Greek Language Support for \texttt{XƎL\LaTeX} and LuaL\LaTeX

Apostolos Syropoulos  
Xanthi, Greece  
asyropoulos@yahoo.com  

2009/11/23  
Last Updated 2022/09/04

Abstract
The \texttt{xgreek} package provides rudimentary support for Greek language typesetting with \texttt{XƎL\LaTeX} and LuaL\LaTeX. 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{XƎL\LaTeX} and LuaL\LaTeX. 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 LuaL\LaTeX was provided by Javier Bezos.

2 The Source Code of \texttt{xgreek}
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{XƎL\LaTeX} blindly follows the Unicode standard, commands like \texttt{\MakeUppercase} produce wrong output. For this reason I first need to set up the correct \texttt{lccodes} and \texttt{uccodes}.

\begin{verbatim}
1 \langle*xgreek\rangle
2 \message{Package 'xgreek' version 3.1.0 by Apostolos Syropoulos}
3 \gdef{lccode}{0370\leq371} \gdef{uccode}{0370\leq370}
4 \gdef{lccode}{0371\leq371} \gdef{uccode}{0371\leq370}
5 \gdef{lccode}{0372\leq373} \gdef{uccode}{0372\leq372}
6 \gdef{lccode}{0373\leq373} \gdef{uccode}{0373\leq373}
7 \gdef{lccode}{0376\leq377} \gdef{uccode}{0376\leq376}
8 \gdef{lccode}{0377\leq377} \gdef{uccode}{0377\leq376}
9 \gdef{lccode}{03FD\leq03FB} \gdef{uccode}{03FD\leq03FD}
\end{verbatim}

1
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}
I redefine \today so as to produce dates in Greek. The names of months are defined by the macro \gr@month.

\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 or}

I redefine \today so as to produce dates in Greek. The names of months are defined by the macro \gr@month.
When either polytonic Greek or ancient Greek is the main language of the document, then the macro \texttt{\gr@c@month} becomes active.

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{\numeralsign} will be explained later.

Many users prefer the use of the letters sigma and tau instead of the stigma symbol in Greek numerals, therefore, by default the \texttt{\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 \texttt{noindentfirst}, which restores the default behavior. In order to be able to do this I need the original value of the boolean variable \texttt{@afterindentfalse}.

LuaLaTeX and XeLaTeX have different ways to load hyphenation patterns. Package \texttt{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”:

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

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

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.

Package \texttt{listings} does not work properly with UTF-8 encoded files. So this option should be used whenever one wants to use this package and see Greek text come out correctly. In version 3.1.0 of this package, I had included the code that modifies package \texttt{listings} in the source code of this package. However, this decision was wrong since when one does not use the \texttt{listings} option, processing of the input file stops with an error message about a text line that contains an invalid character. So the best way to solve this problem is to move the code in a different file and load it when the user has specified \texttt{listings} option. To enable this option, I use a boolean variable.

\begin{verbatim}
newif\if@mylistings
\@mylistingsfalse
\DeclareOption{listings}{\@mylistingstrue}
\end{verbatim}

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

\gr@ill@value When the argument of \greeknumeral has a value outside of the acceptable bounds (0 < x < 999999) a warning will be issued (and nothing will be printed).

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

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

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

\DeclareRobustCommand\anw@false{% 
\DeclareRobustCommand\anw@print{}}
\DeclareRobustCommand\anw@true{% 
\DeclareRobustCommand\anw@print{\numer@lsign}}
\anw@true

\greeknumeral This command is used to get Greek numerals. The command uses \numer@lsign to get 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<\one\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
 \ifnum#1<10000000\expandafter\gr@num@vii\number#1%
 \else
 \ifnum#1<100000000\expandafter\gr@num@viii\number#1%
 \else
 \ifnum#1<1000000000\expandafter\gr@num@ix\number#1%
 \else
 \ifnum#1<10000000000\expandafter\gr@num@x\number#1%
 \else
 \ifnum#1<100000000000\expandafter\gr@num@xi\number#1%
If the value is too large, nothing is printed and a warning is issued.

\else
  \space\texttt{gr\@ill\@value{#1}}\%
\fi
\fi
\fi
\fi
\fi
\fi\fi

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

\gr@num@i 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\ifnum#1=\z@\else\anw@true\fi\anw@print}
\def\gr@num@ii#1{\ifcase#1\or ι\or κ\or λ\or μ\or ν\or ξ\or ο\or π\or \koppa\fi\ifnum#1=\z@\else\anw@true\fi\gr@num@i}
\def\gr@num@iii#1{\ifcase#1\or ρ\or σ\or τ\or υ\or φ\or χ\or ψ\or ω\or \sampi\fi\ifnum#1=\z@\anw@false\else\anw@true\fi\gr@num@ii}
\def\gr@num@iv#1{\katwtonos\ifcase#1\or α\or β\or γ\or δ\or ε\or \stigma\or ζ\or η\or θ\fi\gr@num@iii}
\def\gr@num@v#1{\katwtonos\ifcase#1\or ι\or κ\or λ\or μ\or ν\or ξ\or ο\or π\or \koppa\fi\gr@num@iv}
\def\gr@num@vi#1{\katwtonos\ifcase#1\or ρ\or σ\or τ\or υ\or φ\or χ\or ψ\or ω\or \sampi\fi\gr@num@v}

The first three “digits” always have the numeric mark, except when one is discarded because it's value is zero.

\def\gr@num@i#1{%\ifnum#1=\z@\else\katwtonos\fi\ifcase#1\or α\or β\or γ\or δ\or ε\or \stigma\or ζ\or η\or θ\fi\gr@num@ii}
\def\gr@num@iv#1{%\ifnum#1=\z@\else\katwtonos\fi\ifcase#1\or α\or β\or γ\or δ\or ε\or \stigma\or ζ\or η\or θ\fi\gr@num@iii}
\def\gr@num@v#1{%\ifnum#1=\z@\else\katwtonos\fi\ifcase#1\or α\or β\or γ\or δ\or ε\or \stigma\or ζ\or η\or θ\fi\gr@num@iv}
\def\gr@num@vi#1{%\katwtonos\ifcase#1\or α\or β\or γ\or δ\or ε\or \stigma\or ζ\or η\or θ\fi\gr@num@v}

\Greeknumeral The command \texttt{\Greeknumeral} prints uppercase Greek numerals. The parsing is performed by the macro \texttt{\Greeknumeral}. The printing of the NUMERAL SIGN depends on the value of \texttt{\numer@lsign}.

\def\Greeknumeral#1{%\expandafter\MakeUppercase\expandafter{\Greeknumeral{#1}}}

\greeknumeral This command prints lowercase Greek numerals and the NUMERAL SIGN is always printed.
This command prints uppercase Greek numerals and the NUMERAL SIGN is always printed.

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 command \atticnum can be used to generate acrophonic Attic numerals. The acrophonic Attic numbering system, like the Roman one, employs letters to denote important numbers. Multiple occurrence of a letter 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:

- Ι denotes the number one (1)
- Π 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 𐅃 (a form of Π) denote five times their original value. In particular, the symbol 𐅄 denotes the number 50, the symbol 𐅅 denotes the number 500, the symbol 𐅆 denotes the number 5000, and the symbol 𐅇 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.)

The macro \@atticnum is also defined as a robust command.

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 Athenians 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 acquired ability of human kind; the biquinary numbering notation used by the Athenians and the Romans exploits this natural characteristic of human beings.

498 \@attic@num#1\relax
499 \ifnum\@attic@num<\@one\space%
500 \PackageWarning{xgreek}{%
501 Illegal value (\the\@attic@num) for acrophonic
502 Attic numeral}%
503 \else\ifnum\@attic@num>249999\%
504 \space%
505 \PackageWarning{xgreek}{%
506 Value too large (\the\@attic@num) for acrophonic
507 Attic numeral}%
508 \else
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 $𐅇$ “digit”. The macro finds all such digits by continuously subtracting 50000 from \@attic@num, until \@attic@num becomes less than 50000.
510 \@whilenum\@attic@num>49999\do{%
511 \ifnum\@attic@num>4999%
512 \ifnum\@attic@num>49999\do(%
513 \ifnum\@attic@num>249999\%
514 \else
Next the macro checks for tens of thousands.
512 \@whilenum\@attic@num>9999\do{%
513 \advance\@attic@num-\@m}%
Since a number can have only one $𐅇$ “digit” (equivalent to 5000), it is easy to check whether is should have one and produce the corresponding numeral when it does have one.
514 \ifnum\@attic@num>4999%
515 \ifnum\@attic@num>49999\do(%
516 \fi\relax
The macro should also check for thousands, the same way it checked for tens of thousands.
517 \@whilenum\@attic@num>999\do{%
518 \advance\@attic@num-\@m}%
Since a numeral can have at most one $𐅇$ “digit” (equivalent to 500), this should be handled the way the macro handled the case of the five thousands “digit”.
519 \ifnum\@attic@num>499%
520 \ifnum\@attic@num>4999\do(%
521 \fi\relax
It is time to check hundreds, which follow the same pattern as thousands.
522 \@whilenum\@attic@num>99\do{%
523 \ifnum\@attic@num>499%
524 \ifnum\@attic@num>4999\do(%
525 \fi\relax
14
A numeral can have only one “digit” (equivalent to 50).

\ifnum\@attic@num>49%
  \@case\@attic@num010144\advance\@attic@num-50%
\fi
\relax

The macro now checks now for tens digit.

\@whilenum\@attic@num>9\do{%
  \@case\@attic@num010144\advance\@attic@num by-10%
}\fi

Finally, it has to check for fives and the digits 1, 2, 3, and 4.

\@whilenum\@attic@num>4\do{%
  \@case\@attic@num010144\advance\@attic@num-5%
}\fi
\@case\@attic@num010144
\fi
\fi}

\@atticnum 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}}

\@atticnum 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}}

\greek@alph \greek@Alph Here I redefine the macros \@alph and \@Alph. First, I define some placeholders

\def\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 instaed 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 \texttt{\setlanguage} command which activates the hyphenation patterns of some other language. It is similar to babel's \texttt{\selectlanguage}, but we opted to use a new name to avoid possible name conflicts. Valid arguments include \texttt{monogreek}, \texttt{polygreek}, \texttt{ancientgreek}, and \texttt{american}. As was noted previously, package \texttt{luahyphenrules} provides the command \texttt{\HyphenRules} which has exactly the same functionality as this command. So when using LuaLaTeX users will actually use the \texttt{\HyphenRules} command.

\begin{verbatim}
\def\setlanguage#1{% 
\expandafter\ifx\csname l@#1\endcsname\relax% 
\typeout{\textasciitilde J Error: No hyphenation pattern for}#1\% 
\typeout{ default hyphenation patterns are used.\textasciitilde J}\% 
\language=0% 
\else\language=\csname l@#1\endcsname\fi}
\end{verbatim}

The macros \texttt{\grtoday} and \texttt{\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.

\begin{verbatim}
\def\grtoday{% 
\expandafter\greeknumeral\expandafter{\the\day}\space 
\gr@c@month\space 
\expandafter\greeknumeral\expandafter{\the\year}}
\def\Grtoday{% 
\expandafter\Greeknumeral\expandafter{\the\day}\space 
\gr@c@month\space 
\expandafter\Greeknumeral\expandafter{\the\year}}
\end{verbatim}

\section{The Source Code of xelisting}

If the user has enabled the \texttt{listings} option, then the package loads the rudimentary package \texttt{xelisting}. This package loads the \texttt{listings} package and makes accessible to it all characters in the range 128–255 plus all Greek letters that belong to the Greek and Coptic Unicode block. This is achieved by redefining the command \texttt{\lst@DefEC}.

\begin{verbatim}
\RequirePackage{listings}
\lstset{inputencoding=utf8}
\lst@InputCatcodes
\gdef\lst@DefEC{% 
\lst@CCECUse \lst@ProcessLetter 
\textasciitilde80\textasciitilde81\textasciitilde82\textasciitilde83\textasciitilde84\textasciitilde85\textasciitilde86\textasciitilde87\textasciitilde88\textasciitilde89\textasciitilde8a\textasciitilde8b\textasciitilde8c\textasciitilde8d\textasciitilde8e\textasciitilde8f\% 
\textasciitilde90\textasciitilde91\textasciitilde92\textasciitilde93\textasciitilde94\textasciitilde95\textasciitilde96\textasciitilde97\textasciitilde98\textasciitilde99\textasciitilde9a\textasciitilde9b\textasciitilde9c\textasciitilde9d\textasciitilde9e\textasciitilde9f\% 
\textasciitildea0\textasciitildea1\textasciitildea2\textasciitildea3\textasciitildea4\textasciitildea5\textasciitildea6\textasciitildea7\textasciitildea8\textasciitildea9\textasciitildeaa\textasciitildeab\textasciitildeac\textasciitildead\textasciitildeae\textasciitildeaf\% 
\textasciitildeb0\textasciitildeb1\textasciitildeb2\textasciitildeb3\textasciitildeb4\textasciitildeb5\textasciitildeb6\textasciitildeb7\textasciitildeb8\textasciitildeb9\textasciitildeba\textasciitildebb\textasciitildebc\textasciitildebd\textasciitildebe\textasciitildebf\% 
\textasciitildec0\textasciitildec1\textasciitildec2\textasciitildec3\textasciitildec4\textasciitildec5\textasciitildec6\textasciitildec7\textasciitildec8\textasciitildec9\textasciitildeca\textasciitildecb\textasciitildecc\textasciitildecd\textasciitildece\textasciitildecf\% 
\end{verbatim}