Greek Language Support for XeLaTeX and LuaLaTeX

Apostolos Syropoulos
Xanthi, Greece
asyropoulos@yahoo.com

2009/11/23
Last Updated 2026/07/06

Abstract

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

1 Introduction

The xgreek package provides rudimentary support for Greek language typesetting with XeLaTeXand 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 LuaLaTeX was provided by Javier Bezos.

2 The Source Code of xgreek

According to the Unicode standard (http://www.unicode.org/Public/UNIDATA/UnicodeData.txt), the uppercase form of the letter GREEK SMALL LETTER EPSILON WITH TONOS is represented as the letter GREEK CAPITAL LETTER EPSILON WITH TONOS. However, this is incorrect according to the grammar of the Greek language. When text is transformed into uppercase, all letters lose their accents. If a letter contains GREEK DIALYTIKA TONOS, only the DIALYTIKA remain. Thus, the uppercase form of the letter GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS is the letter GREEK CAPITAL LETTER IOTA WITH DIALYTIKA. Similarly, the uppercase form of the letter GREEK SMALL LETTER EPSILON WITH TONOS is the letter GREEK CAPITAL LETTER EPSILON.

For ancient Greek, a similar rule applies: all accents and breathings disappear, and only letters with DIALYTIKA retain them. For instance, the uppercase form of the letter GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND PERISPOMENI is the letter GREEK CAPITAL LETTER UPSILON WITH DIALYTIKA. Furthermore, for any lowercase letter that has a GREEK YPOGEGRAMMENI, the corresponding uppercase letter is the one with a PROSGEGRAMMENI. For example, the uppercase form of the letter GREEK SMALL LETTER ETA WITH OXIA AND YPOGEGRAMMENI is the letter GREEK CAPITAL LETTER ETA WITH PROSGEGRAMMENI.

There is, however, one exception: the word GREEK SMALL LETTER ETA WITH PSILI AND OXIA (which is the equivalent of GREEK SMALL LETTER ETA WITH TONOS in Modern Greek). This word corresponds to the English word "or." To avoid confusion with the feminine article in the nominative case (i.e., the letter GREEK SMALL LETTER ETA WITH DASIA), this article retains the accent in uppercase letters only text. Unfortunately, implementing this feature poses challenges, as it would necessitate a rule of the form

“SPACE” “GREEK SMALL LETTER ETA WITH PSILI AND OXIA” “SPACE”
—>
“SPACE” “GREEK CAPITAL LETTER ETA WITH OXIA” “SPACE”

LaTeX *blindly* follows the Unicode standard and so the command `\MakeUppercase` produces wrong output as far as the Greek language is concerned. This is also the reason why I have put a lot of `\uccode` and `\lccode` pairs in the previous versions of this package to fix this problem. I very recently noticed that the results from these commands are not as expected since the new implementation of `\MakeUppercase` command does not care about these commands at all. Fortunately the person who did this re-implementation defined the new command `\DeclareUppercaseMapping` that maps the character code of a lower case letter to the actual upper case character. The following commands implement the grammatical rules of the Greek language, except of course for the rule for the Greek disjunctive conjunction.

```
⟨*xgreek⟩
\message{Package 'xgreek' version 3.5.1 by Apostolos Syropoulos}
\DeclareUppercaseMapping{"0386}{A}
\DeclareUppercaseMapping{"0388}{E}
\DeclareUppercaseMapping{"0389}{H}
\DeclareUppercaseMapping{"038A}{I}
\DeclareUppercaseMapping{"038C}{O}
\DeclareUppercaseMapping{"038E}{T}
\DeclareUppercaseMapping{"038F}{Ω}
\DeclareUppercaseMapping{"0390}{Ϊ} \DeclareLowercaseMapping{"0390}{ΐ}
\DeclareUppercaseMapping{"03AC}{A} \DeclareLowercaseMapping{"03AC}{ά}
\DeclareUppercaseMapping{"03AD}{E} \DeclareLowercaseMapping{"03AD}{έ}
\DeclareUppercaseMapping{"03AE}{H} \DeclareLowercaseMapping{"03AE}{ή}
\DeclareUppercaseMapping{"03AF}{I} \DeclareLowercaseMapping{"03AF}{ί}
\DeclareUppercaseMapping{"03B0}{Ϋ} \DeclareLowercaseMapping{"03B0}{ΰ}
\DeclareUppercaseMapping{"03CA}{Ϊ} \DeclareLowercaseMapping{"03CA}{ϊ}
\DeclareUppercaseMapping{"03CB}{Ϋ} \DeclareLowercaseMapping{"03CB}{ϋ}
\DeclareUppercaseMapping{"03CC}{O} \DeclareLowercaseMapping{"03CC}{ό}
\DeclareUppercaseMapping{"03CD}{T} \DeclareLowercaseMapping{"03CD}{ύ}
\DeclareUppercaseMapping{"03CE}{Ω} \DeclareLowercaseMapping{"03CE}{ώ}
\DeclareUppercaseMapping{"1F00}{A} \DeclareUppercaseMapping{"0386}{A}
\DeclareUppercaseMapping{"1F01}{A} \DeclareUppercaseMapping{"0388}{E}
\DeclareUppercaseMapping{"1F02}{A} \DeclareUppercaseMapping{"0389}{H}
\DeclareUppercaseMapping{"1F03}{A} \DeclareUppercaseMapping{"038A}{I}
\DeclareUppercaseMapping{"1F04}{A} \DeclareUppercaseMapping{"038C}{O}
\DeclareUppercaseMapping{"1F05}{A} \DeclareUppercaseMapping{"038E}{T}
\DeclareUppercaseMapping{"1F06}{A} \DeclareUppercaseMapping{"038F}{Ω}
\DeclareUppercaseMapping{"1F07}{A}
\DeclareUppercaseMapping{"1F10}{E}
\DeclareUppercaseMapping{"1F11}{E}
\DeclareUppercaseMapping{"1F12}{E}
\DeclareUppercaseMapping{"1F13}{E}
\DeclareUppercaseMapping{"1F14}{E}
\DeclareUppercaseMapping{"1F15}{E}
\DeclareUppercaseMapping{"1F20}{H}
\DeclareUppercaseMapping{"1F21}{H}
\DeclareUppercaseMapping{"1F22}{H}
\DeclareUppercaseMapping{"1F23}{H}
\DeclareUppercaseMapping{"1F24}{H}
\DeclareUppercaseMapping{"1F25}{H}
\DeclareUppercaseMapping{"1F26}{H}
\DeclareUppercaseMapping{"1F27}{H}
\DeclareUppercaseMapping{"1F30}{I}
\DeclareUppercaseMapping{"1F31}{I}
\DeclareUppercaseMapping{"1F32}{I}
\DeclareUppercaseMapping{"1F33}{I}
\DeclareUppercaseMapping{"1F34}{I}
```

```
\DeclareUppercaseMapping{"1F35}{I}
\DeclareUppercaseMapping{"1F36}{I}
\DeclareUppercaseMapping{"1F37}{I}
\DeclareUppercaseMapping{"1F40}{O}
\DeclareUppercaseMapping{"1F41}{O}
\DeclareUppercaseMapping{"1F42}{O}
\DeclareUppercaseMapping{"1F43}{O}
\DeclareUppercaseMapping{"1F44}{O}
\DeclareUppercaseMapping{"1F45}{O}
\DeclareUppercaseMapping{"1F50}{T}
\DeclareUppercaseMapping{"1F51}{T}
\DeclareUppercaseMapping{"1F52}{T}
\DeclareUppercaseMapping{"1F53}{T}
\DeclareUppercaseMapping{"1F54}{T}
\DeclareUppercaseMapping{"1F55}{T}
\DeclareUppercaseMapping{"1F56}{T}
\DeclareUppercaseMapping{"1F57}{T}
\DeclareUppercaseMapping{"1F60}{Ω}
\DeclareUppercaseMapping{"1F61}{Ω}
\DeclareUppercaseMapping{"1F62}{Ω}
\DeclareUppercaseMapping{"1F63}{Ω}
\DeclareUppercaseMapping{"1F64}{Ω}
\DeclareUppercaseMapping{"1F65}{Ω}
\DeclareUppercaseMapping{"1F66}{Ω}
\DeclareUppercaseMapping{"1F67}{Ω}
\DeclareUppercaseMapping{"1F70}{A}
\DeclareUppercaseMapping{"1F71}{A}
\DeclareUppercaseMapping{"1F72}{E}
\DeclareUppercaseMapping{"1F73}{E}
\DeclareUppercaseMapping{"1F74}{H}
\DeclareUppercaseMapping{"1F75}{H}
\DeclareUppercaseMapping{"1F76}{I}
\DeclareUppercaseMapping{"1F77}{I}
\DeclareUppercaseMapping{"1F78}{O}
\DeclareUppercaseMapping{"1F79}{O}
\DeclareUppercaseMapping{"1F7A}{T}
\DeclareUppercaseMapping{"1F7B}{T}
\DeclareUppercaseMapping{"1F7C}{Ω}
\DeclareUppercaseMapping{"1F7D}{Ω}
\DeclareUppercaseMapping{"1F80}{A_}
\DeclareUppercaseMapping{"1F81}{A_}
\DeclareUppercaseMapping{"1F82}{A_}
\DeclareUppercaseMapping{"1F83}{A_}
\DeclareUppercaseMapping{"1F84}{A_}
\DeclareUppercaseMapping{"1F85}{A_}
\DeclareUppercaseMapping{"1F86}{A_}
\DeclareUppercaseMapping{"1F87}{A_}
\DeclareUppercaseMapping{"1F90}{H_}
\DeclareUppercaseMapping{"1F91}{H_}
\DeclareUppercaseMapping{"1F92}{H_}
\DeclareUppercaseMapping{"1F93}{H_}
\DeclareUppercaseMapping{"1F94}{H_}
\DeclareUppercaseMapping{"1F95}{H_}
\DeclareUppercaseMapping{"1F96}{H_}
\DeclareUppercaseMapping{"1F97}{H_}
\DeclareUppercaseMapping{"1FA0}{Ω_}
\DeclareUppercaseMapping{"1FA1}{Ω_}
\DeclareUppercaseMapping{"1FA2}{Ω_}
\DeclareUppercaseMapping{"1FA3}{Ω_}
\DeclareUppercaseMapping{"1FA4}{Ω_}
\DeclareUppercaseMapping{"1FA5}{Ω_}
```

```
\DeclareUppercaseMapping{"1FA6}{Ω}
\DeclareUppercaseMapping{"1FA7}{Ω}
\DeclareUppercaseMapping{"1FB2}{Α}
\DeclareUppercaseMapping{"1FB3}{Α}
\DeclareUppercaseMapping{"1FB4}{Α}
\DeclareUppercaseMapping{"1FB6}{Α}
\DeclareUppercaseMapping{"1FB7}{Α}
\DeclareUppercaseMapping{"1FC2}{Η}
\DeclareUppercaseMapping{"1FC3}{Η}
\DeclareUppercaseMapping{"1FC4}{Η}
\DeclareUppercaseMapping{"1FC6}{Η}
\DeclareUppercaseMapping{"1FC7}{Η}
\DeclareUppercaseMapping{"1FD2}{Ϊ}
\DeclareUppercaseMapping{"1FD3}{Ϊ}
\DeclareUppercaseMapping{"1FD6}{Ι}
\DeclareUppercaseMapping{"1FD7}{Ϊ}
\DeclareUppercaseMapping{"1FE2}{Ϋ}
\DeclareUppercaseMapping{"1FE3}{Ϋ}
\DeclareUppercaseMapping{"1FE4}{Ρ}
\DeclareUppercaseMapping{"1FE5}{Ρ}
\DeclareUppercaseMapping{"1FE6}{Υ}
\DeclareUppercaseMapping{"1FE7}{Ϋ}
\DeclareUppercaseMapping{"1FF2}{Ω}
\DeclareUppercaseMapping{"1FF3}{Ω}
\DeclareUppercaseMapping{"1FF4}{Ω}
\DeclareUppercaseMapping{"1FF6}{Ω}
\DeclareUppercaseMapping{"1FF7}{Ω}
```

Next, I define the various strings that correspond to the standard LaTeX captions. First, I 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\pagename{Σελίδα}%
\def\seename{βλέπε}%
\def\alsoname{βλέπε επίσης}%
\def\proofname{Απόδειξη}%
\def\glossaryname{Γλωσσάρι}%
```

Macro `\polytonicn@mes` is invoked when polytonic Greek is the main language of the document.

```
\def\polytonicn@mes{%
  \def\refname{Ἀναφορὲς}%
  \def\indexname{Εὑρετήριο}%
  \def\figurename{Σχῆμα}%
  \def\headtoname{Πρὸς}%
  \def\alsoname{βλέπε ἐπίσης}%
  \def\proofname{Ἀπόδειξη}%
}
```

Macro `\@ncientn@mes` is invoked when ancient Greek is the main language of the document (Dinitrios Filippou spotted a couple of spelling errors in the list below).

```
\def\@ncientn@mes{%
  \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{ὅρα}%
  \def\alsoname{ὅρα ὡσαύτως}%
  \def\proofname{Ἀπόδειξις}%
  \def\glossaryname{Γλωσσάριον}%
  \def\refname{Ἀναφοραὶ}%
  \def\figurename{Σχῆμα}%
  \def\headtoname{Πρὸς}%
}
```

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\fi}
\def\today{\number\day \space \gr@month\space \number\year}
```

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

```
\def\gr@c@month{%
  \ifcase\month\or Ἰανουαρίου\or Φεβρουαρίου\or Μαρτίου\or Ἀπριλίου\or
   Μαΐου\or Ἰουνίου\or Ἰουλίου\or Αὐγούστου\or Σεπτεμβρίου\or
  Ὀκτωβρίου\or Νοεμβρίου\or Δεκεμβρίου\fi}
```

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 `\numer@lsign` will be explained later.

```
\def\anwtonos{΄} %GREEK NUMERAl SIGN
\let\numer@lsign\anwtonos
\def\katwtonos{͵} %GREEK LOWER NUMERAL SIGN
\def\koppa{\char"03DF\relax}
\def\sampi{\char"03E1\relax}
\def\Digamma{\char"03DC\relax}
\def\ddigamma{\char"03DD\relax}
\def\anoteleia{\char"0387\relax}
\def\euro{\char"20AC\relax}
\def\permill{\char"2030\relax}
```

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".

```
\def\stigma{στ\relax}
```

The following commands address the fundamental rules of typography. Note that the first command alters the way space is added after punctuation marks, while the last two commands compel LaTeX to include indentation space at the beginning of the first paragraph following a header. Since several users require the ability to disable this specific feature for their own reasons, I have introduced a new package option called `noindentfirst`, which restores the default behavior. To implement this, I need the original value of the boolean variable `\@afterindentfalse`.

```
\frenchspacing
\let\@saveafterindentfalse\@afterindentfalse
\let\@afterindentfalse\@afterindenttrue
\@afterindenttrue
```

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":

```
\ifx\directlua\undefined non LuaLaTeX code\else  LuaLaTeX code\fi
```

```
\ifx\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.

```
\DeclareOption{monogreek}{%
   \ifx\directlua\undefined%
   \language\l@monogreek\else\HyphenRules{monogreek}\fi%
}
\DeclareOption{polygreek}{%
   \ifx\directlua\undefined%
   \language\l@polygreek\else\HyphenRules{polygreek}\fi%
   \polytonicn@mes%
   \let\gr@month\gr@c@month%
}
\DeclareOption{ancientgreek}{%
   \ifx\directlua\undefined%
   \language\l@ancientgreek\else\HyphenRules{ancientgreek}\fi%
   \@ncientn@mes%
   \let\gr@month\gr@c@month%
}
```

If a user wants to use the stigma symbol in Greek numerals, she should use the `stigma` option.

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

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

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

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

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

Package 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 included code that modified the source code of package listings. However, this decision was wrong. In particular, when one did not use the corresponding `listings` option, processing of the input file stopped with an error message about a text line that contains an invalid character. So the best way to solve

this problem was to move the code to a different file and create essentially a new package. This package is automatically loaded when the user pecifies the `listings` option. To make this possible, I used a boolean variable.

```
\newif\if@mylistings
\@mylistingsfalse
\DeclareOption{listings}{\@mylistingstrue}
```

By default the `monogreek` option is activated.

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

If the user has enabled the `listings` option, then the package loads the package xelistings.

```
\if@mylistings
\RequirePackage{xelistings}
\fi
```

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 printing a large number with three *trailing* zeros, those zeros *and* the numeric mark must be discarded. Since each digit is processed by a separate macro *and* the processing needs to be expandable, we require helper macros to ensure that the numeric mark (`\numer@lsign`) is not printed.

The command `\anw@false` disables the printing of the numeric mark by making `\anw@print` expand to nothing. Conversely, the command `\anw@true` (re)enables the printing of the numeric mark. These macros must be robust to prevent improper expansion during file writing or when using `\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 `nonumeralsign` 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
```

The available shorthands for 1.000 (`\@m`) and 10.000 (`\@M`) are used to save a few tokens.

```
        \ifnum#1<\@m\expandafter\gr@num@iii\number#1%
        \else
          \ifnum#1<\@M\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
```

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
}
```

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
  \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}
```

`\gr@num@iv` `\gr@num@v` `\gr@num@vi` The first three "digits" always have the numeric mark, except when one is discarded because it's value is zero.

```
\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 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}
```

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

```
\def\@Greeknumeral#1{%
  \expandafter\MakeUppercase\expandafter{\@greeknumeral{#1}}}
```

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

```
\def\greeknumeral#1{%
  \let\@numer@lsign\numer@lsign%
  \let\numer@lsign\anwtonos%
  \@greeknumeral{#1}
  \let\numer@lsign\@numer@lsign}
```

`\Greeknumeral` This command prints uppercase Greek numerals and the NUMERAL SIGN is always printed.

```
\def\Greeknumeral#1{%
  \let\@numer@lsign\numer@lsign%
  \let\numer@lsign\anwtonos%
  \@Greeknumeral{#1}
  \let\numer@lsign\@numer@lsign}
```

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 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:

- I denotes the number one (1)
- Π denotes the number five (5)
- Δ denotes the number ten (10)
- H denotes the number one hundred (100)
- X denotes the number one thousand (1000)
- M denotes the number ten thousands (10000)

Moreover, the letters Δ, H, X, and M under the letter Γ (a form of Π) denote five times their original value. In particular, the symbol 𝈥, denotes the number 50, the symbol 𝈦 denotes the number 500, the symbol 𝈧 denotes the number 5000, and the symbol 𝈨 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\@attic@num
```

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

```
\DeclareRobustCommand*{\@@atticnum}[1]{%
```

After assigning to variable `\@attic@num` 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 is not, 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 intergers, 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.

```
      \@attic@num#1\relax
      \ifnum\@attic@num<\@ne%
        \space%
        \PackageWarning{xgreek}{%
        Illegal value (\the\@attic@num) for acrophonic
        Attic numeral}%
      \else\ifnum\@attic@num>249999%
        \space%
        \PackageWarning{xgreek}{%
        Value too large (\the\@attic@num) for acrophonic
        Attic numeral}%
      \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 𝈧 "digit". The macro finds all such digits by continuously subtracting 50000 from `\@attic@num`, until `\@attic@num` becomes less than 50000.

```
          \@whilenum\@attic@num>49999\do{%
             ^^^^^^010147\advance\@attic@num-50000}%
```

Next the macro checks for tens of thousands.

```
            \@whilenum\@attic@num>9999\do{%
               M\advance\@attic@num-\@M}%
```

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

```
            \ifnum\@attic@num>4999%
               ^^^^^^010146\advance\@attic@num-5000%
            \fi\relax
```

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

```
            \@whilenum\@attic@num>999\do{%
               X\advance\@attic@num-\@m}%
```

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

```
            \ifnum\@attic@num>499%
               ^^^^^^010145\advance\@attic@num-500%
            \fi\relax
```

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

```
            \@whilenum\@attic@num>99\do{%
               H\advance\@attic@num-100}%
```

A numeral can have only one 𐅭 “digit” (equivalent to 50).

```
            \ifnum\@attic@num>49%
               ^^^^^^010144\advance\@attic@num-50%
            \fi\relax
```

The macro now checks now for tens digit.

```
            \@whilenum\@attic@num>9\do{%
               Δ\advance\@attic@num by-10}%
```

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

```
            \@whilenum\@attic@num>4\do{%
               Π\advance\@attic@num-5}%
            \ifcase\@attic@num\or I\or II\or III\or IIII\fi%
   \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` Here I redefine the macros `\@alph` and `\@Alph`. First, I define some placeholders

`\greek@Alph`

```
\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 numerals 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}
```

`\setlanguage` We provide the `\setlanguage` command which activates the hypehnation 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 LuaLaTeX users will actually use the `\HyphenRules` command.

```
\ifx\directlua\undefined%
  \def\setlanguage#1{%
     \expandafter\ifx\csname l@#1\endcsname\relax%
     \typeout{^^J Error: No hyphenation pattern for}%
     \typeout{ language #1 are loaded,}%
     \typeout{ default hyphenation patterns are used.^^J}%
     \language=0%
     \else\language=\csname l@#1\endcsname\fi}
\else
  \let\setlanguage\HyphenRules
\fi
```

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.

```
\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}}
⟨/xgreek⟩
```

3 The Source Code of xelistings

If the user has enabled the `listings` option, then the package loads the rudimentary package xelistings. This package loads the 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 `\lst@DefEC`.

```
⟨*xelistings⟩
\RequirePackage{listings}
\lstset{inputencoding=utf8}
\lst@InputCatcodes
\gdef\lst@DefEC{%
 \lst@CCECUse \lst@ProcessLetter
  ^^80^^81^^82^^83^^84^^85^^86^^87^^88^^89^^8a^^8b^^8c^^8d^^8e^^8f%
  ^^90^^91^^92^^93^^94^^95^^96^^97^^98^^99^^9a^^9b^^9c^^9d^^9e^^9f%
  ^^a0^^a1^^a2^^a3^^a4^^a5^^a6^^a7^^a8^^a9^^aa^^ab^^ac^^ad^^ae^^af%
  ^^b0^^b1^^b2^^b3^^b4^^b5^^b6^^b7^^b8^^b9^^ba^^bb^^bc^^bd^^be^^bf%
  ^^c0^^c1^^c2^^c3^^c4^^c5^^c6^^c7^^c8^^c9^^ca^^cb^^cc^^cd^^ce^^cf%
  ^^d0^^d1^^d2^^d3^^d4^^d5^^d6^^d7^^d8^^d9^^da^^db^^dc^^dd^^de^^df%
```

```
  ^^e0^^e1^^e2^^e3^^e4^^e5^^e6^^e7^^e8^^e9^^ea^^eb^^ec^^ed^^ee^^ef%
  ^^f0^^f1^^f2^^f3^^f4^^f5^^f6^^f7^^f8^^f9^^fa^^fb^^fc^^fd^^fe^^ff%
  ^^^^0386^^^^0388^^^^0389^^^^038a^^^^038c% <--- Begin of Greek Letters
  ^^^^038e^^^^038f^^^^0390^^^^0391^^^^0392%
  ^^^^0393^^^^0394^^^^0395^^^^0396^^^^0397%
  ^^^^0398^^^^0399^^^^039a^^^^039b^^^^039c%
  ^^^^039d^^^^039e^^^^039f^^^^03a0^^^^03a1%
  ^^^^03a3^^^^03a4^^^^03a5^^^^03a6^^^^03a7%
  ^^^^03a8^^^^03a9^^^^03aa^^^^03ab^^^^03ac%
  ^^^^03ad^^^^03ae^^^^03af^^^^03b0^^^^03b1%
  ^^^^03b2^^^^03b3^^^^03b4^^^^03b5^^^^03b6%
  ^^^^03b7^^^^03b8^^^^03b9^^^^03ba^^^^03bb%
  ^^^^03bc^^^^03bd^^^^03be^^^^03bf^^^^03c0%
  ^^^^03c1^^^^03c2^^^^03c3^^^^03c4^^^^03c5%
  ^^^^03c6^^^^03c7^^^^03c8^^^^03c9^^^^03ca%
  ^^^^03cb^^^^03cc^^^^03cd^^^^03ce% <--- End of Greek Letters
  ^^00}%
\lst@RestoreCatcodes
⟨/xelistings⟩
```