teubner.sty*
An extension to the greek option of the *babel* package

Claudio Beccari†

Turin, September 2015

Contents

1 Introduction 1
2 Usage 2
3 Package options 3
4 The CB Greek fonts 4
5 Font installation 9
6 TEX font metric files 9
7 Greek text and Latin keys 10
8 Ligatures 12
9 Upper case initials and capitalised text 16
10 Other Greek symbols 18
11 Milesian and Attic numerals 19
12 New commands 19
13 Metrics 20
14 Poetry environments 22

1 Introduction

This package teubner.sty is an extension of the greek option of the *babel* package intended to typeset classical Greek with a philological approach. This version 4.x cannot yet typeset the critical apparatus as the philologists are used to, but may be this work will continue and include also that facility. Apparently it is not incompatible with eledmac, but it may be considered a complement.

This document does not substitute the official package documentation teubner.dtx and its typeset version teubner.pdf; it extends the information contained into those files.

This package is supposed to work with my CB fonts available on the Comprehensive TEX Archive Network (CTAN); one of the actions of this package consists in adding to the default "italic" Greek shape another one called "Lispiakos" in

*This paper documents teubner.sty version v.4.7 of 2015/10/15.

†claudio dot beccari at gmail dot com

Greece; this name derives from the high quality of the fonts used in the printers' shops in the city of Lipsia in the past 100 years or so; one of the printer shops that continues printing books for philologists (since 1849) is the B.G. Teubner Verlagsgesellschaft, that publishes the collection called "Bibliotheca Scriptorum Graecorum et Latinorum Teubneriana". The name given to this extension package is in homage to that printing company and to its high quality tradition in printing Greek texts.

This package is generally loaded by default with any complete distribution of the TEX system; in basic distributions it might not be there, but in general it is sufficient to use the distribution facilities for loading it; with MiKTeX, for example one uses the MiKTeX settings and the Package tab. With TEXlive it is sufficient to operate with the program tlmgr; and so on.

The problem, if there is one, is to have the full collection of the CB fonts; in partial distributions the scalable PostScript CB fonts are in one size, 10pt, and it is necessary to use the *10pt* option in order to have the various enlarged or reduced sizes. For professional work, if the user has a partial TEX system installation, it is necessary to load the `cbgreek-full` font collection and be sure that the map files are duly upgraded. In case, read your distribution instructions to see how to control and, possibly, how to provide for this upgrade. Please, notice that this very file has been typeset with the *10pt* option in force, so that if you want to typeset it again from source, you need the minimum amount of vector fonts needed by pdflatex to do its job.

In any case this `teubner` extension package makes the CB font collection (full or reduced) directly compatible by construction with both the CM fonts (OT1 and T1 encoded; the latter ones are often referred to as the EC fonts) and with the LM fonts; it also contains adequate hooks in order to make them compatible and usable with other font collections; this very file has been successfully typeset using the CM and LM fonts, and also resorting to the Times and Palatino eXtended collections requested with the packages `txfonts`[1] and `pxfonts` respectively.

This additional documentation will start by briefly recalling some peculiarities of the CB fonts and their mapping to the Latin keyboard; afterwards it will list the new commands and their syntax.

2 Usage

The *teubner* extension is loaded in the usual way, but there are some simple rules to follow:

```
\usepackage[⟨options⟩]{teubner}
```

Since *teubner* is an extension of the *babel* package, it must be loaded after the latter. If you load it before and/or if you load it after, but you invoked the `babel` package without specifying the ⟨*greek*⟩ language and the `polutoniko`

[1] Antonis Tsolomitis wrote package txfontsb for using a different Greek font collection together with the TX fonts so that the Latin and Greek glyph styles directly match each other; see the txfontsb package documentation for more information.

Option	Meaning
or	obsolete option kept for backwards compatibility
10pt	for using only one real font size and geometrically enlarging or reducing the other sizes from the original single 10pt one
boldLipsian	to be selected in order to use a moderately blacker Lipsian font
NoGlyphNames	to be selected in order to avoid the definition of accented glyphs (default)
GlyphNames	to be selected in order to have available also the set of macros that directly address the accented glyphs

Table 1: Options for the *teubner* extension package

Greek language attribute, `teubner` refuses to be completely loaded and emits a message very clear on this subject, so that you know what you should do in order to use the facilities offered by its extensions.
Warning: Some people like the Lipsian font shape and want to load it also for writing mathematics with it. No problem, but they shouldn't do this with *teubner*, which is useful only to typeset Greek text. For using other alphabets in mathematics there are more suitable ways that rely on the commands described in the `fntguide.pdf` distributed with every TEX system in `$TEXMFDIST/doc/latex/base/`.

3 Package options

The options available to *teubner* are collected in table 1 with their meaning and usefulness.

You should never need (and therefore never use) the *or* option for two reasons: (*a*) glyph name macros are not needed for the first letter of any word since there are no kerning problems with a preceding letter; (*b*) glyph name macros are insensitive to uppercasing.
Warning: You might need to use the *10pt* option because your Greek font collection is the minimal one. But if you are using `teubner`, you are likely to seek the best possible typeset text; therefore you should download and install the full collection of the Greek CB fonts. This full collection is generally already available with any complete distribution of the TEX system.

If you are using Latin fonts different from the CM , EC, or LM collections, you have to control what you get with or without the option *boldLipsian*; generally speaking, this option is best used with darker normal fonts; I have tried the Times, the Palatino, and the Fourier ones, and actually this option is desirable. This might not be true with other vector fonts.

With the inclusion of the extended accent macros in the *teubner* package (see below), the named accented glyphs should not be needed anymore; these accented

glyph name macros are not defined by default, but for backwards compatibility they are available if the *GlyphNames* option is specified; with the 2013 new Greek support for the *babel* package, the extended accent macros are defined in a more efficient way and you should not need them any more; therefore, unless you specify this option, you avoid overloading the internal TEX memory areas, thus leaving extra space for more useful extensions. I suggest you to let the *default* option *NoGlyphNames* act as its name implies, and make use of the extended accent macros, should the need arise (see below). Moreover, if your keyboard has facilities for entering polytonic Greek, or if you installa a keyboard driver that allows you to do so, you can directly input Greek text with the Greek alphabet, without using the transliteration provided by the Greek support for *babel*.

4 The CB Greek fonts

The CB Greek fonts (full collection) come in all shapes, sizes and series as the European Modern fonts that conform with the T1 encoding introduced after the Cork Conference of the TEX Users Group Society in 1991[2]. The CB fonts conform to the encoding that is still being called LGR, since up to now there is no established encoding name for the Greek alphabet among the TEX users, not yet, at least.

The regular shape has capital letters with serifs that are in the same style as the Latin capital ones, while the lower case letters derive from the design by Didot and are very common in all texts. This shape comes also in boldface, together with the corresponding oblique (or slanted) versions. The CB fonts contain also the upright and slanted, medium and boldface small caps alphabets.

The "italic" shape was designed in order to imitate the Olga font designed so as to have a contrasting style compared with the slanted Didot shape, in order to play the same role as the italic letters play with the Latin roman ones. The Olga alphabets come in medium and boldface series, and in oblique and upright shapes.

The CB fonts are completed with the sans serif fonts, the monospaced typewriter fonts and the fonts for slides, besides an outline family that shows the regular shapes and series just with their contours; there is also a shape with serifed lower case letters.

The CB Lipsian fonts imitate the beautiful shapes used in Lipsia; they come in medium, bold, and extra-bold series, without an upright version, and they are meant to replace the corresponding Olga shapes. Their 'simple bold' series is good for mixing with PostScript fonts, whose medium series is slightly blacker than the corresponding EC and LM fonts nurmally used with LATEX; this is easily achieved by passing the *boldLipsian* option when invoking this *teubner* extension file. Notice that the Lipsian font produces an alternative to the ordinary 'italic' Olga shape, it is not any more the default 'italic' shape as it used to be with previous versions of this package; now, in the same document, you can use both shapes and produce both versions: *Βαχύλιδες* and *Βαχύλιδες*. With the availability of the unslanted Olga font and the serifed lowercase alphabet, you can use also the

[2]If you want to or you have to use the single 10pt size fonts, you certainly produce smaller PDF files, at the expense of a slightly poorer typographical quality.

\textui and the \textrs commands so as to obtain Βαχύλιδες and Βαχύλιδες; also in extended boldface: **Βαχύλιδες** and **Βαχύλιδες**.

But if you really want to permanently change the "italic" Greek font shape to the Lipsian one, without keeping the alternative, then in your preamble add the following statements *after* you have loaded the teubner package:

```
\addto\extrasgreek{\def\itdefault{li}}%
\addto\noextrasgreek{\def\itdefault{it}}%
```

With these settings, \textit and \textli become equivalent and both use the Lipsian shape while typesetting Greek text. Reverting to Latin script the \textli text command uses the italic shape by default and there is no need to add anything else to the \noextrasgreek macro.

Technical information

Typesetting documents with different scripts sets forth some problems. The *greek* option to the *babel* package, besides setting up the typographical rules for Greek, as it does for any other language, provides the script change; it defines also the \textlatin macro in order to typeset something with the Latin script while the default script is Greek, as well as the \textgreek macro to typeset something with the Greek script when typesetting with a Latin one. Of course the usual \selectlanguage and \foreinglanguage commands, as well as the otherlanguage environment, provide for a global change of the typesetting characteristics or an environment where the settings are reset to Greek.

With standard *babel* the CB fonts used to work only in conjunction with the CM fonts with either the OT1 (real CM fonts) or T1 (EC fonts) encoding. Since the 2008 distribution of standard *babel*, the CB fonts work fine also with the Latin Modern fonts. This *teubner* extension tries to work seamlessly also with other font families, but it is not that simple. Some technical explanations are necessary.

When LaTeX needs to use a specific font in a certain encoding and belonging to a particular family, series and shape, available in such and such sizes, it gets this information by reading a *font description file*; this file's name is composed with the encoding and the family names glued together and has the extension .fd; for OT1 encoded Computer Modern CM regular (serifed) fonts this file would be ot1cmr.fd. Any other package that is requested for using different fonts defines possibly a different encoding and certainly different family names. For using the Times eXtended TX fonts with T1 encoding, the txfonts.sty package defines the family name txr (for serifed fonts) so that LaTeX reads the font description file t1txr.fd that contains the relevant information for all the series, shapes, and sizes available.

The CB fonts are encoded according to the LGR encoding but have the same family names as the CM ones; since 2008, also the family names of the Latin Modern LM collection are recognised; therefore the relevant font description files for the regular[3] family are lgrcmr.fd and lgrlmr.fd respectively. The two

[3]The regular lower case Greek alphabet does not have serifs; serifs are present in the font shape called 'serifed' that corresponds to the\textrs font command.

collections of description files are not equivalent with one another, and they are not equivalent to the OT1 or T1 encoded CM or LM fonts, in the sense that the series and shapes available for these sets of fonts are not identical, even if most of them are.

In this way with CM and LM fonts the script switching for the same (existing) series and shapes amounts to switching the encoding name. At the same time there might exist some font switching commands that refer to a series or shape that does not exist in the other script families; one important example in our case is the Lipsian shape that is available only with the CB fonts. There is no problem in declaring the Lipsian shape switching commands that behave in a proper way together with the CM and LM fonts, but even if I did my best for working with other font families, I am not 100% confident that my macros restore correctly the other font characteristics when declaring a different series or shape.

Going more technical, the default family settings are stored into the three macros `\rmdefault`, `\sfdefault`, `\ttdefault`; the series and the shape symbols are stored in similar macros; these default macros are accessed every time a font characteristic switching declaration or command is used. In the background the `\selectfont` macro is executed and during the whole process the *current* font characteristic macros are updated. Such macros are `\f@encoding`, `\f@family`, `\f@series`, and `\f@shape`; therefore when just one characteristic is modified, the new value is stored in the relevant current value macro and used in order to create the association with the actual font to be used.

If the font switching macros are used within a group, upon exiting that group the previous values are restored; but if declarations are used without a delimiting scope, there is no simple (universal) way to use another ready-made declaration that resets valid font characteristics.

An example where the necessity of delimiting scopes is shown: suppose we are typesetting with T1 encoded LM roman medium normal (upright) fonts; then the current font characteristics codes are stored in the relevant macros as `T1`, `lmr`, `m`, `n`. We switch to Greek with a language setting declaration, and the current encoding is changed to `LGR`, but the other characteristics remain the same, therefore we would be typesetting with LGR encoded, CB Didot (upright) medium normal font. While typesetting in Greek we switch to Lipsian shape with a declaration such as `\lishape`[4], and this declaration changes only the current shape characteristic, so that the four ones would be `LGR`, `lmr`, `m`, `li`. Up to this point everything runs smoothly because every characteristic that was set is present in the specific font family in use. At this point we revert to typesetting with the Latin script by means of a language switching declaration; the only change that takes place is on the encoding and the four characteristics would become `T1`, `lmr`, `m`, `li`, but... The Latin Modern fonts do not contain a Lipsian shape, therefore there is no actual font that meets the requirements and LaTeX selects the *error font*, the T1 encoded LM

[4]Notice that the `\lishape` declaration and the corresponding text command `\textli` are defined in such a way that they switch to the Lipsian shape only when the Greek encoding LGR is in force; with other encodings they behave as the corresponding italic font commands. In spite of this the example being carried on is valid, because the `\lishape` *declaration* is supposed to be issued while the Greek encoding was in force.

roman medium normal one, that in this case is the correct one, and apparently the font switching process achieved the correct result. This is only apparent: remember that the *error font* was selected, not the right font. If the Latin family and the Greek family hadn't been the same, there would have been other difficulties and the error font would have been selected; this is why, if the user does not pay attention, when the default font should be, say, the TX font, after a switch to Greek, upon reverting to the Latin script the wrong font series or shape might be used. In the next paragraphs some indications are given in order to overcome this feature.

All these technicalities are really too technical, but it's necessary to have some clues in order to find out why sometimes the font switching commands don't work as expected. In some cases the font description file might provide a smart substitution for missing fonts, but it is not always the case.

In this package I tried to forecast most situations, but I am not sure I coped with every font characteristic combination.

In particular I hooked the language changing declarations with suitable default family names; for example, when using the Times or the Palatino eXtended TX or PX fonts, three new family description files are created so as to connect the LGR encoding and the names of the above scalable fonts to the corresponding CB fonts[5].

If you use different scalable fonts you can specify yourself the font associations you want to use; simply, after loading a package that sets other default font family or families, open the the package file and take notice of the new family names; for each ⟨*latin*⟩ family (serifed, sans serif, monospace) create a connection with the corresponding ⟨*greek*⟩ family by means of the following command:

`\ifFamily{`⟨*latin*⟩`}{`⟨*greek*⟩`}`

right after the `\usepackage` command with which you call that package.

Example: suppose you want to use the regular Fourier fonts to replace the default roman normal serifed fonts; then you should load the package `fourier` *after* the `teubner` extension and specify:

```
\usepackage{fourier}
\Lipsiantrue
\ifFamily{futs}{cmr}
```

You can see that the family name `futs`, corresponding to the regular Fourier font family, has been matched to the CB Greek `cmr` font family; the name `futs` has been deduced by reading the `fourier.sty` file from which one can see that the family declaration for the regular Fourier font family is `futs`. This sort of coding does not take place with all fonts: if the Iwona fonts had to be chosen, for example, then the font family name would coincide with the font name, making it difficult to distinguish between the *font name* and the *font family name*.

[5] I chose the CM Greek families, instead of the LM ones, because only the former are described by means of macros that cope with the *10pt* option to the `teubner` package; just in case… The actual used fonts are the same in any case, except possibly for the visual sizes; the CM fonts come in fixed sizes, while the LM fonts are continuously scalable by enlarging or reducing a smaller number of base visual sizes.

Notice the \Lipsiantrue command before issuing the matching command \ifFamily}. This command is optional and is used only if the composer wishes to use the Lipsian fonts; in this case the li shape is also defined, so that the \textli and \lishape commands do not issue any error message. At the end of the execution if the \ifFamily command, the boolean switch is automatically reset to \Lipsianfalse.

Warning: According to several maintainers of the language support packages, the procedure used with \ifFamily, that produces in the local working directory a real, permanent .fd file, is not the best approach to create such pairing of fonts. Günter Milde, for example, produce a *substitutefontfamily* package that defines the command \subtitutefontfamily which creates a similar association, but does not write anything to the disk; simply the created association declarations are stored in the job memory, so that when the typesetting job is over, no traces remain on disk; this certainly avoids clogging the disk with the same files scattered all over in different directories that were the working one for different typesetting jobs.

A small caveat: when you issue for the first time the command \ifFamily you might not see the expected result (while with Milde's package you immediately obtain the expected result), and your Greek text might be typeset with the default "Greek error font". But the second time you typeset your document the expected result is obtained with the correct fonts. This is because with the very first run a new font description file is generated, and this file will be available only in subsequent typesetting runs.

Nevertheless this does not imply that the correct fonts are used if the font switching macros are used without scoping groups or environments. Sometimes, when you use declarations instead of commands, it might be necessary to issue an apparently redundant \rmfamily or \selectfont command in order to re-establish the correct defaults.

In order to insert *short texts* in Greek, either in Didot upright or in Lipsian inclined shape, the *text* commands \textDidot and \textLipsias may be used, as well as a redefined \textlatin *text* command for typesetting a short Latin script text while typesetting in Greek; these macros should already select the correct encoding, family, series and shape in most circumstances.

The *text commands*, contrary to the corresponding *text declarations*, typeset their argument within a group, so that the font characteristics are also correctly restored after the command execution is completed.

Therefore I suggest you to either use the text commands or to use the declarations as environment names (without the initial backslash), so that they provide the necessary group delimiters; it's syntactically correct and useful to input something such as:

```
\begin{Lipsiakostext}
⟨Greek text to be typeset with the Lipsian font⟩
\end{Lipsiakostext}
```

5 Font installation

In order to use the Greek CB fonts and the extensions provided with this package, you need to install them, if they are not already installed by default when you install your preferred TEX system *complete* distribution. You can freely download those fonts from CTAN, where you can find both the PostScript scalable ones and the driver files for generating their bitmaped with METAFONT; since nowadays it's very unlikely that LATEX users limit themselves to a final DVI file, but typeset their documents in PDF (or PS formats), the pixel files are very unlikely needed to produce their final documents; the METAFONT pixel files just allow the DVI previewer (should one be used) to run METAFONT in the background in order to produce the necessary pixel files so as to display on the screen the typeset documents[6]. These days there are other solutions to preview directly the typeset output file in PDF format and to use the previewer for direct and inverse search, so that the DVI format becomes really necessary only in very special and rare occasions. In the future it's very likely that the choice among such PDF previewers is much greater than today: there are certainly the multiplatform shell editors TeXworks, TeXstudio, and Texmaker that incorporate a PDF previewer capable of direct and inverse search; for modern Windows platforms there is the SumatraPDF previewer that may be configured to work with various shell editors in order to perform in the same way. For Mac platforms, besides the default shell editor and previewer TeXShop, and the multi-platform program TeXworks, the Mac specific TeXnicle and Texpad, one can proceed with Aquamacs as a shell editor and Skim as PDF previewer that work together so as to allow direct and inverse search.

6 TEX font metric files

The package `teubner` requires the updated TEX font metric files `.tfm` for all sizes and series of the Lipsian fonts. With your file system explorer go to the folder `$TEXMFDIST/fonts/tfm/public/cbfonts/` and read the date of, say, `grml1000.tfm`; if this date precedes the year 2010, then this `.tfm` file is "old" and must be replaced, and you should update your Greek CB font collection (may be you should update your whole TEX system installation).

The updated `.tfm` files add some ligature and kerning information that is missing from the "old" ones. The Type 1 `pfb` font files have not been modified at all.

[6] I mean documents that have been typeset with latex, not with pdflatex.

α	β	γ	δ	ε	ζ	η	ϑ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω
a	b	g	d	e	z	h	j	i	k	l	m	n	x	o	p	r	c	s	t	u	f	q	y	w

Table 2: Keyboard correspondence between Latin and Greek letters

7 Greek text and Latin keys

In order to input Greek text with a Latin keyboard[7] some simple and mostly obvious key substitutions are performed according to the correspondence shown in table 2.

Notice that there is the possibility of inputting `c` in order to get the final sigma ς, but the CB fonts are conceived with the non-Greek typist in mind, so that it is even possible to input `s` at the end of words, because the whole software is smart enough to detect the word boundary and to use the correct shape of the letter sigma within or at the word end. This mechanism is so "sticky" that it becomes difficult to type an isolated initial or middle sigma; the CB fonts contain an invisible character, `v`, that may be used for several purposes, one of which is to hide the word boundary after a sigma; therefore if you type `sv`, you get σ without any effort.

The invisible character `v` may be used also as a support for (apparently) isolated accents, especially when macros have to be used; if you type `\M{v}` you get ¯, while if you omit the invisible `v` you get – .

In LICR (LaTeX Internal Character Representation), that is being used by the modern (2013) Greek support for *babel*, the `v` character is represented by the macro `\textcompwordmark` whose name recalls a compound word separator; actually it may perform also in this way; but it is a special character, not simply an invisible strut; it has the same category code as the other alphabetic characters, it has a very small width but its height is equal to the other lower case letters without ascenders; it's a real character even if in practice it is invisible; in particular it does not impeach hyphenation as an equally sized normal strut would do. But its usefulness is (*a*) to hide the word boundary; (*b*) to support an accent that would drop to the base line if this character was absent; (*c*) to break ligatures in those rare cases where it's necessary to do so.

Accents, spirits and dieresis may be input *before* each letter (prefix notation) without using any particular control sequence, that is resorting to the ligature mechanism that is part of the font characteristics; but since with certain fonts this mechanism may break kernings, it's better to use the proper accent macros; the correspondence between the Latin symbols and the Greek diacritical marks is shown in table 3; all "upper" diacritical marks must be prefixed (in any order), while the iota subscript must be postfixed. Therefore if you input `>'a|`, you get ᾀ.

Macrons and breves are just single glyphs and do not appear in combination

[7] Although tables 2 and 3 display only Latin ASCII characters in the second line, there are some national keyboards where some of these ASCII symbols can't be typeset by striking a single, possibly shifted, keyboard key or a simple key combination; an example is the Italian keyboard, where both the 'back tick' and the 'tilde' can be input only in one of these two ways: (*a*) open the Character Map accessory [some shell editors can open their internal character map, where to select the desired characters] and select the required glyph(s), or (*b*) while pressing the `Alt` key, input the numerical glyph code from the numeric pad. Very uncomfortable!

Greek diacritics	̓	̔	̈	́	̀	̃	̣
Latin keys	>	<	"	’	‘	~	|
extended accent macros	\>	\<	\"	\’	\‘	\~	

Table 3: Correspondence between the Latin keyboard symbols and Greek diacritical marks; extended accent macros are also shown; notice that a couple of "high" diacritical marks may be joined in one macro (in any order) to produce the same result as with two separate macros; in other words, \>\', \'\>, \>' and \'> are equivalent with one another. For what concerns \~, see remarks below table 4

with any letter, due to the limitation of 256 glyphs per font; but they may be input by means of commands `\M` and `\B` respectively when typesetting in Greek in order to use them as single diacritics; for more than one diacritic superimposed to one another, when macrons and breves are involved, some other commands are available as shown in table 4.

The new (2013) Greek support for *babel* contains now the LICR encoded accent macros that allow to use any encoding input method, including direct input of Greek characters. The internal representation of accent macros resorts to the same symbols shown in table 3; simply they are prefixed by a backslash and/or the accent sequence is prefixed with one backslash; therefore ᾷ may be obtained with `>'a|`, or `\>\'a|`, or `\>'a|`. The last form is to be preferred with some shapes as discussed in the following paragraphs. But the LICR (LaTeX Internal Character Representation) is much more than this; it separates the input encoding from the output font encoding, so that direct literal greek input is possible (of course with the [⟨*utf8*⟩] encoding) and the output is useful for both LGR encoded 8-bit fonts and the UNICODE encoded OpentType fonts; this way *babel* and the Greek support for *babel*, provided by the new `greek.ldf`, may be used also with XeLaTeX and LuaLaTeX. Moreover now (October 2014) file `greek.ldf` can select the ancient Greek hyphenation when typesetting with pdfLaTeX[8]; with *polyglossia* this possibility was available from the very beginning of the Greek support by this package.

Together with the macros for inserting such symbols, a complete set is available for inserting any combination of diacritical marks over or under any letter, not only vowels: see table 4.

Of course the results may not be comparable with the ones one can obtain with the regular ligature mechanism or by using the LICR macros; the advantage of the redefined accent macros is twofold: (a) it is connected to the possibility of inserting macrons and breves and/or to set the various combinations over or under *any* letter, even if it is a consonant; (b) for all accent vowel combinations that have a specific glyph in the font, the actual accented symbol is used so that kernings and ligatures are maintained; this result is achieved also by using the extended accent macros shown in the third line of table 3; as shown elsewhere there is a noticeable difference between *αὐτός* and *αὐτός* or *αὐτός* or *αὐτός*. In

[8]This is why with this release of the *teubner* package, the default language attribute is set to `ancient`.

this example the first word is typed in as `a>ut'os`, while the other words may be typed in as `a\s{u}t\'os`, or `a\us␣t\oa␣s`, or `a\>ut'os`, or even in mixed form `a\us␣t'os`, thanks to the fact that there is no kerning between 'tau' and 'omicron with or without acute'.

The attentive reader might have noticed that some accent macros use the control symbols that are available for other standard LaTeX environments or even defined in the LaTeX kernel. Accent macros `\‘`, `\'` and `\=` are already taken care by the LaTeX kernel that uses a particular trick to use them as control macros in the *tabbing* environment; therefore there should not be any conflict.

On the opposite some conflicts may arise with `\<` and `\>`, as reported by David Kastrup. We have examined these conflict occurrences and we noticed that actually `\>` is defined as a 'math wide space' command; an alias `\:` is defined immediately after the definition, and the LaTeX handbook by Leslie Lamport does not even mention `\>`; therefore `\>` may be considered a leftover from the beginnings of LaTeX, and it would be surprising if anybody used it; nevertheless... The control symbol `\<` has been reported as being used as a macro with a delimited argument `\<`⟨*arg*⟩`>` in the documentation of package *calc*. We immagine that it is very unlikely that this *teubner* package might be used in another package documentation, but...

These conflicts are avoided if Günter Milde package *textalpha* is loaded; we did not find out any conflict in any testing run we made, but such conflicts may arise; therefore we decided to have *teubner* load the *textalpha* package at `\begin{document}` execution time, so as to avoid any possible conflicts.

8 Ligatures

It should be clear from the previous section that the ligature mechanism is the one that offers good results with most accented vowels, while speeding up the keying-in of the text to be typeset; nevertheless there are situations where you might be unsatisfied.

For example compare *αὐτός* with *αὐτός*. The small spacing difference, if any, between tau and the accented omicron is not noticeable, but the spacing difference between alpha and the marked upsilon is remarkable. Where does that difference come from? It comes from the fact that the smooth spirit marker inhibits kerning between the previous alpha and the resulting ligature from the spirit marker and the upsilon[9]. In other words, by inputting `a>ut'os`, as it is suggested in the previous section, the spirit marker and the acute accent inhibit the kerning mechanism with the previous letter. In most instances the lack of such kerning is hardly noticeable, but in others it strikes your attention.

For this reason two mechanisms are implemented in this package:

[9]The CB fonts may be used also with monotoniko spelling; in this case *ΑΫΛΟΣ* comes out well even with ligatures `A"ULOS`, because a special kludge has been devised; with the extended accent macros it would not be too boring to typeset `A\"ULOS` and do away with that kludge; without it and without such extended macros that word would be set as *AΫΛΟΣ* with a much larger space between the capital alpha and the marked capital upsilon.

Example	Syntax	Example	Syntax
ὰ	\G{⟨*letter*⟩}[1]	α͡ω	\ut{⟨*letters*⟩}
ά	\A{⟨*letter*⟩}[1]	ᾴ	\Ab{⟨*letter*⟩}
ᾶ	\C{⟨*letter*⟩}[1]	ᾲ	\Gb{⟨*letter*⟩}
ῒ	\D{⟨*letter*⟩}[1]	ᾷ	\Arb{⟨*letter*⟩}
ᾰ	\B{⟨*letter*⟩}[4]	ᾳ	\Grb{⟨*letter*⟩}
ᾰͅι	\U{⟨*diphthong*⟩}	ᾷ	\Asb{⟨*letter*⟩}
ᾱ	\M{⟨*letter*⟩}[4]	ᾳ	\Gsb{⟨*letter*⟩}
ȧ	\>{⟨*letter*⟩}[3]	ᾴ	\Am{⟨*letter*⟩}
ȧ	\<{⟨*letter*⟩}[3]	ᾲ	\Gm{⟨*letter*⟩}
ȧ	\r{⟨*letter*⟩}	ᾷ	\Cm{⟨*letter*⟩}
ȧ	\s{⟨*letter*⟩}	ᾷ	\Arm{⟨*letter*⟩}
ἵ	\Ad{⟨*letter*⟩}	ᾳ	\Grm{⟨*letter*⟩}
ἴ	\Gd{⟨*letter*⟩}	ᾷ	\Crm{⟨*letter*⟩}
ἷ	\Cd{⟨*letter*⟩}	ᾷ	\Asm{⟨*letter*⟩}
ἅ	\Ar{⟨*letter*⟩}	ᾳ	\Gsm{⟨*letter*⟩}
ἃ	\Gr{⟨*letter*⟩}	ᾷ	\Csm{⟨*letter*⟩}
ἇ	\Cr{⟨*letter*⟩}	ᾳ̣	\Sm{⟨*letter*⟩}
ἅ	\As{⟨*letter*⟩}	ᾳ̊	\Rm{⟨*letter*⟩}
ἓ	\Sb{⟨*letter*⟩}	ᾰͅ	\Rb{⟨*letter*⟩}
ἃ	\Gs{⟨*letter*⟩}	ᾳ̇	\iS{⟨*letter*⟩}
ἇ	\Cs{⟨*letter*⟩}	π̇	\d{⟨*letter*⟩}
᾽	\c{⟨*letter*⟩}	ῐ̇	\bd{⟨*letter*⟩}
u̯	\semiv{⟨*letter*⟩}[2]	ḁ	\ring{⟨*letter*⟩}[2]

Table 4: Accent macros

REMARKS

[1]These four commands superimpose the corresponding accent over the ⟨*letter*⟩ belonging to the Greek alphabet; the corresponding commands \‘, \’, \~, and \" play the same rôle also with non Greek letters; this allows the stacking of several accents on the same base ⟨*letter*⟩.

[2]Most commands may be used also with latin letters.

[3]While typesetting in Greek within a `tabbing` environment, both commands \> and \< play a different rôle as tab shifters; in this case either \s and \r are used for the spirit macros, or the standard ligature mechanism is used, especially if the spirit diacritics fall on the very first vowel of a word. Suggestion; never use the `tabbing` environment; there are more effective ways to align chunks of text, for example the `tabular` environment.

[4]In *teubner* versions preceding 4.0 macros for typesetting macrons and breves in Greek had the same name as the corresponding macros used when typesetting with Latin script, that is \= and \u. With the new Greek support for the *babel* package that was implemented during year 2013, the Latin macros were not functional any more in the proper way, and were updated in *teubner* version 4.0 with new names.

1. the extended accent macros created by Günter Milde are now the default setting in the Greek support for *babel*, and
2. the character names have been defined with macros that access directly their

own glyphs.

The first solution has been described in the previous section and the extended accent macros are shown in table 3. At the expense of one slash, these macros create a chain of commands that deeply exploit the LaTeX 2_ε kernel commands and allow to fetch directly each accented character; notice that there is no need to treat the | sign this way, because its postfixed position does not break the kerning/ligature mechanism. The spirit macros \> and \< work also with the consonant 'r' to produce ῤ, ῥ. The extended accent macros work correctly also with capital letters (where, in all caps, accents should not be typeset, except the dieresis, while in normal initial capitalising the upper diacritics must be written at the left of the initial letter); for example[10] Ἄυλος, ἍΫΛΟΣ was typeset with `\>'Aulos, A\"ULOS`. Notice also that the initial capital vowel has a spirit and possibly an accent, or is not preceded by anything, since the possible spirit with or without accent falls on the second element of a descending diphthong: αὐτός, Αὐτός, and ΑΥΤΟΣ; εἶναι, Εἶναι, EINAI. If you really want to show an example of how accents should *not* be used in all caps words, you can type ΑΫ́ΤΌΣ but you have to use the macros of table 4 as such: `A\>UT\A{O}S`.

Another solution is available if the *GlyphNames* option is specified to the `teubner` package: a set of macros has been defined such that it is possible to input the accented characters directly, without resorting to the ligature mechanism. Such macros have a common structure; they are formed with the letters that make up the complex glyph in a certain order, precisely every macro is made up as such:

1. the first character, obviously, is the backslash character \;
2. the next character is the name of the base character, one of the vowels a, e, h, i, o, u, w, or the consonant r, or one of the capitalised vowels I or U;
3. the next optional character is the code for dieresis, or smooth, or rough spirit, with one of the letters d, s, r;
4. the next character is the code for the circumflex, acute, or grave accent with one of the letters c, a, or g;
5. the last optional character indicates iota subscript with the presence of an i;
6. there are no glyph names for upper case letters, since they should never be marked with diacritics, except for the diaeresis over I and Υ and for these glyphs adequate names are provided.

This means that, for instance, `\asai` stands for ᾷ. For your convenience such macros are collected in table 5. Of course one can always resort to the accent–vowel combination as exemplified at the end of the previous section; the above example ᾷ may be obtained also with `\>'a|` or `\As{a}|`.[11]

[10] In the capitalised word the spirit and accent at the left of A imply a hiatus between the A and the υ; in the all caps word this hiatus is marked with the "dialytika" (dieresis) over the Υ. This is an unusual example, but it shows also why the `\uppercase` and the `\MakeUppercase` require some attention in polytonic Greek.

[11] Postfixed markings do not pose any problem with kernings and ligatures; this is why the postfixed ligature for the iota subscript may still be used also when the accent–vowel combinations are used.

\aa	á	\ag	ȧ	\ac	ã	\ai	ą	\ar	ả	\as	ạ
\asa	ẩ	\asg	ẳ	\asc	ẵ	\asi	ậ	\aai	ặ		
\ara	ằ	\arg	ȃ	\arc	ẫ	\ari	ḁ	\agi	ạ	\aci	ą̃
\arai	ą́	\argi	ą̉	\arci	ą̃	\asai	ą̂	\asgi	ą̂	\asci	ą̃
\ha	ή	\hg	ή	\hc	ῆ	\hi	ῃ	\hr	ἡ	\hs	ἣ
\hsa	ἧ	\hsg	ἣ	\hsc	ἧ	\hsi	ᾗ	\hai	ᾖ		
\hra	ἦ	\hrg	ἣ	\hrc	ἧ	\hri	ᾐ	\hgi	ᾐ	\hci	ᾗ
\hrai	ᾗ	\hrgi	ᾗ	\hrci	ᾗ	\hsai	ᾗ	\hsgi	ᾗ	\hsci	ᾗ
\wa	ώ	\wg	ώ	\wc	ῶ	\wi	ῳ	\wr	ὡ	\ws	ὠ
\wsa	ὧ	\wsg	ὧ	\wsc	ὧ	\wsi	ᾧ	\wai	ᾧ		
\wra	ὦ	\wrg	ὣ	\wrc	ὧ	\wri	ᾠ	\wgi	ᾠ	\wci	ᾧ
\wrai	ᾧ	\wrgi	ᾧ	\wrci	ᾧ	\wsai	ᾧ	\wsgi	ᾧ	\wsci	ᾧ
\ia	ί	\ig	ί	\ic	ῖ	\ir	ἱ	\is	ἱ		
\isa	ἷ	\isg	ἷ	\isc	ἷ	\ida	ΐ	\idg	ΐ		
\ira	ἷ	\irg	ἷ	\irc	ἷ	\idc	ῗ	\id	ϊ	\Id	Ϊ
\ua	ύ	\ug	ύ	\uc	ῦ	\ur	ὑ	\us	ὑ		
\usa	ὗ	\usg	ὗ	\usc	ὗ	\uda	ΰ	\udg	ΰ		
\ura	ὗ	\urg	ὗ	\urc	ὗ	\udc	ῧ	\ud	ϋ	\Ud	Ϋ
\ea	έ	\eg	έ	\er	ἑ	\es	ἑ				
\esa	ἕ	\esg	ἕ	\era	ἕ	\erg	ἕ				
\oa	ό	\‘o[1,3]	ό	\<o[2,3]	ὁ	\os	ὁ				
\osa	ὅ	\osg	ὅ	\ora	ὅ	\org	ὅ				
\rs	ῥ	\rr	ῥ								

Table 5: Glyph name macros

REMARKS

[1]Some users remarked that with previous *teubner* versions \og macro conflicted with the homonymous command available when typesetting French texts with the `french` option to *babel* in force; for this reason it's necessary to use \‘o in its place.

[2]As mentioned in the body of this text the command \or is incompatible with the primitive TEX command with the same name, therefore it is necessary to use \<o in its place.

[3]In both cases macros are unnecessary, specifically for \<o, since an initial accented letter has no preceding letter whith whom some kerning might be necessary; therefore at the start of a word use 'o or <o, and use \'o in a word internal position, should some kerning be necessary; actually the shape of the omicron seldom requires any kerning.

What I suggest is to typeset your paper with the regular accent vowel ligatures and to substitute them in the final revision with the extended accent macros or the glyph name macros only in those instances where the lack of kerning is disturbing. The extended accent macros should set forth less conflicts with other packages and should be the first choice when cleaning up the final revision. The glyph names will be kept in future releases of this package just for backwards compatibility. Up to now these glyph name macros are disabled by default but, if the option *GlyphNames* is specified in the call statement to the `teubner.sty` package, they are turned on and become available.

9 Upper case initials and capitalised text

As we have seen in the previous sections the ligature and kerning mechanisms are strictly connected and the extended accent macros may be useful in solving certain situations. The suggestion at the end of the previous section holds true, but some remarks must be underlined in order to use at its best the coexistence of both methods for using the accented glyphs.

The remarks concern the use of capital letters and, to a certain extent also the caps and small caps font shape. As mentioned before in Greek typography words with capital initials starting with a vowel require their diacritics in front and at the left of the initial capital vowel (for example: *Ἀριστοτέλης*) provided the vowel is not the first element of a Greek diphthong; in the latter case the diacritics go on top of the diphthong lower case second element (for example: *Αἰσχύλος*). On the opposite, all-caps words, such as in book titles or, with certain LaTeX classes, in the headings, are typeset without any diacritic, except the diaeresis.

I have already shown the unusual example of the word *ἄϋλος* where the first two vowels do not form a diphthong, but a hiatus; and this is why the diacritics fall on top of the alpha, not on the second element of the diphthong, for the very reason that there is no diphthong. In order to stress this unusual situation the word is most often spelled as *ἄϋλος*, where the dieresis is sort of redundant, because the diacritics over the alpha already denote the hiatus. Similarly *ἀϋπνία* is spelled with the redundant dieresis, although the soft spirit on the alpha already marks the hiatus. When these words require a capital initial, they become *Ἄϋλος* and *Ἀϋπνία*.

Notice that an initial single vowel may receive only a spirit marker with or without an accompanying accent, never a dieresis or a single accent[12]. I tried hard to set up all the ligature and extended accent macros so as to do the right things when capitalising or uppercasing, and if you follow the suggestions given in this section you should not meet any particular inconvenience.

Assume these words have to go in an all-caps header that is made up (behind the scenes) by making use of the `\MakeUppercase` command; they have to be spelled *ΑΫΛΟΣ* and *ΑΫΠΙΝΙΑ*, even if in the sectioning command argument they were spelled in lowercase with all the necessary diacritics.

This is where the ligature and/or the extended accent macros may show their different behaviour. With the `teubner` generated secondary `LGRaccents-glyphs.def` file, that contains all the extended accent macros, such macros do not disappear in uppercasing, i.e. in transforming the mixed case argument of the `\MakeUppercase` command into an all-caps letter string. On the opposite the ligature sequences completely loose any reference to diacritics.

In other words we have:

1) `\MakeUppercase{>'aulos}` yields *ΑΥΛΟΣ*;
2) `\MakeUppercase{>'a"ulos}` yields *ΑΫΛΟΣ*;
3) `\MakeUppercase{>'a\"ulos}` yields *ΑΫΛΟΣ*;

[12] Of course we are talking of the polytonic spelling, since this package `teubner` sets this spelling as the default one; nobody forbids to spell in monotonic even when the polytonic spelling is assumed, but in some rare instances there might be some inconsistencies.

4) `\MakeUppercase{\>'a\"ulos}` yields $A\ddot{Y}\Lambda O\Sigma$;
5) `\MakeUppercase{\>\'a"ulos}` yields $A\ddot{Y}\Lambda O\Sigma$;
6) `\MakeUppercase{\asa\ud␣los}` yields $\check{a}\ddot{\imath}\Lambda O\Sigma$
7) `\MakeUppercase{e>uzw"'ia}` yields $EYZ\Omega\acute{I}A$;
8) `\MakeUppercase{e>uzw'"ia}` yields $EYZ\Omega\ddot{I}A$;
9) `\MakeUppercase{e>uzw\"\'ia}` yields $EYZ\Omega\ddot{I}A$;
10) `\MakeUppercase{e>uzw\"ia}` yields $EYZ\Omega\ddot{I}A$.

and it's easily seen that: 1) corresponds to a non redundant lowercase correct spelling but misses the required dieresis in upper case; 2) is correct but it relies on a special kludge on which it's better not to rely on, because in future releases of the fonts it may be eliminated; meanwhile it is usable; 3) is correct; 4) is correct because `\MakeUppercase` does not act on macros, but `\>'` *is* a macro, precisely an accent macro, that acts on the following letter before uppercasing, and the uppercase of an alpha with diacritics si simply the capital alpha; 5) is corret, but relies in the same kludge as case 2, so the same warnings apply; case 6) displays the result of using the accented glyph names which are not subject to capitalisation since they are given by means of macros; therefore they can never be used for words that are possibly subject to all-caps transformation.

Let's examine another case where the lower case word has both the dieresis and an accent over the same vowel: *εὐζωΐα*. Case 7) shows the effect of uppercasing when using ligatures, and the result is not correct because the dieresis does not fall over the capital iota, but between the capital omega end th capital iota; 8) by simply inverting the sequence of the dieresis and the acute accent a correct result is obtained, but kerning problems might take place because of the invisible presence of the lower cased acute accent; notice that the same result would be obtained if instead of the dieresis-vowel ligature the extended dieresis macro had been used; case 9) displays the situation when both extended accent macros are used so that the result does not suffer of any kerning problem; finally, case 10) displays a correct upper case result but the lower case counterpart would miss the acute accent.

When using small caps or caps and small caps, in other words, when using the `\scshape` declaration or the `\textsc` command you should pay attention to other details; for example:

11) `\textDidot{\scshape >Arqim'hdhs}` yields ΑΡΧΙΜΗΔΗΣ
12) `\textDidot{\scshape \>Arqim'hdhs}` yields ΑΡΧΙΜΗΔΗΣ
13) `\textDidot{\scshape \>Arqim\'hdhs}` yields ΑΡΧΙΜΗΔΗΣ
14) `\textDidot{\scshape >Arqim\'hdhs}` yields ΑΡΧΙΜΗΔΗΣ
15) `\textDidot{\scshape \>Arqim\ha dhs}` yields ΑΡΧΙΜΗΔΗΣ
16) `\textDidot{\scshape \>{v}Arqim\'hdhs}` yields ᾼΡΧΙΜΗΔΗΣ

With caps and small caps a spirit, or a spirit-accent combination prefixed to a capital letter does not produce any mark; this is the usual modern Greek habit of avoiding accents with capital letters and the `\scshape` specification to Greek fonts excludes all the ligatures and kernings with such signs; the same holds true also with small caps. The extended accent macros explicitly avoid any mark over capital letters, but even in front they are "silenced" by the very characteristics

”	’	((	«	))	»
\GEodq	„	\GEcdq	“	:	:
\GEoq	,	\GEcq	‘	?	;
\ENodq	“	\ENcdq	”	;	·
\stigma	ϛ	\varstigma	ϛ	\Stigma	Ϛ
\coppa	ϙ	\koppa	ϟ	\Koppa	Ϙ
\sampi	ϡ	\Sampi	ϡ	\permill	‰
\textdigamma	ϝ	\Digamma	Ϝ	\euro	€
\f	ϝ	\F	Ϝ	\shwa	ə

Table 6: Greek and other symbols

of the specific font shape. With small caps there is a similar situation, and the extended accent macros produce the expected result. But accents are apparently used in French also when proper names are typeset in caps and small caps, contrary to Greek, so that in French the spelling of case 16 is correct; for the initial spirit, therefore it is necessary to resort to the invisible character obtained with the Latin letter ‘v’, that is uppercased to itself and is suitable for supporting any diacritic mark at the proper height. Julien Browaeys pointed out this specific typesetting French tradition; I thank him very much for his feedback on this point.

Conclusion: when writing the input code for a sectioning command, the argument of which has to be transformed to upper case, use \" (in the proper sequence with other accents) for the internal diereses, as in cases 3), 4), 8), and 9) above. With the extended accent macros there are really few problems; the remaining ones take place in complete uppercasing remain with French traditions in caps and small caps that can be easily handled through the use of the v special invisible glyph. If you want to avoid also this possibility, then either avoid classes that typeset their headings in all-caps, or use any external package that defines sectioning commands with two optional arguments, one for the heading contents and the other for the table of contents; or use equivalent tricks: for example the `memoir` class has a specific command `\nouppercaseheads` that eliminates any heading uppercasing.

10 Other Greek symbols

Other Greek symbols may be obtained with ligatures or explicit commands; table 6 contains such ligatures and symbols; notice that some of these are specific additions introduced with this extension package.

I draw your attention on the necessity of using the ligature ” for producing the simple apostrophe, which, by the way, in Greek typography must always be followed by a space. The single tick mark ’ produces an acute accent, not an apostrophe, this is why it is necessary to use the double tick mark ligature. Actually also a double quote mark " followed by a space produces an apostrophe followed by a space.

11 Milesian and Attic numerals

The Milesian numerals should not worry anybody, because they are seldom used as isolated symbols; the greek or polutonikogreek language option or the polutoniko attribute of the Greek language with the *babel* package offer the commands `\greeknumeral` and `\Greeknumeral`, that convert common arabic positive numbers in the Milesian counterparts within a Greek section of your document; the corresponding commands followed by an asterisk change the digamma glyph with the stigma one[13]:

if you type `\greeknumeral{1996}` you get ͵αϡϙϝʹ

if you type `\Greeknumeral{1996}` you get ͵ΑϠϘϜʹ

if you type `\greeknumeral*{1996}` you get ͵αϡϙϛʹ

if you type `\Greeknumeral*{1996}` you get ͵ΑϠϘϚʹ

if you type `\greeknumeral{123456}` you get ͵ϱ͵κ͵γυνϝʹ

This was typeset on ιεʹ Ὀκτωβρίου ͵βιεʹ, which is October 15, 2015.

The teubner package offers also the possibility of typesetting the Attic numerals, without the need of loading Apostolos Syropoulos' athnum package; the functionality is the same, although the code is different; in order to avoid clashes, the teubner command for transforming Arabic numerals into Attic ones is `\AtticNumeral`. As for the original macro, the maximum value that can be transformed is 99999, while, of course, no vanishing or negative numbers can be transformed:

if you type `\AtticNumeral{2015}` you get ΧΧΔΠ

if you type `\AtticNumeral{1999}` you get Χ𐅅ΗΗΗΗ𐅄ΔΔΔΔΠΙΙΙΙ

if you type `\AtticNumeral{55555}` you get 𐅇𐅆𐅅𐅄Π

12 New commands

This extension package introduces many new commands for typesetting Greek in a philological way. Most such commands are collected in table 7.

A short remark on the command `\ap`: this useful command inserts *anything* as a superscript of anything else; it works both in text mode and in math mode[14]. In particular while typesetting a philological text in different languages and with different alphabets, `\ap` typesets the superscript with the current language and alphabet; if any change is required, the `\ap`'s argument can contain any language

[13] The stigma version is the standard one with the *babel* language support for Greek; with this package we adopted the digamma as the "regular" sign with the value of 6, and attributed stigma to the "variant", archaic representation of Milesian numbers. Here the archaic qoppa sign is used for the value 90; compare the archaic sign ϙ with the modern one ϟ.

[14] Numerical superscripts or apices do not require math mode; numerical footnote labels are automatically inserted by LaTeX's `\footnote` command; non numerical footnote labels are easily inserted with LaTeX's `\footnotemark` and `\footnotetext` commands with their optional arguments.

or alphabet specific declaration. You can typeset something such as *Βαχύλιδες*[a] by switching language and alphabet as required; the specific declarations and the commands contained in table 7 come handy also in these cases.

13 Metrics

Philological writings often require the description of metrics; for this purpose a new font has been developed that contains most of the frequent metric signs; the corresponding macros have been defined so as to set the metric glyphs as if they were text; but, most important, a new definition command has been introduced so as to enable to declare new control sequences to represent complete metric feet or even complete verse metrics.

The metric glyph names are collected in table 8, while the declaration command is described hereafter.

The syntax for that definition command is similar to that of `\newcommand`;

`\newmetrics{`⟨*name*⟩`}{`⟨*definition*⟩`}`

where ⟨*name*⟩ is a control sequence name made up of letters (as usual with LaTeX) with the exception that it may start with one of the digits 2, or 3, or 4. Of course the ⟨*definition*⟩ must reflect the replication by 2, or 3, or 4 times; moreover if the ⟨*name*⟩ starts with a digit, when it is used *it must be followed by a space*. Some examples follow:

```
\newmetrics{\iam}{\barbrevis\longa\brevis\longa}
\newmetrics{\2iam}{\iam\iam}
\newmetrics{\4MACRO}{\longa\longa\longa\longa}
```

The above definitions produce the following results (notice the space before the colon):

`\iam`: ⏑‒⏑‒

`\2iam␣`: ⏑‒⏑‒⏑‒⏑‒

`\4MACRO␣`: ‒‒‒‒

The definitions may contain also some symbols collected in table 7, such as ||, for example, and other symbols from the other tables.

Another important metric command is the following:

`\metricstack{`⟨*base*⟩`}{`⟨*superscript*⟩`}`

which is meant for superimposing some superscript (generally a number) over some metric symbol, which may be a single symbol or a metric foot, such as $\overset{48}{\text{‒}\breve{}\breve{}}$; since the superscript gets printed in math mode, the superscript hiatus $^{\mathrm{H}}$ may be obtained with `\Hiatus` when it falls between two metric symbols, but must be well described as a math roman element when it is superscripted over something else; similarly any other superscript which is not a math symbol must be suitably set as a math roman object.

The environment for setting metric sequences grouped with braces is described in the next section, since it is generally used within the composition of verses.

Example	Syntax	Example	Syntax
Βαχύλιδες	(declaration)	abcde	(declaration)
Βαχύλιδες	\textLipsias{⟨*text*⟩}	{*αβγ*}	\lesp{⟨*text*⟩}
Βαχύλιδες	\textDidot{⟨*text*⟩}	•	\LitNil
text	\textlatin{⟨*text*⟩}	g̃	\cap{⟨*letter*⟩}[a]
(*Βαχύλιδες*)	\frapar{⟨*text*⟩}	⟩—	\Coronis
(	\lpar	⌞	\lmqi
)	\rpar	⌟	\rmqi
(?)	\qmark	⌞*αβγ*⌟	\mqi{⟨*text*⟩}
…	\Dots[⟨*number*⟩]	⌜	\lmqs
. . .	\DOTS[⟨*number*⟩]	⌝	\rmqs
– – –	\Dashes[⟨*number*⟩]	⌜*αβγ*⌝	\mqs{⟨*text*⟩}
– – –	\DASHES[⟨*number*⟩]	$\overbrace{\alpha\beta\gamma}$	\zeugma{⟨*text*⟩}
$\widehat{\beta\alpha}$	\slzeugm{⟨*two letters*⟩}	$\overbrace{\alpha\beta}$	\rszeugma{⟨*two letters*⟩}
foo	\ap{⟨*text*⟩}	$\underbrace{\alpha\beta\gamma}$	\siniz{⟨*text*⟩}
⚲	\sinafia	——	\paragr
⋮	\:	——	\dparagr
⋮	\;	⊗	\FinisCarmen
⋮	\?	†	\crux
::	\antilabe	`*αβγ*´	\apici{⟨*text*⟩}
|	|	ˈ	\apex
‖	\dBar	~	\responsio
‖	\tBar	$\int$	\Int
[	\lbrk	$*a$	\star
]	\rbrk	$**a$	\dstar
⌊*αβγ*⌋	\ladd{⟨*text*⟩}	$***a$	\tstar
⟦*αβγ*⟧	\lladd{⟨*text*⟩}	|| ||	\,
⟨*αβγ*⟩	\Ladd{⟨*text*⟩}	|| ||	\!
⟪ *αβγ*⟫	\LLadd{⟨*text*⟩}	0123456789	\OSN{⟨*digits*⟩}
$\overbrace{\alpha\beta\gamma\delta\varepsilon\zeta\eta}$	\nexus{⟨*text*⟩}	k̛	\kclick
$\overset{\smile}{\mathrm{AB}}$	\Utie{⟨*2 letters*⟩}	*α***h***β*	\h
*α***j***β*	\yod	*aəβ*	\shwa
*α***q***β*	\q	*AFB*	\F
aF̄β	\f	i̱	\semiv{⟨*letter*⟩}
h^{v}	\skewstack{⟨*base*⟩}{⟨*apex*⟩}	ẹ̄	\md{⟨*letter*⟩}
ẹ̆	\Ud{⟨*letter*⟩}	ẹ̄	\mO{⟨*letter*⟩}
ẹ̆	\UO{⟨*letter*⟩}	ę	\Open{⟨*letter*⟩}
ę	\nasal{⟨*letter*⟩}	đ	\cut{⟨*b*|*d*|*g*⟩}
⊢	\dracma	⋌	\denarius
⚡	\stater	⌙	\etos
ↄ	\hemiobelion	ↄ	\tetartemorion
s̟	\splus	š	\stimes

[a] Ansten Mørch Klev spotted a conflict with the homonymous math command; this is now resolved. Thank you Ansten.

Table 7: Extended commands

Command	Metric symbol	Command	Metric symbol
\longa	–	\brevis	⏑
\bbrevis	⏒	\barbrevis	⊐
\ubarbrevis	⏓	\ubarbbrevis	⏔
\ubarsbrevis	⏓⏑⏔	\coronainv	⏑
\corona	⏜	\ElemInd	⏜
\catal	^	\ipercatal	+
\anceps	×	\banceps	x̄
\ancepsdbrevis	x̄̆	\hiatus[1]	H
\iam[2]	⊐–⏑–	\chor	–⏑⏑–
\enopl	⏑–⏑⏑–⏑⏑–	\4MACRO	––––
\aeolchorsor	–⏜⏑⏑⏜	\hexam	–⏑⏑–⏑⏑–⏑⏑–⏑⏑–⏑⏑––
\2tr	–⏑–× –⏑–×	\pentam	–⏔–⏔–‖–⏑⏑–⏑⏑–
\ubrevislonga	⊐	\aeolicbii	∞
\aeolicbiii	∞∞	\aeolicbiv	∞∞∞[3]

Table 8: Metric symbols

REMARKS

[1] A similar command \Hiatus produces the same visible result as \hiatus, except for the fact that it does not occupy horizontal space; it is useful in the definitions of full verse metrics where a hiatus needs to be inserted between two consecutive metric symbols; for example: $\overset{\text{H}}{--}$.

[2] This extension package predefines some examples of metric feet and complete verses.

[3] Sometimes it might be convenient to use a shortcut for inserting the Aeolic bases by inputting I or II or III, while the \metricsfont declaration is in force, in order to get ∞ or ∞∞ or ∞∞∞.

14 Poetry environments

In order to set poetry it is always possible to use the standard LaTeX `verse` environment; nevertheless such simple environment is not suited for philological purposes, except perhaps for very short citations. This extension package contains three new environments with various levels of complexity. Due to their relative complexity an example will be given for each one with both the input code and the corresponding result. All three environments require that any language change be declared before their opening statement, otherwise the language change lasts only to the end of the verse. It's worth noting that if you feel uncomfortable with Italian names for verses, you can use the Latin aliases, `versus`, `Versus`, and `VERSUS`.

`versi` This environment does not actually set each verse on a separate line; it rather resembles an in-line list; it resorts to a command `\verso` that inserts a small vertical separator with a progressive number over it. Both the environment opening and the command `\verso` accept arguments according to the following syntax:

```
\begin{versi}{⟨label⟩}
⟨verses⟩
\end{versi}
```

`\verso[`⟨*number*⟩`]`

where ⟨*label*⟩ is a short text (let's say not more than 15 characters) indicating for example the poem title and the stanza number; the whole set of verses will be typeset with a left margin wide enough to contain ⟨*label*⟩; the optional argument ⟨*number*⟩ indicates the starting value for the verse enumeration; the default value is 1, but if it is specified, it is required only with the first occurrence of `\verso` or when the enumeration is restarted. In this environment the standard LaTeX command `\\` behaves normally as in regular text.

```
\begin{versi}{Meropis fr. 3}
>'enj'' <o m'en e\ladd{>isplh} \verso[68] j'un
Mer'opwn k'ien. <h \ladd{d'e dia} \verso pr'o\\
a>iqem\hc i sj~htos \ladd{>'elassen.}
\verso <'o d'' >ex'equt''; o>u
g'ar \ladd{<omo~iai}\\
\ladd{>a} \verso j'anatai jnhta~isi bol\ladd{a'i kat'a}
\verso ga~ian >'asin.\\
prh\lladd{m}n\ladd{~hs d\dots} \verso thse. m'elas d'e
perie.\ladd{\dots}\verso rw
\end{versi}
```

Meropis fr. 3 *ἔνθ' ὁ μὲν ε[ἰσπλη]* [68] *θύν Μερόπων κίεν. ἡ [δὲ δια]* [69] *πρὸ αἰχεμῆι σθῆτος [ἔλασσεν.]* [70] *ὃ δ' ἐξέχυτ'· οὐ γὰρ [ὁμοῖαι] [ἀ]* [71] *θάναται θνηταῖσι βολ[αὶ κατὰ]* [72] *γαῖαν ἄσιν. πρη[μ]ν[ῆς δ. . .]* [73] *τησε. μέλας δὲ περιε.[. . .]* [74] *ρω*

Since each verse in this environment is not on a single line, unless it's deliberately specified, this environment may be used also for prose whose sentences are numbered as, for example, biblical versicles:

Mt: 6,8–6,13 [8] *μὴ οὖν ὁμοιωθῆτε αὐτοῖς· οἶδεν γὰρ ὁ πατὴρ ὑμῶν ὧν χρείαν ἔχετε πρὸ τοῦ ὑμᾶς αἰτῆσαι αὐτόν.* [9] *οὕτως οὖν προσεύχεσθε ὑμεῖς·*

Πάτερ ἡμῶν ὁ ἐν τοῖς οὐρανοῖς·
ἁγισθήτω τὸ ὄνομά σου·
[10] *ἐλθέτω ἡ βασιλεία σου·*
γενηθήτω τὸ θέλημά σου,
ὡς ἐν οὐρανῷ καὶ ἐπὶ γῆς·
[11] *τὸν ἄρτον ἡμῶν τὸν ἐπιούσιον δὸς ἡμῖν σήμερον·*
[12] *καὶ ἄφες ἡμῖν τὰ ὀφειλήματα ἡμῶν,*
ὡς καὶ ἡμεῖς ἀφήραμεν τοῖς ὀφειλέταις ἡμῶν·
[13] *καὶ μὴ εἰσενέγκῃς ἡμᾶς εἰς πειρασμόν,*
ἀλλὰ ῥῦσαι ἡμᾶς ἀπὸ τοῦ πονηροῦ,
ὅτι σου ἔστιν ἡ βασιλεία καὶ ἡ δύναμις καὶ ἡ δόξα
εἰς τοὺς αἰῶνας· ἀμήν.

Versi This environment is very similar to the standard LaTeX environment `verse`; the difference is that `Versi` automatically enumerates the verses (displaying only verse numbers that are multiples of 5) with a number in the left margin. The syntax is as follows:

```
\begin{Versi}[⟨number⟩]
⟨verses⟩
\end{Versi}
```

where ⟨*number*⟩ is the starting value of the verse enumeration; of course each verse is separated from the next one with the usual command `\\`, which has been redefined so that it just divides the verses and provides to the possible display of the verse number; it accepts the optional information that the standard LaTeX command usually accepts, both the asterisk and the vertical space amount.

```
\begin{Versi}[45]
ta; pr'osje qeir~wn b'ian\\
\paragr de\ladd{'i}xomen;
t‘a d" >epi'onta da\ladd{'imo}n srine~i.---\\
t'os" e>'ipen >ar'etaikmos <'hrws;\\
t{\rbrk}'afon d‘e naub'atai\\
f{\rbrk}wt‘os <uper'afanon\\[1ex]
j{\rbrk}'arsos; <Al'iou te gambr~wi q'olwsen >~htor
\end{Versi}
```

45 *τα· πρόσθε χειρῶν βίαν*
δε[ί]ξομεν· τὰ δ' ἐπιόντα δα[ίμο]ν σρινεῖ.—
τόσ' εἶπεν ἀρέταικμος ἥρως·
τ]άφον δὲ ναυβάται
φ]ωτὸς ὑπεράφανον
50 *θ]άρσος· Ἁλίου τε γαμβρῶι χόλωσεν ἦτορ*

VERSI This third poetry environment behaves similarly to `Versi` but it displays a double verse enumeration in the left margin. The principal verse enumeration is displayed when the value is a multiple of 5; the second enumeration, just to the left of the verses, may be turned on and off; when the secondary enumeration is on, the verses are flush left, while when it is off the verses are suitably indented. The turning on and off of the secondary enumeration is achieved by means of the commands `\SubVerso` and `\NoSubVerso`; the syntax is as follows:

```
\begin{VERSI}[⟨outer number⟩]
⟨verses⟩
\end{VERSI}
```

`\SubVerso[`⟨*inner number*⟩`]`
`\NoSubVerso`

where ⟨*outer number*⟩ is the starting value of the primary verse enumeration, while ⟨*inner number*⟩ is the starting value of the secondary enumeration. The commands `\SubVerso` and `\NoSubVerso` must be input at the very beginning of the verse they should be applicable to. The command `\\` behaves as in LaTeX, and accepts the usual optional arguments.

With the environments `Versi` and `VERSI`, when typesetting in two column format, you have the possibility of specifying `\BreakVersitrue` (and of course `\BreakVersifalse`) for allowing (or disallowing) line breaks of verses; broken verses are continued on the next line with a generous indentation so as to recognize them as belonging to the same verse; the verse counter is not incremented when breaking verses across lines.

```
\begin{VERSI}[40]
k'elomai pol\ua stonon\\
\SubVerso[18]
>er'uken <'ub\apex rin; o>u g‘ar >‘an j'eloi-\\
\NoSubVerso
m'' >'ambroton >erann‘on >Ao\lbrk ~us\\
\SubVerso
>ide~in f'aos, >epe'i tin'' >h"ij\siniz{'e\lbrk w}n\\
s‘u dam'aseias >a'ekon-\\
\NoSubVerso
ta; pr'osje qeir~wn b'ian\\
\SubVerso
\paragr de\ladd{'i}xomen;
t‘a d'' >epi'onta da\ladd{'imw}n krine~i.\GEcdq\\
\SubVerso[1]
t'os'' e>~ipen >ar'etaiqmos <'hrws;\\
t\rbrk 'afon d‘e na\ua batai\\
f\rbrk wt‘os <uper'afanon\\
j\rbrk 'arsos;\Dots[4]
\end{VERSI}
```

40 *κέλομαι πολύστονον*
18 *ἐρύκεν ὕβ᾽ριν· οὐ γὰρ ἄν θέλοι-*
μ᾽ ἄμβροτον ἐραννὸν Ἀο[ῦς
20 *ἰδεῖν φάος, ἐπεί τιν᾽ ἠϊθέ[ων*
21 *σὺ δαμάσειας ἀέκον-*
45 *τα· πρόσθε χειρῶν βίαν*
23 *δε[ί]ξομεν· τὰ δ᾽ ἐπιόντα δα[ίμω]ν κρινεῖ.*“
1 *τόσ᾽ εἶπεν ἀρέταιχμος ἥρως·*
2 *τ]άφον δὲ ναύβαται*
3 *φ]ωτὸς ὑπεράφανον*
50 4 *θ]άρσος·....*

bracedmetrics This is an environment different from the preceding ones, although it always deals with verses. Its purpose is to set the verse metric lines grouped with a right brace, so as to show the variants of a certain metric scheme.

In order to align the metric variants and in order to place the right brace in the proper place it is necessary to fix specific lengths in terms of a unit that is compatible with the metric symbols; therefore the syntax of such spacing command and of the environment itself is the following

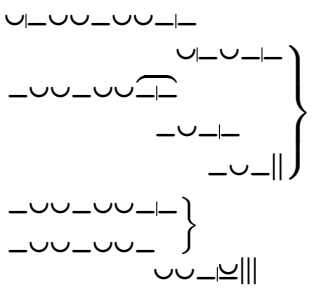
```
\begin{bracedmetrics}{⟨length⟩}
⟨metric lines⟩
\end{bracedmetrics}

\verseskip{⟨number⟩}
```

where ⟨*number*⟩ specifies the number of metric symbols the `\verseskip` should be equivalent to. Approximately the `\verseskip` will be as long as a sequence of ⟨*number*⟩ long syllables; the ⟨*length*⟩ specified as the width of the environment should equal the longest metric line contained in the block, and should be specified by means of the `\verseskip` command with its argument; but since the metric symbols are not all of the same length, it is wise to count the symbols of the longest metric line and to add a couple of units; after producing the first draft it is possible to review the number specified as the argument of `\verseskip`. Of course the same `\verseskip` command may be used to align the various fragments of metric lines within the environment. Examine the following example of input code:

```
\begin{verse}
    \brevis\svert\longa\brevis\brevis\longa
        \brevis\brevis\longa\svert\longa\\
    \begin{bracedmetrics}{\verseskip{13}}
        \Hfill \brevis\svert\longa\brevis\longa
            \svert\longa\\
        \longa\brevis\brevis\longa\brevis\brevis
            \zeugma{\longa\svert\longa}\\
        \Hfill\longa\brevis\longa\svert\longa
            \verseskip{2}\\
        \Hfill\longa\brevis\longa\dBar
    \end{bracedmetrics}\\
    \begin{bracedmetrics}{\longa\brevis\brevis\longa
        \brevis\brevis\longa\svert\longa}
        \longa\brevis\brevis\longa\brevis\brevis
            \longa\svert\longa\\
        \longa\brevis\brevis\longa\brevis\brevis\longa
    \end{bracedmetrics}\\
\verseskip{7}\brevis\brevis\longa\svert\ubarbrevis\tBar
\end{verse}
```

which produces:



Acknowledgements

This project was initially carried on with the help of Mr Paolo Ciacchi when he was writing his ancient Greek philology “master thesis”. After he got his “master’s” degree at the University of Trieste, I continued by myself, but I remain really indebted to Mr Ciacchi.