Font Creation Report

29 July, 2009

Prepared by: Mr. HENG, SOTHEARITH

Cambodia Country Component

PAN Localization project

PAN Localization Cambodia (PLC) of IDRC

CONTENTS

I. INTRODUCTION TO FONT

There are three types of font:

- Bitmap font :

  A bitmap font is one that stores each glyph as an array of pixel It is less commonly known as a raster font. Bitmap fonts look best at their native pixel size. At nonnative size, many text rendering system perform nearest-neighbor resembling introducing ugly jagged edges. More advance system perform anti-aliasing on bitmap fonts whose size does not match the size that the application requests. This technique work well for increasing the size, as it tends to blur the edge.

- True type font :
  - TrueType is a digital font technology designed by Apple Computer, and now used by both Apple and Microsoft in their operating systems.
  - A digital font contains much more than just the characters associated with a given alphabet or script. A TrueType font file includes
    - Many different kinds of information used by the TrueType rasterizer and the operating system software to ensure that characters display on the computer screen or print out exactly as the font designer intended them to
    - Information about how the characters should be spaced vertically and horizontally within a block of text, character mapping details (governing the variety of characters included in the font and the keystrokes needed to access them), and much more besides.
    - Manufacturer's details, such as copyrights, names and licensing permissions
- Open type font

  - Allow a rich mapping between characters and glyphs, which supports ligatures, positional forms, alternates, and other substitutions.
  - Include information to support features for two-dimensional positioning and glyph attachment.
  - Contain explicit script and language information, so a text-processing application can adjust its behavior accordingly.
  - Have an open format that allows font developers to define their own typographical features.

II. PLANNING

In Khmer language we have

- 35 character normal:

ក, ខ, គ, ឃ, ង, ច, ឆ, ជ, ឈ, ញ, ដ, ឋ, ឌ, ឍ, ណ, ត, ថ, ទ, ធ, ន, ប, ផ, ព, ភ, ម, យ, រ, ល, វ, ស, ហ, ឡ, អ

- Independent vowel:

ឥ, ឦ, ឧ, ឩ, ឪ, ឫ, ឬ, ឭ, ឮ, ឯ,

III. FONT DESIGN

- **Tools : Fontlab**

Font plays important role in displaying the characters set on the screen of computer. In Khmer script, font is designed base on six lines to support Khmer writing and easy to read.

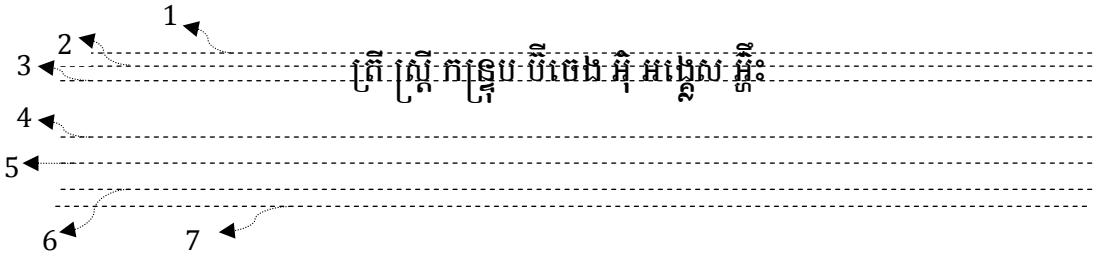


Table 1: Line position

Line position	Khmer characters
1	◌̇
2	◌ិ, ◌ី, ◌ឹ, ◌ឺ, ◌ំ, ◌̄, ◌̇, ◌៉, ◌៎, ◌៏, ◌̈, ◌́
3	◌ិ, ◌ី, ◌ឹ, ◌ឺ, ◌ើ, ◌ៃ, ◌ៃ, ◌ំ, ◌̄, ◌̇, ◌៉, ◌៎, ◌ិ, ◌៏, ◌̈, ◌́, ◌̃

4	ក, ខ, គ, ឃ, ង, ច, ឆ, ជ, ឈ, ញ, ដ, ឋ, ឌ, ឍ, ណ, ត, ថ, ទ, ធ, ន, ប, ផ, ព, ភ, ម, យ, រ, ល, វ, ស, ហ, ឡ, អ ...
5	◌្ក, ◌្ខ, ◌្គ, ◌្ឃ, ◌្ង, ◌្ឆ, ◌្ជ, ◌្ឈ, ◌្ញ, ◌្ដ, ◌្ឋ, ◌្ឌ, ◌្ឍ,◌្ណ, ◌្ធ, ◌្ផ, ◌្ព, ◌្ភ, ◌្ម, ◌្ត, ◌្ល, ◌្ថ, ◌្ស, ◌្ទ, ◌្យ, ◌្រ, ◌្ល, ◌្វ, ◌្ប, ◌្ហ, ◌្ឡ, ◌្អ, ◌ុ, ◌ូ, ◌ួ ...
6	◌្ក, ◌្ខ, ◌្គ, ◌្ឃ, ◌្ង, ◌្ឆ, ◌្ជ, ◌្ឈ, ◌្ញ, ◌្ដ, ◌្ឋ, ◌្ឌ, ◌្ឍ,◌្ណ, ◌្ធ, ◌្ផ, ◌្ព, ◌្ភ, ◌្ម, ◌្ត, ◌្ល, ◌្ថ, ◌្ស, ◌្ទ, ◌្យ, ◌្រ, ◌្ល, ◌្វ, ◌្ប, ◌្ហ, ◌្ឡ, ◌្អ, ◌ុ, ◌ូ, ◌ួ ...
7	

In Khmer writing culture some characters such as ◌ា, ◌្រ, ◌្ឃ, ◌្ឈ, ◌្ឍ, ◌្យ, ◌្ប, ◌្ផ, ◌ើ, ◌ៀ should be change automatically when has other subscribe.

Ex: ក + ◌ា → កា

ប + ◌ា → បា

ក្ត + ◌្ឃ → ក្ត្ឃ

ស្ត + ◌្រ → ស្ត្រ

ក្ត + ◌ៀ → ក្តៀ

To make change of all these characters, we have to design all their shape in the font file.

IV. KHMER RENDERING ENGINE

The rendering engine for Khmer language has a role to process the code of Khmer typing into the sequence of code will be showed on the screen.

Ex:

ម ◌ា ន ក េ◌ា អ ◌ី ម ◌ួ យ → មានកៅអីមួយ

ហ េ◌ា ម ◌ិ ន ឆ ◌្ល េ◌ី យ រ េ◌ៀ ន → ហៅមិនឆ្លើយរៀន

V. KHMER UNICODE RANGE

VI. ADDING OPEN TYPE TABLE

Tools: Msvolt

1. **Set glyph name and Unicode**

Before we start creating Open type table, we have to assign the name and Unicode value to each glyph. A usable name that is easily remembered during the process of making the Open Type tables should be assign as below:

Ex:	name	Unicode
	uni1780	uni+1780
	uni1781	uni+1781
	uni1782	uni+1782
	uni1783	uni+1783

2. **Set glyph property**

After we have already assigned the name and Unicode, we have to set glyph property.

- mark for the glyph that are used as subscript such as ◌ុ,◌ូ, ◌ួ …
- simple for the glyph that are used as base such as ក, ខ, ត …
- ligature for the glyph that are used as the substitution of glyph such as កា, ខា, គា ...
- component for the glyph ◌ើ, ◌ៀ

3. **Add script**

We have to add a script and language pair that will contain our OTL table.

Ex : khmer<khmr>

4. **Add glyph group**

We should put the glyph into group before creating OTL table. With glyph group, we can work faster and easier.

5. **Add feature**

- **Pre-base form**

Feature Tag: "pref"

The 'pref' feature is used to substitute the pre-base form of RO to other base glyph.

Associated lookup:

- pref_form : when we type “ਲਂ”+”ੁ”+”ਾ” and we want the result on the screen is “ਲ੍ਂ”, so we have to change ਾ to ੍ਲ. The rendering engine will put ੍ਲ before ਲਂ automatically.

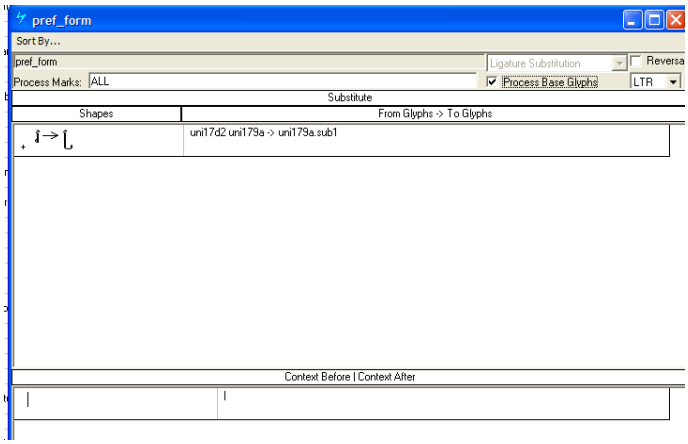


Figure 1: pref_form

- **Below-base form**

Feature Tag: "blwf"

The 'blwf' feature is used to substitute the below-base form of a consonant in conjuncts.

Associated lookup:

- base_below : to write the subscript of consonant such as ੍ਹ, ੍ਙ, ੍ਕ … . When we type ਨ੍ਹ, ਨ ,ਿ we get ਨ੍ਹਿ; the rendering engine will put ੍ਹ below ਨ automatically.

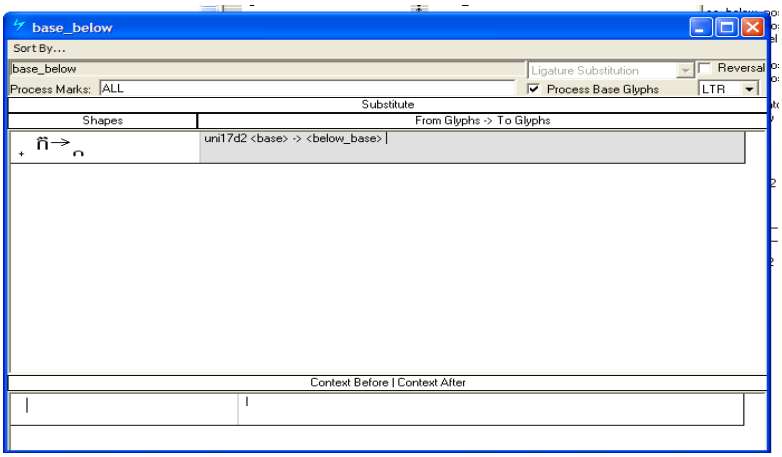


Figure 2: base_below

- **Above-base form**

Feature Tag: "abvf"

The 'abvf' feature is applied to the glyph to substitute the Unicode defined shape to the portion of the glyph that is located above the base glyph. This is possible because the piece of the letter which is displayed before the letter has been inserted into the glyph store in front of the base glyph by the Uniscribe engine.

Associated lookup:

- above_mark : when we type ਪਾ ੰ the keyboard will generate ਪਾਪਾ ੰ. To avoid this problem we have change ਿ ੰ to ੰ. When we type ਨ ਿ ੰ the rendering engine will substitute ਨ with ਪਾ and substitute ਪਨ with ੰ. We get result ਪਨੰ.

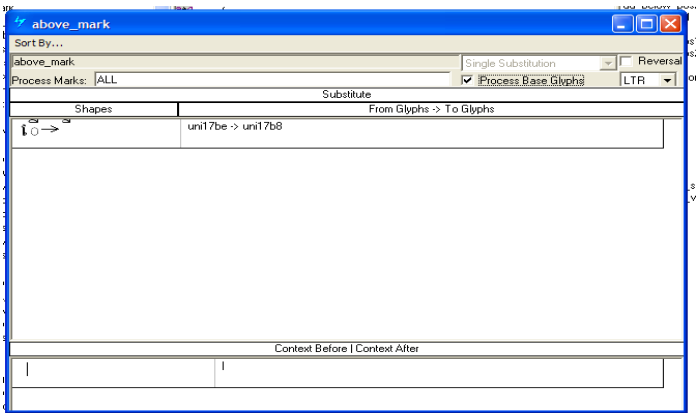


Figure 3 : above_mark

- zero_non_joiner : to separate the glyph when we type zero no joiner.

Ex: ਸੰ ੱ zero non joiner ੀ → ਸੰੱੀ not ਸ੍ਰੀ

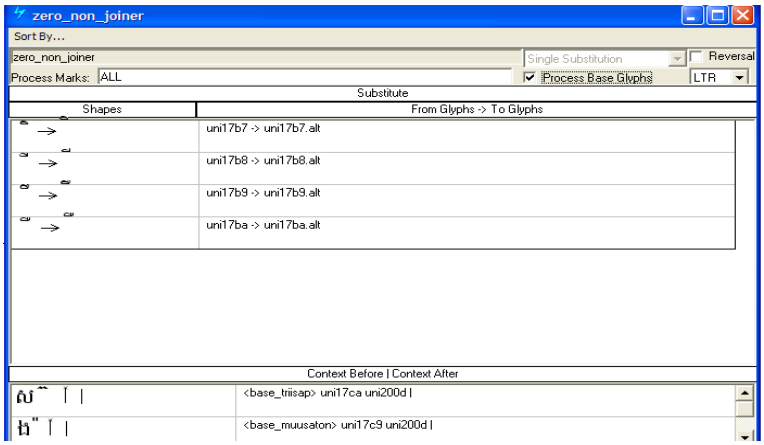


Figure 4: zero_non_joiner

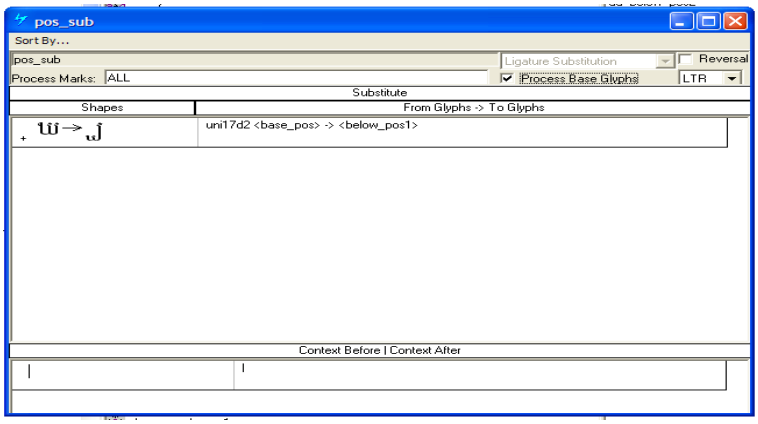
- **Post-base form**

Feature Tag: "pstf"

The 'pref' feature is used to substitute the post-base form with a post-base ligature.

Associated lookup:

- pos_sub : to write the pos subscript of consonant ਘ, ਙ, ਲ, ਵ, ਸ, ਮ. When we type ਨ੍ਹ, ਘ, ਘ,ਿ we get ਨ੍ਹਿ; the rendering engine will put ੍ਹ below ਨ automatically.



-

Figure 5: pos_sub

- mark_nyam :
- mark_sam : to write the word ស៊ាំ …
- mark_bang : to write the word ប៉ាំ …

- **Pre-base substitutions**

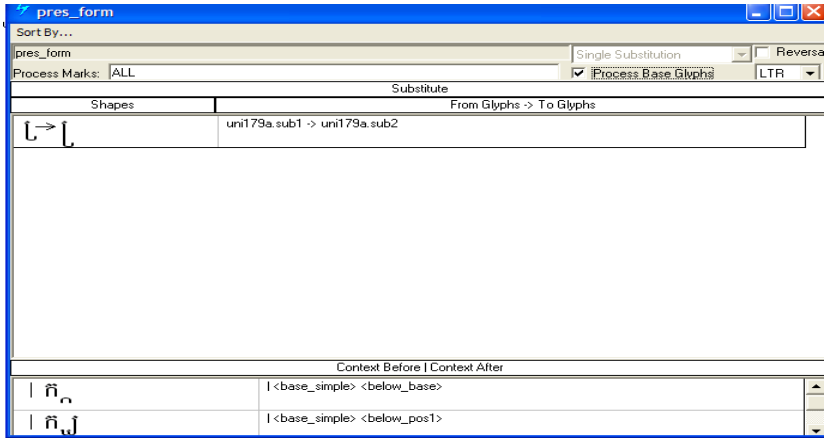
Feature Tag: **"pres"**

The 'pres' feature is used to produce the pre-base form of conjuncts.

Associated lookup:

- **pres_form** : to write subscript of RO when already has subscript.

Ex: ង្គ, ន្ត …



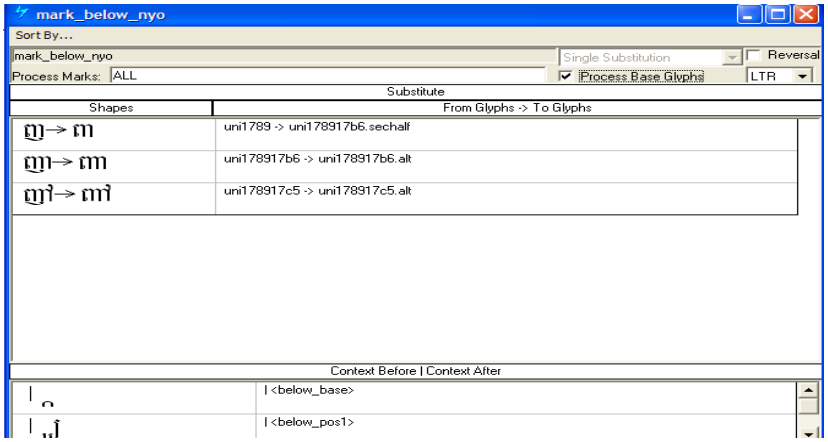
- **Below-base substitutions**

Feature Tag: "blws"

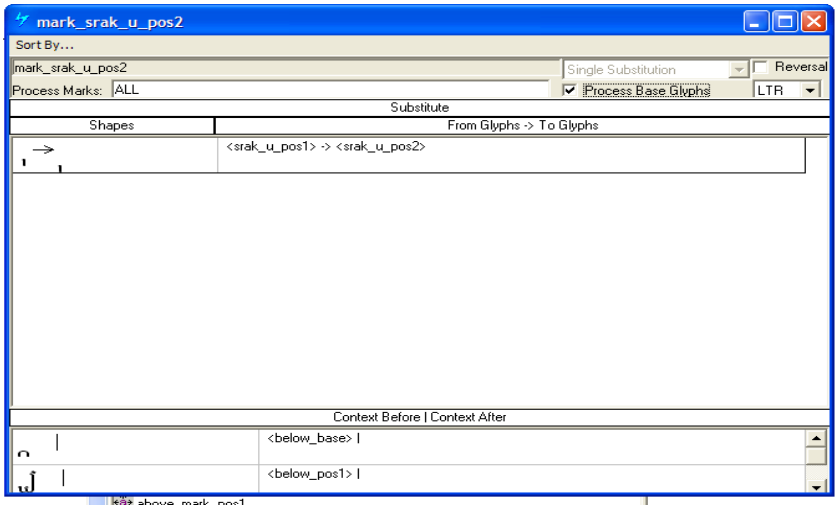
The 'blws' feature is used to produce the below-base substitutions that may be required for typographical correctness.

Associated lookup:

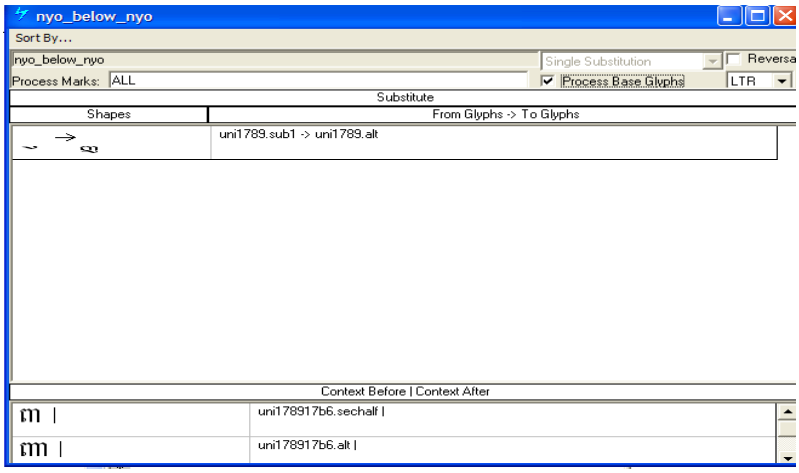
- mark_below_nyo : to mark subscript below nyo and change the nyo to nyo.alt



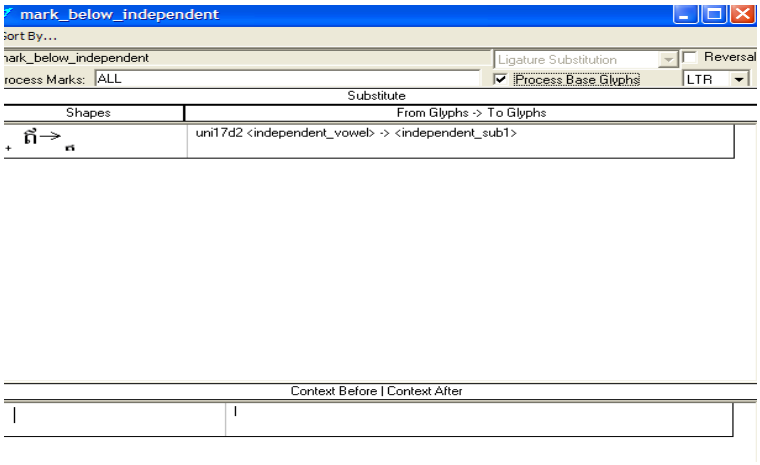
- mark_srak_u_pos2 : to mark srak u when we already have as subscript. We change er to er2



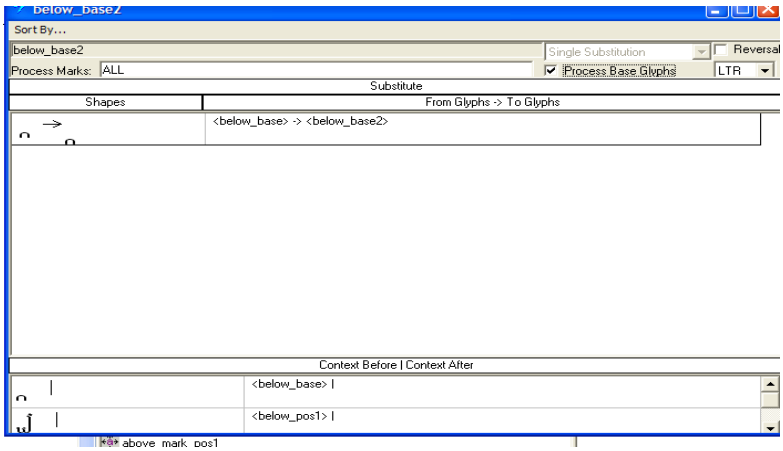
- nyo_below_nyo : to mark the subscript nyo below nyo



- mark_below_independent : to mark the subscript of independent vowel.



- below_base2 : to mark the second subscript.



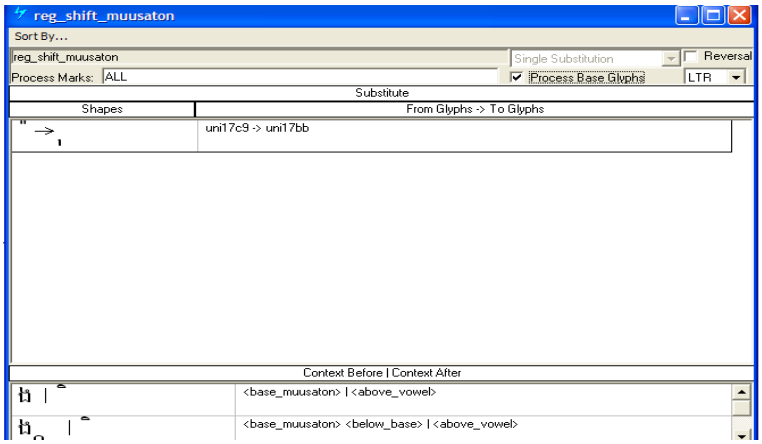
- **Above-base substitutions**

Feature Tag: "abvs"

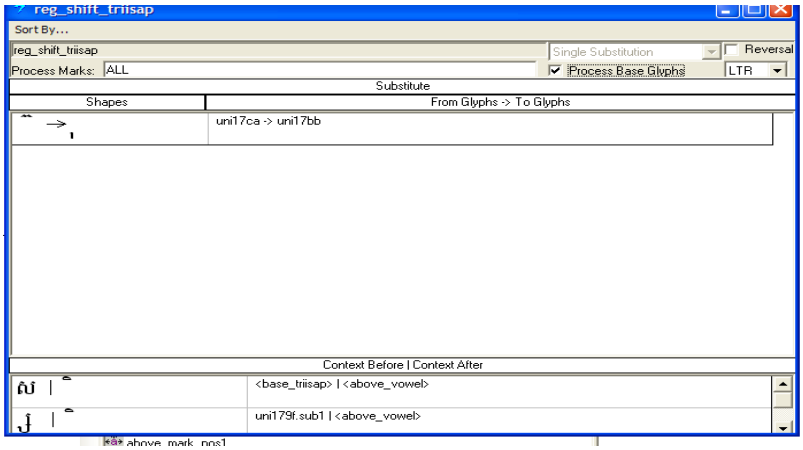
The 'abvs' feature is used to substitute the above-base substitutions that may be required for typographical correctness.

Associated lookup:

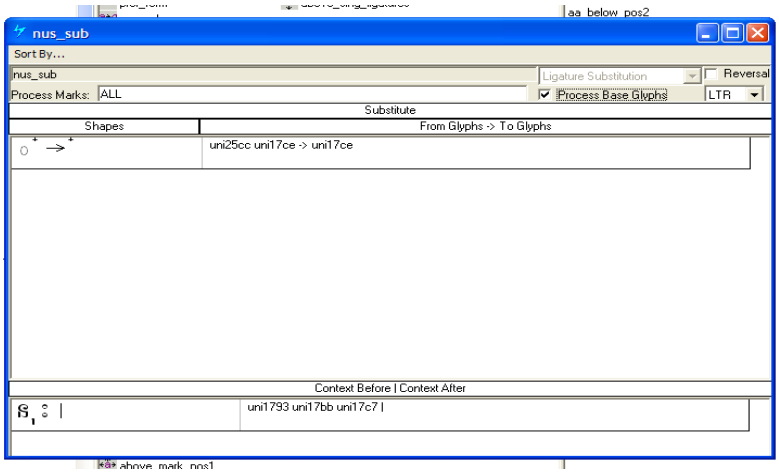
- Reg_shift_muusaton : to write the substitution in khmer language.



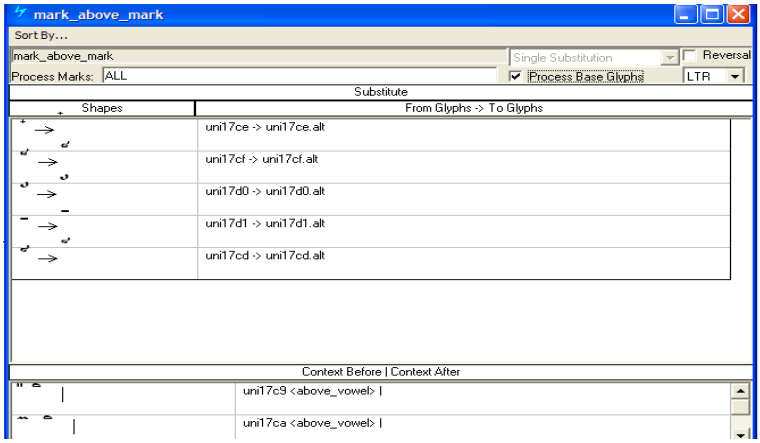
- Reg_shift_triisap



- Nus_sub



- Mark_above_mark



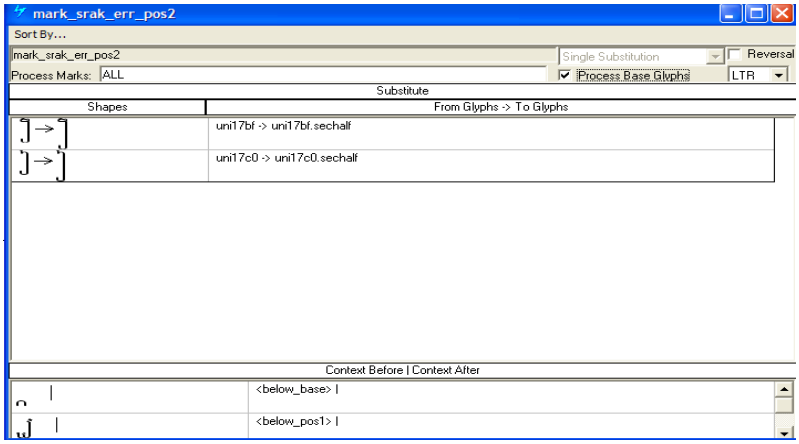
- **Post-base substitutions**

Feature Tag: "psts"

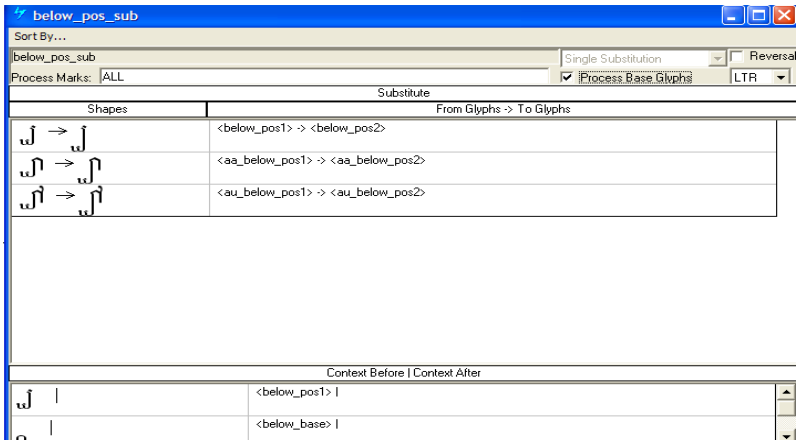
The 'psts' feature is used to substitute the post-base substitutions that may be required for typographical correctness. For example, a subscript type 3 glyph that needs to have a lower descent when a subscript type 1 glyph is on the syllable.

Associated lookup:

- mark_srak_er_pos2 : to mark srak er when we already have subscript.



- below_pos_sub : to mark the pos-substitution



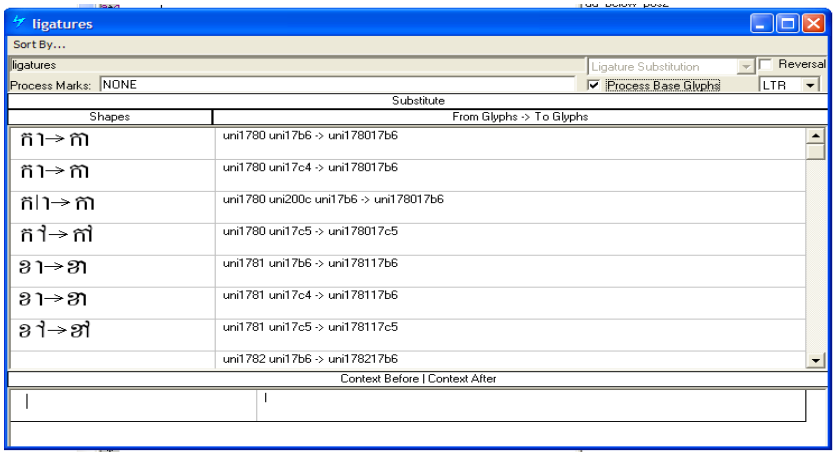
- **Contextual ligatures**

Feature Tag: "clig"

The 'clig' feature is used to map glyphs to their contextual ligated form. Unlike other ligature features, 'clig' specifies the context in which the ligature is recommended. This capability is important in some script designs and for swash ligatures. The 'clig' table maps sequences of glyphs to corresponding ligatures in a chained context (GSUB lookup type 8). Ligatures with more components must be stored ahead of those with fewer components in order to be found.

Associated lookup:

- ligatures : to mark srak aa to glyph.



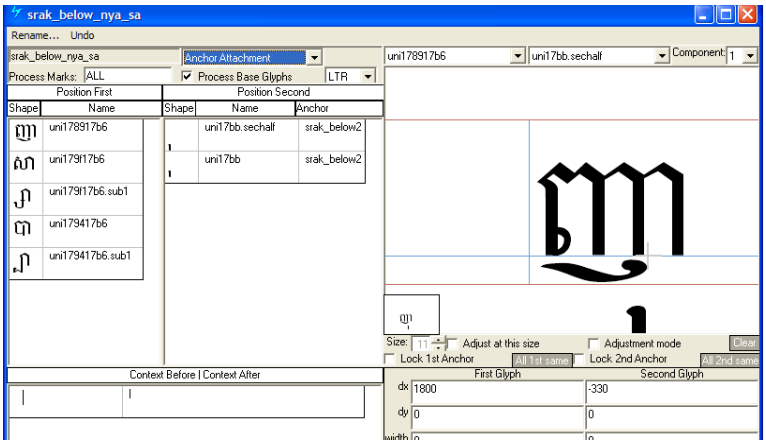
- **Below-base mark positioning**

Feature Tag: "blwm"

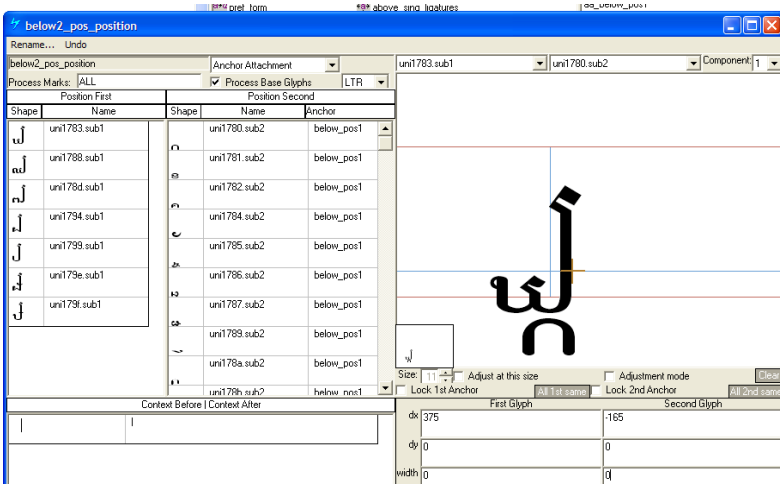
The 'blwm' feature is used to position all below-base marks on base glyphs. The best method for encoding this feature in an OpenType font is to use a chaining context positioning lookup that triggers mark-to-base and mark-to-mark attachments for below-base marks. The 'blwm' table provides positioning information (x,y) to enable mark positioning.

Associated lookup:

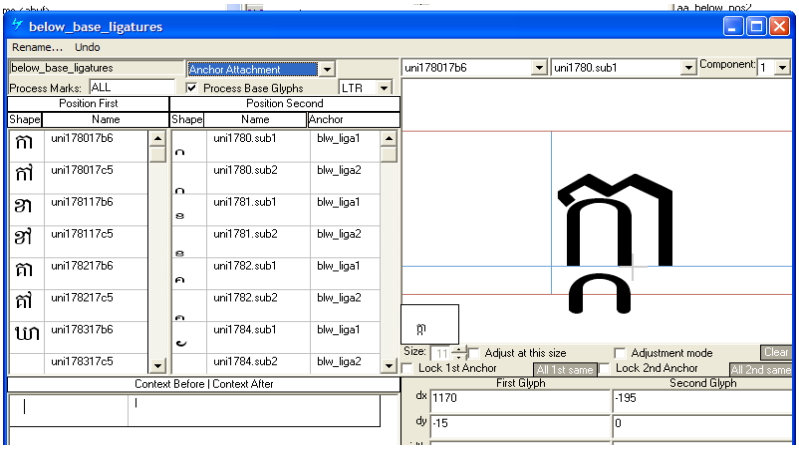
- srak_below_nya_sa : to adjust the position between ស្ន, ញ្ញ and ុ



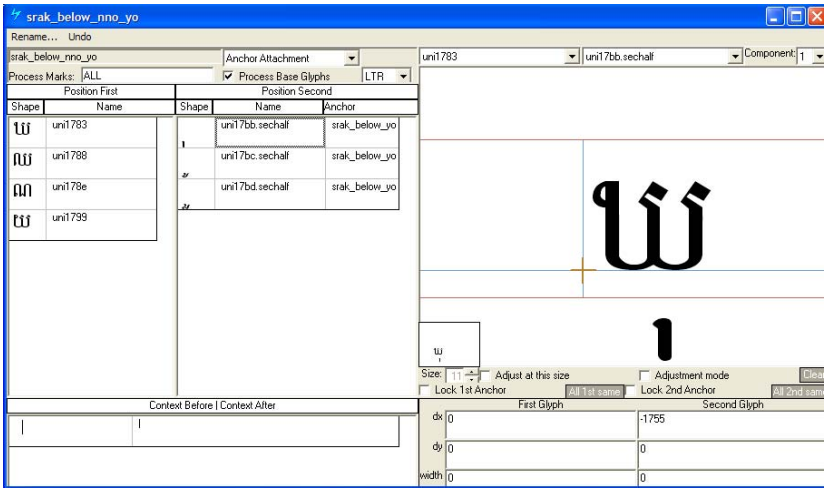
- below2_pos_position : to adjust the position between pos-subscript level1 and below subscript level 2.



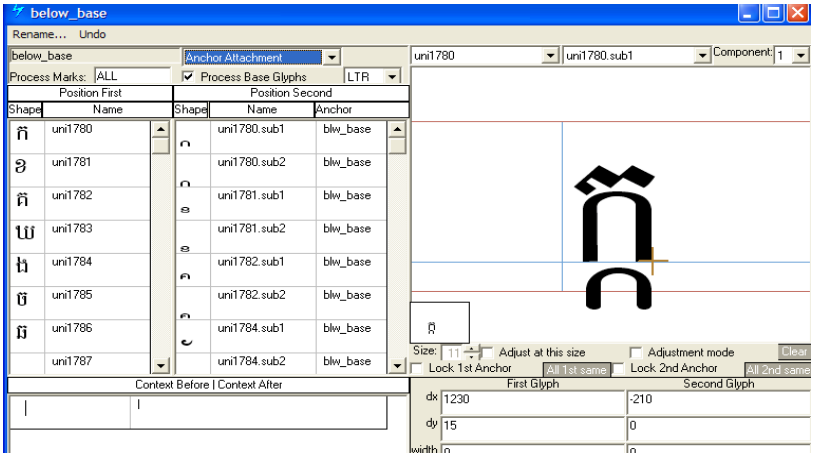
- below2_pos_position_ligature : to adjust the position between pos-subscript ligature level1 and below subscript level 2.
- below_base_ligatures : to adjust the position between base ligatures and below subscript.



- srak_below_nno_yo : to adjust the position between consonant and below vowel.



- below_base : : to adjust the position between consonant and below subscript.



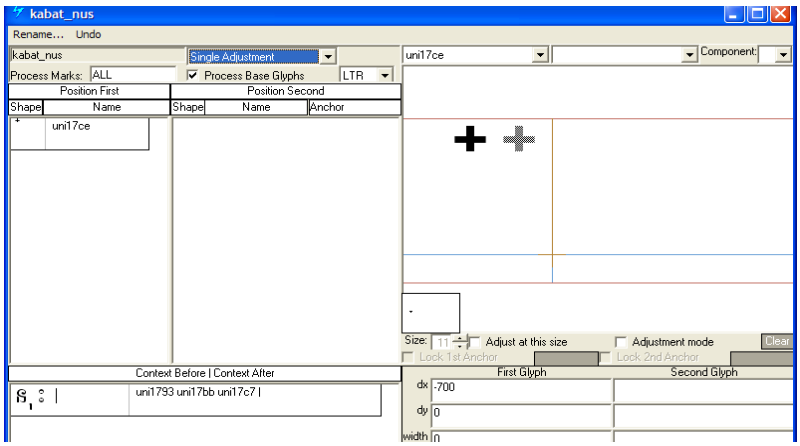
- **Above-base mark positioning**

Feature Tag: "abvm"

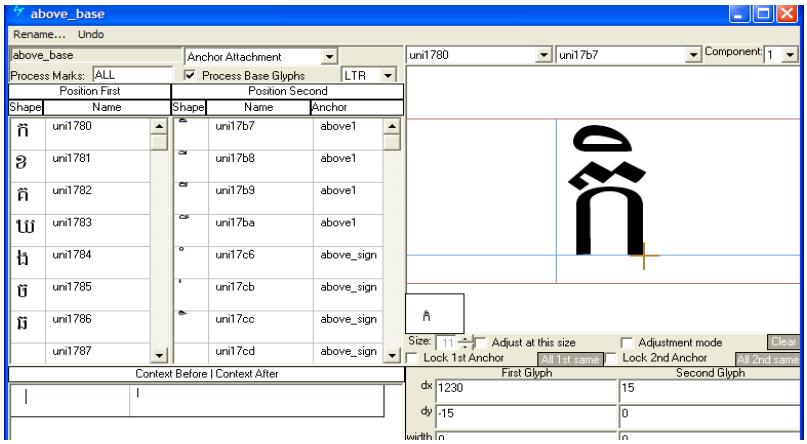
The 'abvm' feature is used to position all above-base marks on base glyphs. The best method for encoding this feature in an OpenType font is to use a chaining context positioning lookup that triggers mark-to-base and mark-to-mark attachments for below-base marks. The 'abvm' table provides positioning information (x,y) to enable mark positioning.

Associated lookup:

- kabat_nus : : adjust the position between ន្ទះ and ◌̊.



- above_base : to adjust the position between consonant and above vowel, above sign.



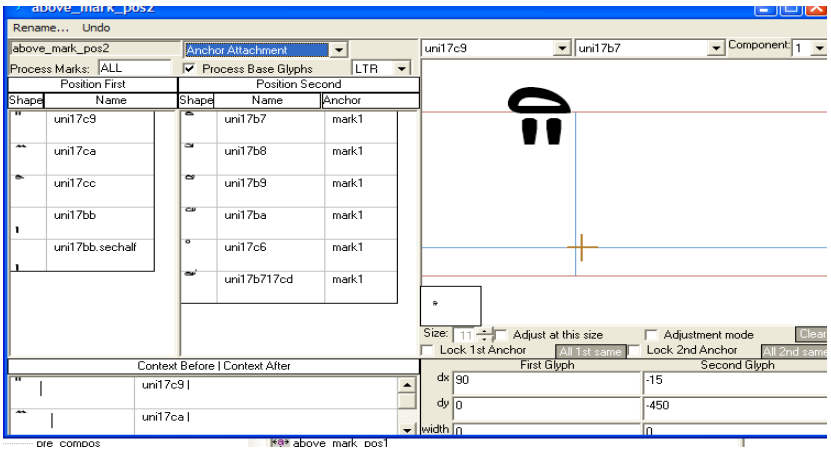
- **Mark to mark positioning**

Feature Tag: "mkmk"

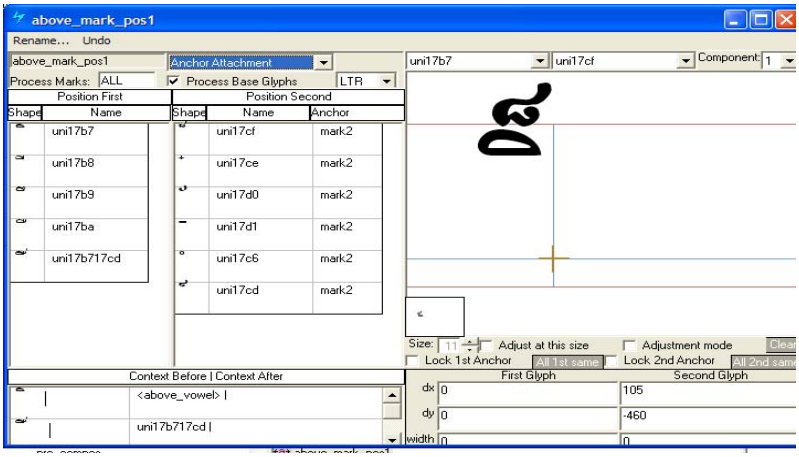
The 'mkmk' feature positions mark glyphs in relation to another mark glyph. This feature may be implemented as a Mark To Mark Attachment lookup.

Associated lookup:

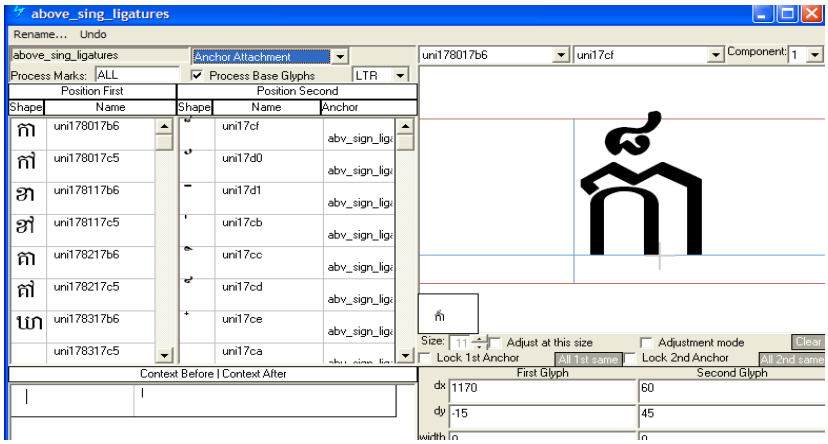
- above_mark_pos2 : to adjust the position between above sign and above vowel.



- above_mark_pos1 : to adjust the position between above vowel and above sign.



- above_sign_ligatures : to adjust the position between base ligature and above sign, above vowel.



- **Decomposition**

Feature Tag : <ccmp>

Associated lookup:

- Pre_compos : when we type ◌ِ and ◌ٔ, we change to ◌ٕ