TLG Technical Note 002: Greek Sort Order

Authored: Nick Nicholas, TLG
Maintained by tlg@uci.edu
Created: October 1999
Last Revised: 2002-09-09

The following defines the algorithm used to sort Greek words in the TLG project. The sort is used to order the TLG word index, as distributed on CD ROM, and words as retrieved in the TLG search engine.

1. Preliminaries

A TLG word list is sorted primarily by Greek alphabetical order, according to the algorithm outlined below. For most purposes, it is necessary only to know the order of the Beta code letters (`'ABGDEVZHQIKLMNCOPRSTUFXYW`), and that smooth breathings sort before rough breathings, acute accents before circumflex. The actual sort performed to produce the word list is more precise, and is detailed below.

2. Sorting Key

To sort the list in Greek alphabetical order, a binary code is used (hereinafter referred to as the **sorting key**). The sorting key is derived from the ASCII representation of a Greek word, in Beta code. Each character is assigned the low seven bits of a byte, and the sign bit (bit 7) is turned on for each byte, to distinguish the sorting key from any surrounding ASCII where necessary.

> The sorting key is conceptually the same as that used by the Packard Humanities Institute in the production of TLG CD ROMs #B-#D, and on the IBYCUS computer. However, the IBYCUS sorting process used 5-bit values in a 16-bit word. The need to use modern microcomputer-compatible 8-bit words, and to avoid any 8-bit word having a null value (which would be taken as indicating the end of the string), has made it necessary for us to instead allocate 8 bits for each value.

To sort a Beta code word list, each word in the list is converted to a string containing the sorting key, then a tab character, then the ASCII original of the word. As a result, where the sorting key algorithm yields identical results for any two words, ASCII sort takes over for the ASCII original strings. For instance, the algorithm ignores a third iota subscript in a word; so when presented with the strings `A|A|AR` ᾳᾳᾳρ and `A|A|A|R` ᾳᾳᾳρ, it will generate an identical sorting key for both. The strings would then be sorted in ASCII order (as given in the remainder of the string), whereby `R` < `|`, hence `A|A|AR` < `A|A|A|R`.

With the exception of the partial word bytes (see below), each byte of the sorting key also has its 6th bit turned off, and its 5th bit turned on. The remaining five bits of each byte contain the same information as each 5-bit value in the IBYCUS code. That means that each byte contains 160 (binary `1010 0000`) + the 5-bit value.

> The input to the sorting algorithm is expected to have been subjected to normalization through the algorithm outlined in *TLG Technical Note 001*.

The sorting keys for any two words are compared byte by byte. If all bytes are equal up to the length of one word, the shorter string precedes the longer; for example, `KAI` καὶ < `KAINOS` καινος. In the following, the sorting keys computed for each word are given in hexadecimal (with the standard prefix `0x`.)

3. Letters and breathings

In this scheme, values 0-4 of the 5-bit sorting key value represent the Greek breathings, while values 5-30 represent the Greek alphabet:

- 0: No breathing
- 1: Smooth breathing after vowel (*not crasis*)
- 2: Smooth breathing after diphthong (*not crasis*)
- 3: Rough breathing after vowel (*not crasis*)
- 5-30: `'ABGDE VZHQI KLMNC OPRST UFXYW`
  (*'αβγδε Fζηθι κλμνξ οπρστ υφχψω*)

Each word is encoded so that the bytes representing each letter of the word (codes 5-30) are given first, followed by the breathing code (if any) for the entire word. For example, the following appear in sorted order:

`OI` *οι*

`0xB5AF [A0]`
160+**21** 160+**15** [160+**0**]
`O I` No breathing.

`OI)` *οἰ*

`0xB5AF A2`
160+**21** 160+**15** 160+**2**
`O I` Smooth breathing on diphthong.

`O(I` *ὁι*

`0xB5AF A3`
160+**21** 160+**15** 160+**3**
`O I` Rough breathing on vowel.

`OI(` *οἱ*

`0xB5AF A4`
160+**21** 160+**15** 160+**4**
`O I` Rough breathing on diphthong.

Given the sorting keys, the order of the words is:

1. `0xB5AF:  OI` *οι*
2. `0xB5AF A2: OI)` *οἰ*
3. `0xB5AF A3: O(I` *ὁι*
4. `0xB5AF A4: OI(` *οἱ*

4. Coronides

If there is crasis in a word, the code for the first coronis follows the breathing code. This code, like those following, is also a 5-bit value prefixed by the bits `101`. In that code, bit 0 indicates the type of coronis (0 if smooth, 1 if rough), and bits 1-4 indicate the location within the word (counting from the start of the word):

`KA)/N` *κἄν*

`0xB0A6 B3A0 A4BC`

160+**16**	160+**6**	160+**19**	160+**0**	160+2*(**2**)+	**0**	160+2*(15-1)+0
K	A	N	No breathing;	2nd letter from beginning:	Smooth coronis;	Acute (see below).

Sorting coronides separately from breathing marks allows a distinction to be made between `E)GW)=|MAI` *ἐγῷμαι* and `E)GW=|MAI` *ἐγῶμαι*. If there is no coronis in the word, the 5-bit value assigned is zero.

> On the TLG Search engine, rough coronides are normalized to smooth: thus no distinction is made between `XW)S` *χὡς* and `X(WS` *χὡς*, since the TLG Word Index already treats both as `XW)S` *χὡς*. The distinction between rough and smooth coronis is only preserved on TLG CD ROM #E as of this writing.

5. Accents

The coronis code is followed (if necessary) by an accent code. In this code, bit 0 indicates the accent (0 if acute, 1 if circumflex), and bits 1-4 indicate the location within the word (15 minus the number of letters from the end of the word: the last letter of the word has thus the location value 15-0). If the word is unaccented, or bears an accent before the 14th last letter, the location value is zero:

`OI(/` ***οἵ***

`0xB5AF A4A0 BE`

160+**21**	160+**15**	160+**4**	160+**0**	160+2*(15-**0**)+ **0**
O	I	Rough on diphthong;	No coronis;	Last letter: acute.

`MU=WY` ***μῦωψ***

`0xB2BA BEB9 A0A0 BB`

160+**18**	160+**26**	160+**30**	160+**25**	160+**0**	160+**0**
M	U	W	Y	No breathing;	No coronis;

160+2*(15-**2**)+ **1**

3rd last letter: circumflex.

`MUW=Y` ***μυῦψ***

`0xB2BA BEB9 A0A0 BD`

160+**18**	160+**26**	160+**30**	160+**25**	160+**0**	160+**0**
M	U	W	Y	No breathing;	No coronis;

160+2*(15-**1**)+ **1**

2nd last letter: circumflex.

Thus, `MU=WY` *μῦωψ* sorts before `MUW=Y` *μυῦψ*: `0xB2BA BEB9 A0A0` **`BB`** < `0xB2BA BEB9 A0A0` **`BD`**. In general, earlier accents sort before later accents in the word, and for words accented in the same position, acutes sort before circumflexes. (Graves are conflated with acutes in normalization, and are not considered separately.) In classical terms, the sort order is: *Proparoxytone* < *Paroxytone* < *Properispomenon* < *Oxytone* < *Perispomenon*.

6. Iota subscripts

The accent code, when necessary, is followed by the iota subscript code. In that code, the 5-bit value gives the location of the subscript: the number of letters from the end of the word, plus one (an absent value is denoted by a 5-bit value of zero):

`MU/WPA|` ***μύωπᾳ***

`0xB2BA BEB6 A6A0 A0B8 A1`

160+**18**	160+**26**	160+**30**	160+**22**	160+**6**	160+**0**

M	U	W	P	A	No breathing;
160+**0**	160+2*(15-**3**)+	**0**	160+(**0**+1)		
No coronis;	4th last letter:	acute;	Last letter: iota subscript.		

Thus words with later iota subscripts are sorted before words with earlier iota subscripts, *ceteris paribus*.

Up to two iota subscripts may be encoded for a word:

QRA=|KH| *θρᾷκῃ*

0xAEB7 A6AB ADA0 A0BB A3A1

160+**14**	160+**23**	160+**6**	160+**11**	160+**13**	160+**0**
Q	R	A	K	H	No breathing;
160+**0**	160+2*(15-**2**)+	**1**	160+(**2**+1)	160+(**0**+1)	
No coronis;	3rd last letter: circumflex;		3rd last letter: iota subscript;	Last letter: iota subscript.	

7. Hypodiastole

When necessary, the iota subscript codes are followed by a hypodiastole code, giving the location of the hypodiastole counting from the start of the word (where the first letter is 1):

O(/,TI *ὅ,τι*

0xB5B9 AFA3 A0BA A0A0 A2

160+**21**	160+**25**	160+**15**	160+**3**	160+**0**
O	T	I	Rough on vowel;	No coronis;
160+2*(15-**2**)	+**0**	160+**0**	160+**0**	
3rd last letter:	acute;	No iota subscript (1);	No iota subscript (2);	
160+**1**				
1st letter, hypodiastole.				

8. Diacritic Hierarchy

Note that the diacritic codes are included only as needed: the sorting key follows the hierarchy *letters > breathing > coronis > accent > 1st iota subscript > 2nd iota subscript > hypodiastole,* and a value is not output if no values lower than it are output.

Thus, the encoding of OI(*οἱ* need not include accent, subscript, coronis or hypodiastole information. The encoding of MOI *μοι* need not even include breathing information, and can be terminated in three bytes (encoding just the letters.) On the other hand, the encoding of PH| *πῃ* requires 6 bytes: a byte needs to be filled for the absent breathing, coronis and accent, before the subscript byte.

All such filler bytes contain the value zero for their 5-bit value, since the absence of a feature precedes the presence of the feature in sorting order. Thus,

- OI(*οἱ* < OI(/ *οἵ*

  (0xB5AF A4 < 0xB5AF A4**A0 BE**);

- PH| πῃ < PH=| πῆ

  (0xB6AC A0A0 **A0**A1 < 0xB5AC A0A0 **BF**A1).

9. Partial Words

The final byte of the sorting key contains partial word information. In this byte, to distinguish it from preceding bytes, the binary prefix is 100 instead of 101. Because of this, a partial word byte can be appended to the remainder of the key without superfluous subscript, accent, coronis or breathing bytes intervening. If the word is partial to the left, the byte has the value 144 (1001 0000); if it is partial to the right, it has the value 136 (1000 1000); if it is fragmentary on both sides, the byte has the value 152 (1001 1000). As a result, the following sorting order obtains:

1. TOI *τοι*
   0xB9B5 AA
   160+**25** 160+**21** 160+**10**
   T O I
2. TOI! *τοι.*
   0xB9B5 AA88
   160+**25** 160+**21** 160+**10** **136**
   T O I Partial to the right.
3. !TOI *.τοι*
   0xB9B5 AA90
   160+**25** 160+**21** 160+**10** **144**
   T O I Partial to the left.
4. !TOI! *.τοι.*
   0xB9 B5AA 98
   160+**25** 160+**21** 160+**10** **152**
   T O I Partial on both sides.
5. TOI= *τοῖ*
   0xB9B5 AAA0 A0BF
   160+**25** 160+**21** 160+**10** 160+**0** 160+**0** 160+2*(15-**0**)+ **1**
   T O I No breathing; No coronis; Last letter: circumflex.
6. TOI=! *τοῖ.*
   0xB9B5 AAA0 A0BF 88
   160+**25** 160+**21** 160+**10** 160+**0** 160+**0** 160+2*(15-**0**)+ **1** **136**
   T O I No breathing; No coronis; Last letter: circumflex; Partial to the right.

Because of the lower byte prefix, partial words always sort immediately after their equivalent full words, and before any words differing by additional diacritics. The following is a complete illustration of the TLG sorting algorithm.

1. EGWMHN *εγωμην* 0xA9A7 BEB2 AD**B3**
2. !EGWMHN *.εγωμην* 0xA9A7 BEB2 ADB3 **90**
3. E)GWMHN *ἐγωμην* 0xA9A7 BEB2 ADB3 **A1**
4. !E)GWMHN *.ἐγωμην* 0xA9A7 BEB2 ADB3 A1**90**
5. E)GW=MHN *ἐγῶμην* 0xA9A7 BEB2 ADB3 A1**A0 B9**
6. E)GW=|MHN *ἐγῷμην* 0xA9A7 BEB2 ADB3 A1A0 B9**A4**
7. E)GWMH/N *ἐγωμήν* 0xA9A7 BEB2 ADB3 A1A0 **BC**

8. E)GW)MHN *ἐγὼμην* 0xA9A7 BEB2 ADB3 A1**A6**

9. !E)GW)MHN *.ἐγὼμην* 0xA9A7 BEB2 ADB3 A1A6 **90**

10. E)GW)/MHN *ἐγὼ́μην* 0xA9A7 BEB2 ADB3 A1A6 **B8**

11. E)GW)/MHN! *ἐγὼ́μην.* 0xA9A7 BEB2 ADB3 A1A6 B8**88**

12. E)GW)=MHN *ἐγὼ̃μην* 0xA9A7 BEB2 ADB3 A1A6 **B9**

13. !E)GW)=MHN *.ἐγὼ̃μην* 0xA9A7 BEB2 ADB3 A1A6 B9**90**

14. E)GW)=|MHN *ἐγῲ̃μην* 0xA9A7 BEB2 ADB3 A1A6 B9**A4**

15. E)GW)=|MH|N *ἐγῲ̃μῃν* 0xA9A7 BEB2 ADB3 A1A6 B9A4 **A2**

16. E)GW)=|,MH|N *ἐγῲ̓̃,μῃν* 0xA9A7 BEB2 ADB3 A1A6 B9A4 A2**A3**

17. E)GW)=|,MH|N! *ἐγῲ̓̃,μῃν.* 0xA9A7 BEB2 ADB3 A1A6 B9A4 A2A3 **88**

18. !E)GW)=|,MH|N! *.ἐγῲ̓̃,μῃν.* 0xA9A7 BEB2 ADB3 A1A6 B9A4 A2A3 **98**

19. E)GW)MH/N *ἐγὼμήν* 0xA9A7 BEB2 ADB3 A1A6 **BC**

20. E)GW(MH/N *ἐγὼ́μήν* 0xA9A7 BEB2 ADB3 A1**A7**

21. E)GWMHN *ἐγωμην* 0xA9A7 BEB2 ADB3 **A3**

22. E(GW=MHN *ἕγῶ̃μην* 0xA9A7 BEB2 ADB3 A3**A0 B9**

23. E(GW)=|MH|N *ἕγῲ̃μῃν* 0xA9A7 BEB2 ADB3 A3**A6 B9A4 A2**

24. EGWMHC *εγωμηξ* 0xA9A7 BEB2 AD**B4**