%          % Heshvan
              59 \or%          % Kislev
              89 \or%          % Tebeth
             118 \or%          % Shebat
             148 \or%          % Adar I
             148 \or%          % Adar II
             177 \or%          % Nisan
             207 \or%          % Iyar
             236 \or%          % Sivan
             266 \or%          % Tammuz
             295 \or%          % Av
             325 \or%          % Elul
             400%              % Dummy
        \fi%
        \CheckLeapHebrewYear{#2}%
        \if@HebrewLeap%            % in leap year
            \ifnum #1 > 6%         % if month after Adar I
                \advance #3 by 30% % add  30 days
            \fi%
        \fi%
        \DaysInHebrewYear{#2}{\tmpf}%
        \ifnum #1 > 3%
            \ifnum \tmpf = 353%    %
                \advance #3 by -1% %
            \fi%                   %  Short Kislev
            \ifnum \tmpf = 383%    %
                \advance #3 by -1% %
            \fi%                   %
        \fi%
%
        \ifnum #1 > 2%
            \ifnum \tmpf = 355%    %
                \advance #3 by 1%  %
            \fi%                   %  Long Heshvan
            \ifnum \tmpf = 385%    %
                \advance #3 by 1%  %
            \fi%                   %
        \fi%
        \global\@common = #3}%
    #3 = \@common}
```

`\AbsoluteFromHebrew` The macro `\AbsoluteFromHebrew{`*day*`}{`*month*`}{`*year*`}{`*absdate*`}` calculates the absolute date of Hebrew date `day.month.year`.

```
\def\AbsoluteFromHebrew#1#2#3#4{%
    {%
        #4 = #1%
        \HebrewDaysInPriorMonths{#2}{#3}{#1}%
        \advance #4 by #1%         % Add days in prior months this year
        \HebrewElapsedDays{#3}{#1}%
        \advance #4 by #1%         % Add days in prior years
```

```
        \advance #4 by -1373429%   % Subtract days before Gregorian
        \global\@common = #4}%     %   01.01.0001
    #4 = \@common}
```

\HebrewFromGregorian The macro \HebrewFromGregorian{*Gday*}{*Gmonth*}{*Gyear*}{*Hday*}{*Hmonth*}-{*Hyear*} evaluates Hebrew date Hday, Hmonth, Hyear from Gregorian date Gday, Gmonth, Gyear.

```
\def\HebrewFromGregorian#1#2#3#4#5#6{%
    {%
        \countdef\tmpx= 17%        % \tmpx==\count17
        \countdef\tmpy= 18%        % \tmpy==\count18
        \countdef\tmpz= 19%        % \tmpz==\count19
%
        #6 = #3%                   %
        \global\advance #6 by 3761%  approximation from above
        \AbsoluteFromGregorian{#1}{#2}{#3}{#4}%
        \tmpz = 1  \tmpy = 1%
        \AbsoluteFromHebrew{\tmpz}{\tmpy}{#6}{\tmpx}%
        \ifnum \tmpx > #4%              %
            \global\advance #6 by -1% Hyear = Gyear + 3760
            \AbsoluteFromHebrew{\tmpz}{\tmpy}{#6}{\tmpx}%
        \fi%                           %
        \advance #4 by -\tmpx%     % Days in this year
        \advance #4 by 1%          %
        #5 = #4%                   %
        \divide #5 by 30%          % Approximation for month from below
        \loop%                     % Search for month
            \HebrewDaysInPriorMonths{#5}{#6}{\tmpx}%
            \ifnum \tmpx < #4%
                \advance #5 by 1%
                \tmpy = \tmpx%
        \repeat%
        \global\advance #5 by -1%
        \global\advance #4 by -\tmpy}}
```

⟨/calendar⟩

2 Hebrew in LaTeX 2.09 compatibility mode

\documentstyle command in the preamble of LaTeX document indicates that it is a LaTeX 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 LaTeX 2.09 should compile under compatibility mode, unless it uses low-level commands such as `\tenrm`.

2.1 The DOCSTRIP modules

The following modules are used in the implementation to direct DOCSTRIP in generating the external files:

newcode	produce `hebrew_newcode.sty`
pccode	produce `hebrew_p.sty`
oldcode	produce `hebrew_oldcode.sty`

2.2 Obsolete style files

For each of the Hebrew LaTeX 2.09 Hebrew styles, we produce a file which uses correct input encoding and calls babel with Hebrew and English language options. This means that any styles which say `\input hebrew_newcode.sty` or `\documentstyle[...hebrew_newcode...]{...}` should still work.

```
⟨*newcode | pccode | oldcode⟩
\NeedsTeXFormat{LaTeX2e}
⟨/newcode | pccode | oldcode⟩

⟨*newcode⟩
\@obsoletefile{hebrew.sty}{hebrew_newcode.sty}
\RequirePackage[8859-8]{inputenc}
⟨/newcode⟩
⟨*pccode⟩
\@obsoletefile{hebrew.sty}{hebrew_p.sty}
\RequirePackage[cp862]{inputenc}
⟨/pccode⟩
⟨*oldcode⟩
\@obsoletefile{hebrew.sty}{hebrew_oldcode.sty}
\RequirePackage[si960]{inputenc}
⟨/oldcode⟩

⟨*newcode | pccode | oldcode⟩
\RequirePackage[english,hebrew]{babel}
⟨/newcode | pccode | oldcode⟩
```