Greek Language Support for 
\XeLaTeX and \LuaLaTeX

Apostolos Syropoulos  
Xanthi, Greece  
asyropoulos@yahoo.com

2009/11/23  
Last Updated 2017/02/10

Abstract

The \texttt{xgreek} package provides rudimentary support for Greek language typesetting with \texttt{XeLaTeX} and \texttt{LuaLaTeX}. In particular, it provides support for modern Greek (either monotonic or polytonic) and ancient Greek.

1 Introduction

The \texttt{xgreek} package provides rudimentary support for Greek language typesetting with \texttt{XeLaTeX} and \texttt{ LuaLaTeX}. Users will be able to typeset documents in either modern Greek (monotonic or polytonic) or ancient Greek by selecting the appropriate package option. The default “language” is monotonic Greek.

Support for \texttt{LuaLaTeX} was provided by Javier Bezos.

2 The Source Code

According to the Unicode standard

\url{http://www.unicode.org/Public/UNIDATA/UnicodeData.txt}
the uppercase form of \texttt{GREEK SMALL LETTER EPSILON WITH TONOS} is \texttt{GREEK CAPITAL LETTER ETA WITH TONOS}. This is certainly wrong. The main reason is that accents are not part of the letter as for example is the case with \texttt{LATIN SMALL LETTER K WITH CARON}. Since, \texttt{XeLaTeX} blindly follows the Unicode standard, commands like \texttt{\MakeUppercase} produce wrong output. For this reason I first need to set up the correct \texttt{uccodes} and \texttt{lccodes}.

1 \texttt{(xgreek)}
2 \texttt{\message{Package `xgreek' version 3.0 by Apostolos Syropoulos}}
3 \texttt{\global lccode"0370"=0371 \global uccode"0370"=0370}
4 \texttt{\global lccode"0371"=0371 \global uccode"0371"=0370}
Next I define the various strings that correspond to the standard \LaTeX\ captions. I first define the strings for monotonic Greek.

\def\prefacename{Πρόλογος}\
\def\refname{Αναφορές}\
\def\abstractname{Περίληψη}\
\def\bibname{Βιβλιογραφία}\
\def\chaptername{Κεφάλαιο}\
\def\appendixname{Παράρτημα}\
\def\contentsname{Περιεχόμενα}\
\def\listfigurename{Κατάλογος σχημάτων}\
\def\listtablename{Κατάλογος πινάκων}\
\def\indexname{Ευρετήριο}\
\def\figurename{Σχήμα}\
\def\tablename{Πίνακας}\
\def\partname{Μέρος}\
\def\enclname{Συνημμένα}\
\def\ccname{Κοινοποίηση}\
\def\headtoname{Προς}\
\def\seename{βλέπε}\
\def\alsoname{βλέπε επίσης}\
\def\proofname{Απόδειξη}\
\def\glossaryname{Γλώσσαρι}

Macro \texttt{\polytonicname} is invoked when polytonic Greek is the main language of the document.

\def\polytonicname{\texttt{\polytonicname}}\def\refname{Ἀναφορὲς}\
\def\indexname{Εὑρετήριο}\
\def\figurename{Σχῆμα}\
\def\headtoname{Πρὸς}\
\def\alsoname{βέλπε ἐπίσης}\
\def\proofname{Ἀπόδειξη}\
\def\glossaryname{Γλώσσαρι}
Macro \texttt{\ancientnames} is invoked when ancient Greek is the main language of the document.

\begin{verbatim}
\def\ancientnames{%
\def\prefacename{Προοίμιον}%
\def\abstractname{Περίληψις}%
\def\bibname{Βιβλιογραφία}%
\def\chaptername{Κεφάλαιον}%
\def\appendixname{Παράρτημα}%
\def\contentsname{Περιεχόμενα}%
\def\listfigurename{Κατάλογος σχημάτων}%
\def\listtablename{Κατάλογος πινάκων}%
\def\indexname{Εὑρετήριον}%
\def\tablename{Πίναξ}%
\def\partname{Μέρος}%
\def\enclname{Συνημμένως}%
\def\ccname{Κοινοποίησις}%
\def\headtoname{Πρὸς}%
\def\pagename{Σελὶς}%
\def\seename{ὃρα}%
\def\alsoname{ὃρα ὡσαύτως}%
\def\proofname{Ἀπόδειξις}%
\def\glossaryname{Γλωσσάριον}%
\def\refname{Ἀναφοραὶ}%
\def\figurename{Σχῆμα}%
\def\headtoname{Πρὸς}%
}
\end{verbatim}

I redefine \texttt{today} so as to produce dates in Greek. The names of months are defined by the macro \texttt{\gr@month}.

\begin{verbatim}
\def\gr@month{%
\ifcase\month\or Ιανουαρίου\or Φεβρουαρίου\or Μαρτίου\or Απριλίου\or Μαΐου\or Ιουνίου\or Ιουλίου\or Αυγούστου\or Σεπτεμβρίου\or Οκτωβρίου\or Νοεμβρίου\or Δεκεμβρίου\fi}
\def\today{\number\day \space \gr@month\space \number\year}
\end{verbatim}

When either polytonic Greek or ancient Greek is the main language of the document, then the macro \texttt{\gr@c@month} becomes active.

\begin{verbatim}
\def\gr@c@month{%
\ifcase\month\or Ιανουαρίου\or Φεβρουαρίου\or Μαρτίου\or Απριλίου\or Μαΐου\or Ιουνίου\or Ιουλίου\or Αυγούστου\or Σεπτεμβρίου\or Οκτωβρίου\or Νοεμβρίου\or Δεκεμβρίου\fi}
\end{verbatim}

Next, I define a few macros that allow one to access characters that are not usually easily accessible from the keyboard (e.g., the sampi or the koppa symbol). The list includes a command for the unicode symbol GREEK ANO TELEIA, which, in some systems, is confused with MIDDLE DOT. The use of command \texttt{\numer@lsign} will be explained later.

\begin{verbatim}
\def\anwtonos{ʹ} %GREEK NUMERAL SIGN
\let\numer@lsign\anwtonos
\def\katwtonos{͵} %GREEK LOWER NUMERAL SIGN
\end{verbatim}
Many users prefer the use of the letters sigma and tau instead of the stigma symbol in Greek numerals, therefore, by default the `\stigma` command expands to “στ”. The following commands take care of the basic rules of typography. Note that the first command changes the way space is added after punctuation symbols and the last two commands force LaTeX to add indentation space to the first paragraph after a header. Since a number of users need, for their own reasons, to be able to disable this particular feature I have introduced a new package option, namely `noindentfirst`, which restores the default behavior. In order to be able to do this I need the original value of the boolean variable `\@afterindentfalse`.

LuaLaTeX and XeLaTeX have different ways to load hyphenation patterns. Package `luahyphenrules` by Javier Bezos facilitates this process for people who want to use LuaLaTeX and the “traditional” way to load hyphenation patterns. To ensure proper inclusion of LuaTeX staff, I use the following “idiom”:

```latex
\iffalse\directlua\undefined\else LuaLaTeX code\fi
\iffalse\directlua\undefined\else\RequirePackage{luahyphenrules}\fi
```

The code that follows specifies which hyphenation patterns will be active. The XeLaTeX code is quite standard and depends on the babel pattern loading mechanism, while the LuaLaTeX code uses the `\HyphenRules` macro, which has essentially the functionality of the `\selectlanguage` macro.
If a user wants to use the stigma symbol in Greek numerals, she should use the \texttt{stigma} option.

```latex
\DeclareOption{stigma}{% 
\def\stigma{\char'03DB\relax} }
```

As noted above, the new option \texttt{noindentfirst} restores the default \LaTeX{} behavior of adding no indentation to the first paragraph after any header.

```latex
\DeclareOption{noindentfirst}{% 
\let\@afterindentfalse\@saveafterindentfalse
}
```

Nowadays it is customary in Greece to use Greek numerals without the GREEK NUMERAL SIGN at the end of numeral. Thus, the \texttt{nonumeralsign} option disables the typesetting of the GREEK NUMERAL SIGN at the end of Greek numerals.

```latex
\DeclareOption{nonumeralsign}{% 
\let\numer@lsign\relax
}
```

By default the \texttt{monogreek} option is activated.

```latex
\ExecuteOptions{monogreek} \ProcessOptions
```

Now I am going to define the macros that typeset alphabetic Greek numerals. The code is borrowed from the greek option for the babel package.

\begin{itemize}
\item \texttt{\gr@ill@value} When the argument of \texttt{\greeknumeral} has a value outside of the acceptable bounds ($0 < x < 999999$) a warning will be issued (and nothing will be printed).
\end{itemize}

```latex
\def\gr@ill@value#1{\PackageWarning{xgreek}{Illegal value (#1) for greeknumeral}}
```

\begin{itemize}
\item \texttt{\anw@false} When a large number with three \textit{trailing} zeros is to be printed those zeros \textit{and} the numeric mark need to be discarded. As each ‘digit’ is processed by a separate macro \textit{and} because the processing needs to be expandable we need some helper macros that help remember to \textit{not} print the numeric mark (\texttt{\numer@lsign}).
\end{itemize}

The command \texttt{\anw@false} switches the printing of the numeric mark off by making \texttt{\anw@print} expand to nothing. The command \texttt{\anw@true} (re)enables the printing of the numeric marc. These macros need to be robust in order to prevent improper expansion during writing to files or during \texttt{\uppercase}.

```latex
\DeclareRobustCommand\anw@false{\}
\DeclareRobustCommand\anw@print{} 
\DeclareRobustCommand\anw@true{\numer@lsign}
```

\begin{itemize}
\item \texttt{\anw@true}
\end{itemize}
\@greeknumeral  This command is used to typeset Greek numerals. The command uses \numer@lsign to typeset the NUMERAL SIGN. Obviously, when the user has specified the no-numeralsign option, then numeral comes out without the trailing NUMERAL SIGN. However, when a user wants to typeset a Greek numeral, the numeral must come out correctly, regardless of what appears in headers, etc. And that is exactly the reason why this command is inaccessible to users. The command \@greeknumeral needs to be fully expandable in order to get the right information in auxiliary files. Therefore we use a big \if-construction to check the value of the argument and start the parsing at the right level.

\def\@greeknumeral#1{% 
If the value is negative or zero nothing is printed and a warning is issued.
\ifnum#1<\@ne\space\gr@ill@value{#1}%
\else
\ifnum#1<10\expandafter\gr@num@i\number#1%
\else
\ifnum#1<100\expandafter\gr@num@ii\number#1%
\else
\ifnum#1<1000\expandafter\gr@num@iii\number#1%
\else
\ifnum#1<10000\expandafter\gr@num@iv\number#1%
\else
\ifnum#1<100000\expandafter\gr@num@v\number#1%
\else
\ifnum#1<1000000\expandafter\gr@num@vi\number#1%
\else
The available shorthands for 1,000 (\@m) and 10,000 (\@M) are used to save a few tokens.
\ifnum#1<\@m\expandafter\gr@num@ii\number#1%
\else
\ifnum#1<\@M\expandafter\gr@num@i\number#1%
\else
\ifnum#1<100000\expandafter\gr@num@iv\number#1%
\else
\ifnum#1<1000000\expandafter\gr@num@v\number#1%
\else
\ifnum#1<10000000\expandafter\gr@num@vi\number#1%
\else
If the value is too large, nothing is printed and a warning is issued.
\space\gr@ill@value{#1}%
\fi
\fi
\fi
\fi
\fi
\fi
\fi
\fi
\fi
}\}

What is left to make complete the definition of command \greeknumeral is a set of macros to produce the various digits.

\gr@num@i
\gr@num@ii
\gr@num@iii

As there is no “digit” representing 0 in this system, the zeros are simply discarded. When there is a large number with three trailing zeros also the numeric mark is discarded. Therefore these macros need to pass the information to each other about the (non-)translation of a zero.

\def\gr@num@i#1{%
\ifcase#1\or α\or β\or γ\or δ\or ε\or \stigma\or ζ\or η\or θ\fi

The alphabetic numbering system is not the only numbering system employed by Greeks. In fact, Greeks used various systems that are now known as acrophonic numbering systems. Many scholars are familiar with the acrophonic Attic...
numbering system and the the command \atticnum can be used to generate acro-
phonic Attic numerals. The acrophonic Attic numbering system, like the Roman
one, employs letters to denote important numbers. Multiple occurrence of a let-
ter denote a multiple of the “important” number, e.g., the letter I denotes 1, so III denotes 3. Here are the basic digits used in the acrophonic Attic numbering
system:

- I denotes the number one (1)
- II denotes the number five (5)
- Δ denotes the number ten (10)
- Η denotes the number one hundred (100)
- Χ denotes the number one thousand (1000)
- Μ denotes the number ten thousands (10000)

Moreover, the letters Δ, Η, Χ, and Μ under the letter Π (a form of II) denote five
times their original value. In particular, the symbol Π denotes the number 50, the
symbol Π denotes the number 500, the symbol Π denotes the number 5000, and
the symbol Π denotes the number 50,000. It must be noted that the numbering
system does not provide negative numerals or a symbol for zero.

\@@atticnum Now, let me definite the macro \@@atticnum. This macro uses one integer variable
(or counter in ‘\TeX’s jargon.)
\newcount\@@atticnum

The macro \@@atticnum is also defined as a robust command.
\DeclareRobustCommand*{\@@atticnum}[1]{% After assigning to variable \@@atticnum the value of the macro’s argument, we
make sure that the argument is in the expected range, i.e., it is greater than zero,
and less or equal to 249999. In case it isn’t, it simply produces a \space, warns the
user about it and quits. Although, the \atticnum macro is capable to produce an
Athenian numeral for even greater integers, the following argument by Claudio
Beccari convinced me to place this upper limit:

According to psychological perception studies (that ancient Atheni-
ans and Romans perfectly knew without needing to study Freud and
Jung) living beings (which includes at least all vertebrates, not only
humans) can perceive up to four randomly set objects of the same kind
without the need of counting, the latter activity being a specific ac-
quired ability of human kind; the biquinary numbering notation used
by the Athenians and the Romans exploits this natural characteristic
of human beings.

\@@atticnum#1\relax
\ifnum\@@atticnum<\@ne%
\space%

Having done all the necessary checks, it is possible to proceed with the actual computation. If the number is greater than 49999, then it certainly has at least one \( \text{𐅇} \) “digit”. The macro finds all such digits by continuously subtracting 50000 from \( \@\text{attic@num} \), until \( \@\text{attic@num} \) becomes less than 50000.

Next the macro checks for tens of thousands.

Since a number can have only one \( \text{𐅆} \) “digit” (equivalent to 5000), it is easy to check whether it should have one and produce the corresponding numeral when it does have one.

The macro should also check for thousands, the same way it checked for tens of thousands.

Since a numeral can have at most one \( \text{𐅅} \) “digit” (equivalent to 500), this should be handled the way the macro handled the case of the five thousands “digit”.

It is time to check hundreds, which follow the same pattern as thousands.

A numeral can have only one \( \text{𐅄} \) “digit” (equivalent to 50).

The macro now checks now for tens digit.

Finally, it has to check for fives and the digits 1, 2, 3, and 4.
The command \@atticnum has one argument, which is a counter. It calls the command \@@atticnum to process the value of the counter.
\def\@atticnum#1{% 
\expandafter\@@atticnum\expandafter{\the#1}}

The command \atticnum is a wrapper that declares a new counter in a local scope, assigns to it the value of the argument of the command and calls the macro \@atticnum. This way the command can process correctly either a number or a counter.
\def\atticnum#1{% 
\@attic@num#1\relax 
\@atticnum{\@attic@num}}

Here I redefine the macros \@alph and \@Alph. First, I define some placeholders
\let\latin@alph\@alph
\let\latin@Alph\@Alph
Then I define the Greek versions; the additional \expandafter s are needed in order to make sure the table of contents will be correct (e.g., when there are appendices).
\def\greek@alph#1{\expandafter\@greeknumeral\expandafter{\the#1}}
\def\greek@Alph#1{\expandafter\@Greeknumeral\expandafter{\the#1}}

By default Greek alphabetic enumerals instead of Latin numerals are used to enumerate items in an enumeration environment.
\let\@alph\greek@alph
\let\@Alph\greek@Alph
If for some reason, one needs to have the Latin numerals back, then she has to invoke command \nogreekalph. And if she wants to switch back, then she has to use the \greekalph command:
\def\nogreekalph{% 
\let\@alph\latin@alph 
\let\@Alph\latin@Alph}
\def\greekalph{% 
\let\@alph\greek@alph 
\let\@Alph\greek@Alph}

We provide the \setlanguage command which activates the hypenlation patterns of some other language. It is similar to babel’s \selectlanguage, but we opted to use a new name to avoid possible name conflicts. Valid arguments include monogreek, polygreek, ancientgreek, and american. As was noted previously, package luahyphenrules provides the command \HyphenRules which has exactly the same functionality as this command. So when using Lua\LaTeX users will actually use the \HyphenRules command.
The macros \grtoday and \Grtoday produces the current date, only that the
month and the day are shown as greek numerals instead of arabic as it is usually
the case. In addition, the two commands differ in that the later produces the
Greek numerals in uppercase.