Khmer orthographic syllables

Norbert Lindenberg
2023-01-12

Introduction

This document provides a proposal for updating section 16.4 Khmer of The Unicode Standard with information about an improved orthographic syllable structure for the Khmer script. The proposal is based on the discussion in L2/22-290 Khmer Encoding Structure (Nov 2022), which is the result of a collaboration between SIL International, the Royal Academy of Cambodia, the Cambodia Academy of Digital Technology, and individual contributors. It is provided to the Unicode Technical Committee now to encourage feedback before *Khmer Encoding Structure* is published and implemented in Cambodia.

A well-defined orthographic syllable structure is essential to ensure interoperability between smart keyboards, predictive input systems, spelling checkers, font rendering systems, fonts, systems for searching and sorting text, optical character recognition systems, speech input and output systems, text normalization, and other text processing software. It is also required as a basis for subsetting for security-sensitive applications, such as domain names and identifiers in programs.

The Khmer script has so far lacked a well-defined and agreed-upon orthographic syllable structure. At least three publications contain attempts at defining one:

- the Unicode Standard, section 16.4 Khmer (revised in Unicode 4),
- the description of the OpenType Khmer shaping engine,
- the widely used guide How to Type Khmer Unicode by the Open Forum of Cambodia.

These three attempts are all incomplete and incompatible. Fonts and OpenType implementations also differ substantially in what they consider valid or invalid, as documented in Issues in Khmer syllable validation. They often allow multiple encoded character sequences to represent the same orthographic syllable, causing problems in search and possible spoofing, as documented in Spoof-Vulnerable Rendering in Khmer Unicode Implementations. The ICANN Root Zone Label Generation Rules for the Khmer Script define a syllable structure that is not a subset of any of the three above.

The proposed encoding structure is generally compatible with existing fonts and font rendering systems for Modern Khmer, but has some compatibility issues with Middle Khmer and edge cases in Modern Khmer. The issues and a transition plan that minimizes them are discussed on pages 37-42 of *Khmer Encoding Structure (Nov 2022)*.

Document history

- L2/21-241 Khmer Encoding Structure – earlier version, with cover letter.
- L2/22-023 Recommendations to UTC #170 January 2022 on Script Proposals – includes the Script Ad Hoc's comments on the above in section 13 Khmer.
- L2/22-290 Khmer Encoding Structure (Nov 2022).

Since the earlier version, many of the issues in it have been resolved, including a strong unification of Middle and Modern Khmer. Solutions have been found to encoding final coengs in a way that greatly reduces the likelihood of their accidental abuse in Modern Khmer (where they do not occur). This makes the whole question of whether separate script handling is required for Middle Khmer no longer relevant since the structures are the same.

Proposal

The structure of orthographic syllables proposed in *Khmer Encoding Structure* requires at least the following changes to section 16.4 Khmer of the Unicode Standard (page numbers are as of Unicode 15.0).

① Update the subsection "Ordering of Syllable Components"

Change the first paragraphs of the subsection "Ordering of Syllable Components" (page 686), ending with "five coded characters", to use the orthographic syllable structure derived in *Khmer Encoding Structure (Nov 2022)*, pages 1-21. As in that document, this proposal, for ease of reading, omits the "U+" prefix that the Unicode Standard requires for Unicode code points in its EBNF. This prefix will have to be added when editing the Standard.

> ***Ordering of Syllable Components.*** The standard order of components in an orthographic syllable as expressed in extended BNF (see Appendix A.2) is
>
> ~~B{R|C}{S{R}}*{{Z}V}{O}{S}~~
>
> ~~where~~
>
> ~~B is a base character (consonant character, independent vowel character, and so on)~~
>
> ~~R is a *robat*~~
>
> ~~C is a consonant shifter~~
>
> ~~S is a subscript consonant or independent vowel sign~~

~~V is a dependent vowel sign~~

~~Z is a zero width non-joiner or a zero width joiner~~

~~0 any other sign~~

Base Robat? Coengs? FinalCoeng? Shifter? Vowels? Modifiers? Final?

where

Base := [1780-17A2 17A5-17B3]
Robat := 17CC

Coengs := (Conjoiner BaseNonRo)? Conjoiner Base
Conjoiner := 17D2
BaseNonRo := [Base -- [179A]]

FinalCoeng :=
    Conjoiner ZWJ BaseRightCoeng
      (?= ShifterSeq? (VowelBelowContext? VowelSignAboveContext | VowelBelowContext))
  | FinalCoengNonRo
      (?= ShifterSeq?
        [17C2-17C3]? VowelBelowContext? VowelAboveContext? VowelRightContext)
  | (?<= ConsGroupWeak) Conjoiner ZWJ ConsStrong (?= 17C9 ZWNJ? VowelSignAboveContext)
BaseRightCoeng := [1783 1788 178D 1794 1799 179E 179F 17A1]
ShifterSeq := [17C9 17CA] ZWNJ?
VowelBelowContext := [17C1-17C3]? [17BB-17BD]
VowelSignAboveContext :=
    VowelAboveContext
  | [17C1-17C3]? 17D0
VowelAboveContext :=
    [17C1-17C3]? [17B7-17BA 17BE 17DD]
  | 17B6 17C6
VowelRightContext := [17B6 17BF 17C0 17C4 17C5]
FinalCoengNonRo := Conjoiner ZWJ BaseNonRo
ConsGroupWeak :=
    ConsWeak Robat? (Conjoiner ConsWeak){0,2}
  | BA Robat? (Conjoiner Base){0,2}
  | Base Robat? Conjoiner BA (Conjoiner Base)?
ConsWeak := [1784 1789 178E 1793 1798-179D 17A1]
BA := 1794
ZWJ := 200D
ZWNJ := 200C

```
Shifter :=
      (?<= ConsGroupStrong FinalCoengNonRo?) 17CA ZWNJ (?= VowelAboveContext)
    | (?<! ConsGroupStrong FinalCoengNonRo?) 17C9 ZWNJ (?= VowelSignAboveContext)
    | [17C9 17CA]
ConsGroupStrong :=
      ConsStrong Robat? CoengNonBa{0,2}
    | ConsNonBa Robat? (CoengStrong CoengNonBa? | CoengNonBa CoengStrong)
ConsStrong :=
    [1780-1783 1785-1788 178A-178D 178F-1792
     1795-1797 179E-17A0 17A2]
CoengStrong := Conjoiner ConsStrong
CoengNonBa := Conjoiner ConsNonBa
ConsNonBa := [1780-1793 1795-17A2]

Vowels :=
      17C1 [17BC-17BD]? [17B7 17B9-17BA]?
    | [17C2-17C3]? [17BC-17BD]? [17B7-17BA] 17B6
    | [17C2-17C3]? [17BB-17BD]? 17B6
    | [17BE [17BC-17BD]? 17B6?
    | [17C1-17C5]? 17BB (?! [17D0 17DD])
    | [17BF 17C0]
    | [17C2-17C5]? [17BC-17BD]? [17B7-17BA]?

Modifiers := Modifier1 Modifier2?
Modifier1 :=
      [17C6 17CB 17CD-17CF 17D1]
    | (?<! 17BB [17B6 17C4 17C5]?) [17D0 17DD]
Modifier2 := [17C6 17CB 17CD-17D1 17DD]
Final := [17C7 17C8]
```

For example, the common word ខ្ញុំ khnyom “I” is composed of the following three linguistic elements: (1) consonant character ខ *kha* as ~~ʙ~~ Base; (2) subscript consonant sign ◌្ *coeng nyo* as ~~ꜱ~~ Coeng; and (3) dependent vowel ~~sign~~ ◌ុំ *om* as ~~ø~~ a linguistic vowel. In the Unicode Standard, *coeng nyo* and *om* are further decomposed, and the whole word is represented by five coded characters.

In figure 16-3, change the fourth example to reflect the new encoding of final coeng:

ហើរ្យ ha + ~~*oe + coeng + yo*~~ *coeng* + ZWJ + *yo* + *oe* [haəi] “already”

At the end of the subsection, after figure 16-3, insert:

The following Khmer characters do not participate in orthographic syllables, some because their use is discouraged, others because they're standalone letters, numbers, symbols, or punctuation:

[17A3 17A4 17B4 17B5 17D3-17DC 17E0-17E9 17F0-17F9 19E0-19FF]

② Update the subsection "Consonant Shifters"

Append the following sentence to the first paragraph to describe the special behavior of consonant shifters applied to *ba*:

The second of these examples shows an exceptional use of *muusikatoan*: Applied to *ba*, it converts the consonant to *pa* without changing the series, and in this use it can be rendered with the alternative subscript glyph. On the other hand, *triisap* when applied to *ba* to change its series to *bo* never changes to the alternative subscript glyph.

Update the first sentence of the second paragraph as follows:

~~Although the consonant shifter in handwriting may be written after the subscript, the~~ The consonant shifter should always be encoded ~~immediately following the base consonant, except when it is preceded by U+200C zero width non-joiner.~~ after the initial consonant group, which may consist of base, robat, and coengs, and before any vowels or modifiers.

Update the sample strings for this paragraph:

ខ្ញុំ mo ~~+ *muusikatoan*~~ + *coeng* + *ngo* + *muusikatoan* + *ai* [mŋai] "one day"

ខ្ជួតៗ mo ~~+ *triisap*~~ + *coeng* + *ha* + *triisap* + *ae* + *ta* + *lek too* [mhɛ̀ːtmhɛ̀ːt] "bland"

Update the following paragraph, eliminating an inconsistency within it:

If either *muusikation* or *triisap* needs to keep its superscript shape (as an exception to the general rule that states other superscripts typically force the alternative subscript glyph for either character), U+200C ZERO WIDTH NON-JOINER should be inserted ~~before~~ after the consonant shifter to show the normal glyph for a consonant shifter when the general rule requires the alternative glyph. In such cases, U+200C ZERO WIDTH NON-JOINER is inserted before the vowel sign, as shown in the following examples:

Delete the first sample string for this paragraph, because it uses the combination of *ba* with *triisap*, in which

triisap is never rendered with the alternative subscript glyph, and the use of ZERO WIDTH NON-JOINER is therefore inappropriate.

Update the second sample string for this paragraph:

> ប្រតឹងអឺៈ *ba* + *coeng* + *ro* + *ta* + *yy* + *ngo* + *qa* + *triisap*+ ZWNJ ~~*+ triisap*~~ + *y* + *reahmuk* [prɔtəːŋʔɯh] “urgent, too busy”

③ Update other sample strings to reflect the new orthographic syllable structure

In the discussion of subscript consonant signs for a closing consonant on page 679, change the examples to reflect the new encoding of final coeng:

> ទ្ងំ to + ~~*aa + nikahit + coeng + ngo*~~ *coeng* + ZWJ + *ngo* + *aa* + *nikahit* [tɛ̀əŋ] “both” (= ទាំង) (≠ *ទ្ងាំ [tŋɔ̀əm])

> ហើ្យ ha + ~~*oe + coeng + yo*~~ *coeng* + ZWJ + *yo* + *oe* [haəi] “already” (= ហើយ) (≠ *ហើ្យ [hjaə])

EBNF Enhancements

To fully enable the grammar used in the above proposal, section A.2 Extended BNF of the Unicode Standard needs to be enhanced with the look ahead and look behind assertions taken from the Perl Compatibile Regular Expressions package:

- `(?= x)` – look ahead: if followed by x
- `(?<= x)` – look behind: if preceded by x
- `(?! x)` – negative look ahead: if not followed by x
- `(?<! x)` – negative look behind: if not preceded by x