L2/09-358

2009-10-22

Title:	Discussion document for polishing Koranic support in Unicode
Action:	For discussion at UTC and by experts. No action is requested.
Authors:	Roozbeh Pournader
Date:	2009-10-22

Introduction

Although we are almost there, Unicode still has a few ambiguities and missing characters for properly encoding the modern day representation of Koranic text. Experts have different opinions on how this should be fixed. This document tries to point to the existing problems, as of Unicode 5.2, and make some recommendations.

Recommendation

- Change the properties of U+06DE ARABIC START OF RUB EL HIZB from that of a combining mark to a normal spacing symbol (properties could be copied from U+06E9 ARABIC PLACE OF SAJDAH);
- Encode four missing characters (three open *tanweens* and one combining version of small *waw*);
- Suggest a solution for missing carriers and horizontally-stacking *harakat* in the text of the standard (for example, recommend the use of U+0640 ARABIC TATWEEL and U+00A0 NO-BREAK SPACE as carriers).

Background

At a recent talk by Thomas Milo at the Unicode and Internationalization Conference 33, titled “The Unicode-based Koran: a Conflict Between Calligraphic Tradition and Computer Typography” brought this issue back into the author’s agenda. At the talk, Thomas Milo explained some of the existing problems, and proposed his solution based on his ideal Arabic character model. Unfortunately, changing the Unicode Arabic model for basic characters, like U+0621 ARABIC LETTER HAMZA, may be impossible this late in the standardization process.

The author is using the opportunity to bring into attention the other issues he has found with Unicode Koranic support, and also try to explain the still-existing point of frustration of people who try to encode Koranic text.

Rub El Hizb Symbol

To help the devout Muslims recite the whole Koran during the holy month of Ramadan, the Koran is divided into thirty parts. These parts, called *juz’* (جزء), are different from the traditional 114 *surah*s (سورة): they divide the Koran into roughly equal amounts of text, so the devout can spend almost equal amount of time each day for their recitations. For further help timing, each *juz’* is further divided into two *hizb*s (حزب), and each *hizb* into four *rub’-al-hizb*s. In printed Korans, this sectioning is typically specified in the margins, with a marker inside running text to point the exact location of one *rub’-al-hizb* ending and another starting (see Figure 1).

The Unicode character U+06DE ARABIC START OF RUB EL HIZB is that in-text marker. In printed Korans, is appears in running text by itself, usually adjacent to an end-of-ayah marker. The rendering behavior is the same as U+06E9: the character just sits there and does not interact with other adjacent text.

But in Unicode, U+06DE is specified to be a combing character. The author believes this to be a simple mistake,

since it was encoded next to U+06DD ARABIC END OF AYAH (which was originally a combining character too). Looking at the reference glyph for U+06DE somehow confirms this too: there is a dotted circle in the glyph, but there is no way something could fit in there.

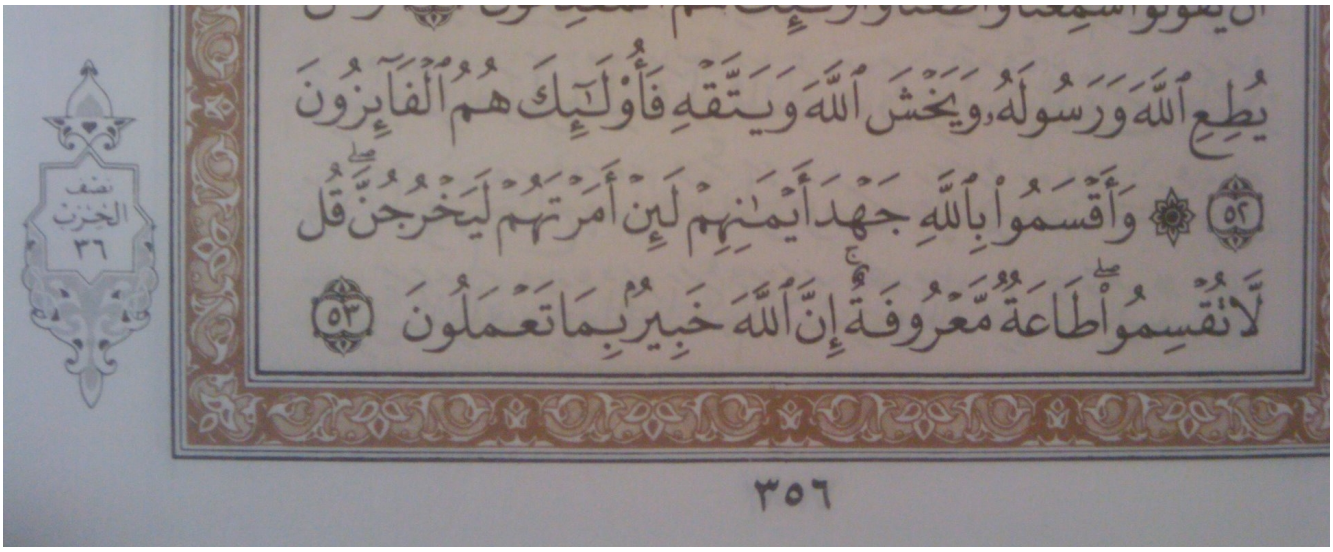

Figure 1. Sample from a Koran printed in Iran. The character U+06DE ARABIC START OF RUB EL HIZB is visible at the beginning of line -2, after an End of Ayah marker. The large decoration at the left margin of the page helps readers find the inline marker easier. The text in the center of the decoration reads "half of *hizb* 36" (نصف الحزب ٣٦), indicating that the reader is at approximately 71/120 of the whole text of the Koran.

Missing characters

For encoding the most common modern form of published Koranic text, four characters are missing. Three are open forms of *tanween*, which mark a pronunciation difference with normal tanweens, already encoded at U+064B..064D. The other is a combining companion of U+06E5 ARABIC SMALL WAW, needed for encoding some words (compare to the pair U+06E6 ARABIC SMALL YEH and U+06E7 ARABIC SMALL HIGH YEH).

The three open *tanween*s have been proposed at least twice, by Thomas Milo in L2/01-325 and by Jonathan Kew in L2/02-275. There is no indication that the first document was ever on the UTC table, and although the second document appears on the agenda for UTC #92, there is no mention of it in the minutes. These may not have ever been discussed in the UTC.

There is also a third document which appears like a Unicode proposal but was never submitted for consideration of the committee. It's Arabeyes's "Proposal to add four Arabic characters to the BMP of the UCS and fix four characters" written by Mohammad Yousif and Nadim Shaikli, publicly available at http://arabeyes.org/~nadim/tmp/unicode_quran_prop.pdf

The open *tanween*s point toward a different pronunciation of *tanween* in Koranic text. While a normal *tanween* would be pronounced as [an], [in], or [un] with a clear [n] sound (called *iẓhār*), an open *tanween* will hint toward either nasalization of the [n] sound (called *ikhfā'*), or its total disappearance, geminating the next consonant (called *idqām*).

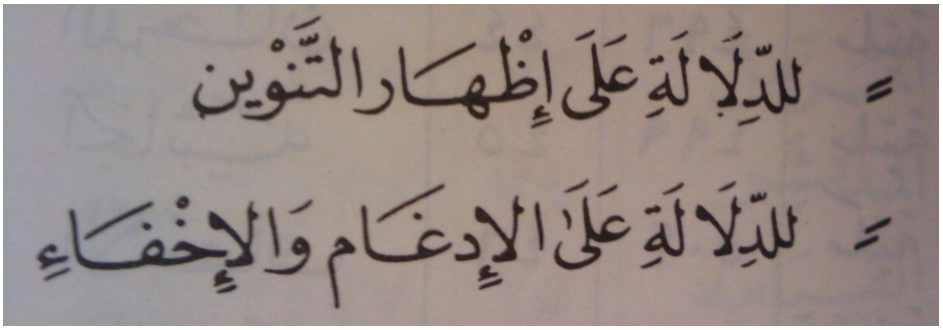

Figure 2. Symbol legend in Arabic language, appearing at the end of a Koran published in Iran. A normal *tanween* is shown first, saying it marks *iẓhār* (إظهار). Then an open tanween is shown, saying it makrs *idqām* (إدغام) and *ikhfā'* (إخفاء).

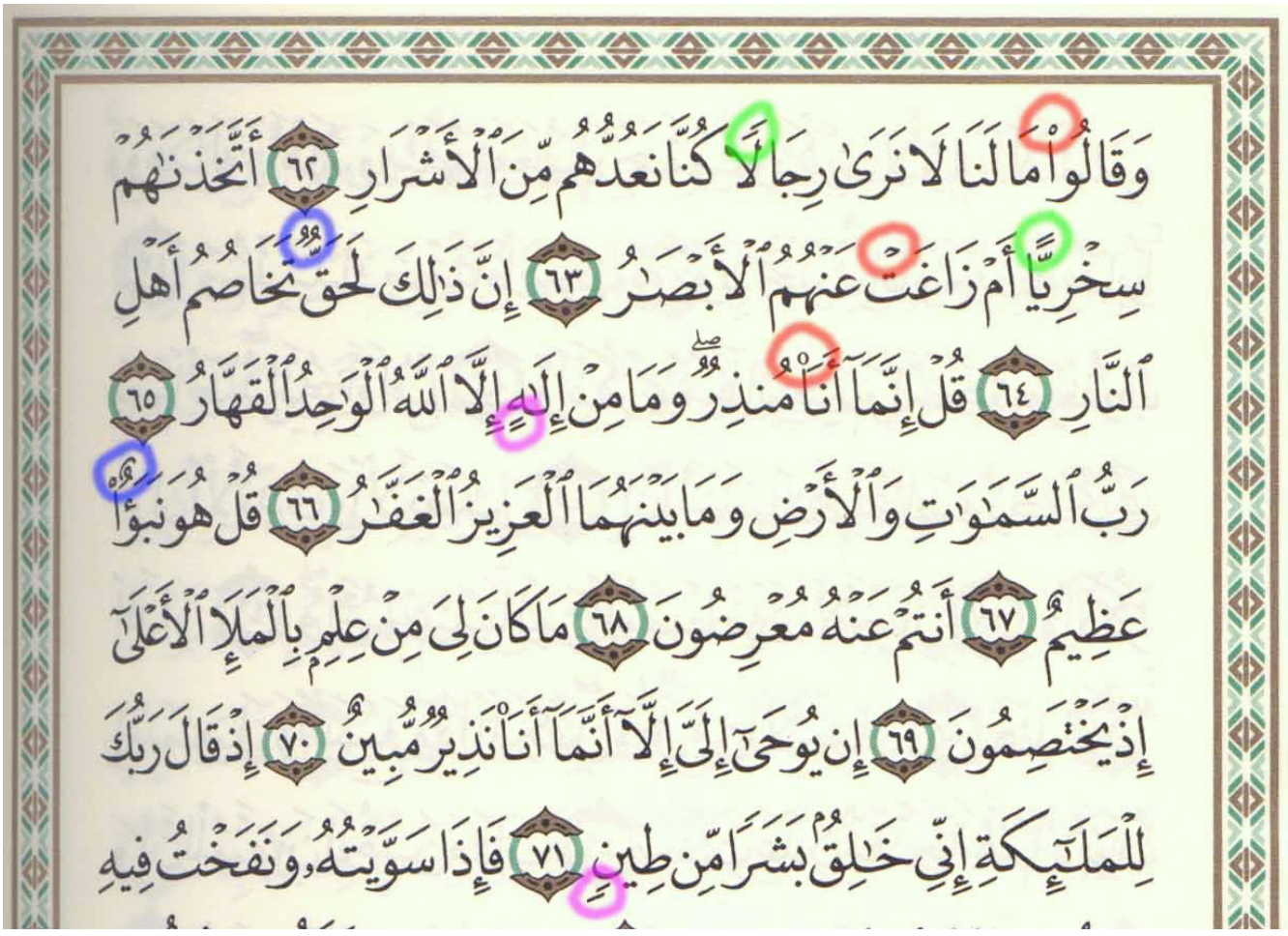

Figure 3. Examples of the three open *tanween* characters as opposed to normal *tanween*s, from L2/02-275. Note that a intelligent rendering software, in order to determine the proper shape of the Kasratan on the last line, would need to skip over at least three characters that compose the maker: an End of Ayah character and two digits.

Theoretically, the shape of the *tanween* could be determined by the next pronounced consonant, and advanced reciters even learn the rules to determine it themselves. But determination of the next pronounced consonant would need skipping over various characters to be able to determine the next pronounced consonant. The skipped-over characters may include spaces, unpronounced *alef*s (possibly with some combining marks), symbols like START OF RUB EL HIZB and PLACE OF SAJDA, END OF AYAH symbols with their pack of following digits, page breaks, or *sura* breaks. The author believes that while is achievable in software, it is overkill for text rendering engines who want to reflect the actual textual content of Koranic text.

The other missing character, ARABIC SMALL HIGH WAW, is exemplified in the Arabeyes document, although not as a new character. The character appears mid-word in Koranic text, as shown in Figure 4 (left-hand). U+06E5 cannot be used for this purpose, as it will break the skeleton of the word.

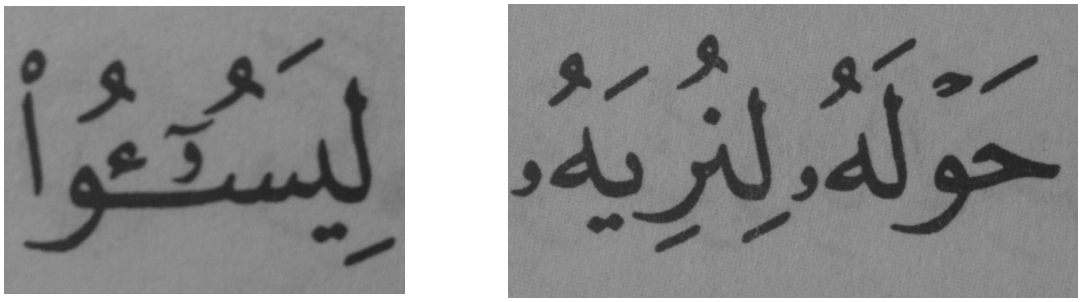

Figure 4. On the right side, there are two examples of spacing U+06E5 ARABIC SMALL WAW at the end of each word. On the left side, there is a combining version (applied to either a *seen*, or a *tatweel*), not encoded yet. Also note the visual difference between ARABIC SMALL HIGH WAW (indicating a long [u:] sound) and a normal U+064C ARABIC DAMMA (indicating a short [u]).

Missing carriers and horizontally-stacking harakat

Modern printed versions of the Koran prefer keeping the original word skeletons intact. For example, the left-hand part of Figure 4, the word *liyasu'u*, will probably be written as something like ليسؤوا in a modern Arabic orthography for modern text. Not changing the skeleton, is keeping with the tradition of writing the Koran, and some Muslim consider the original skeleton shapes the only authentic shapes, as the dots and the *harakat* are later additions and subject to interpretations and disagreements. (More examples can be seen in Figure 5.)

But keeping the skeleton intact has required the fully-marked-up Korans to inserts marks in places where a mark usually doesn't appear in modern Arabic text. In the example word, after the *seen*, a reciter will read a *damma*, a small high *waw*, a (small high?) *madda*, a *hamza* (above?), and another *damma* before arriving at the next letter, a *waw*. There are three visual stacks of *harakat* visible. The first stack has the first *damma*, the second stack has the small high *waw* and *madda*, and the third stack consists of a *hamza* and the second *damma*.

There are different approaches to encode that text:

1. Encode all the characters as combining marks over *seen*, and say that the three stacks happen because of stylistic reasons. (This will result in over-stacking and ineligibility in Koranic texts rendered by normal software, and may be unachievable because of the effect of normalization on Koranic text, since normalization may change the order of *harakat*). Sample string: <..., seen, damma, small high waw, small high madda, hamza above, damma, waw, ...>
2. Specify that for breaking stacks, a special character like U+034F COMBINING GRAPHEME JOINER

should be used. (This will solve the normalization issue of case 1, but will still result in over-stacking in non-specialized sofware.) Sample string: <..., seen, damma, **CGJ**, small high waw, small high madda, **CGJ**, hamza above, damma, waw, ...>

3. Change the behavior of characters like U+0621 ARABIC LETTER HAMZA etc. to support such dual-joining behavior in Koranic text, or encode new characters for *hamza* etc. with the new behavior. Sample string: <.., seen, damma, **(dual-joining) high waw**, small high madda, **(dual-joining?) hamza**, damma, waw, ...>
4. Specify that a U+0640 ARABIC TATWEEL or U+00A0 NO-BREAK SPACE should be used to carry such harakat. (This is not semantically corrrect, but has the least changes of the existing Unicode Arabic model, and existing software could continue to display in an acceptable rendering.) Sample text: <..., seen, damma, **tatweel**, small high waw, small high madda, **tatweel**, hamza above, damma, waw, ...> An example needing NBSP could be seen in Figure 5.

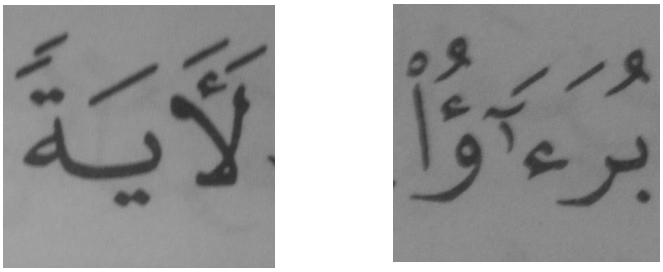

Figure 5. On the right side, U+0670 ARABIC LETTER SUPERSCRIPT ALEF (a combining character) is seen standing alone between a *hamza* and a *waw-hamza* (this would need the use of NBSP in the fourth approach). On the left side, a combining *hamza* is seen between a *lam* and an *alef* forming a ligature. If the fourth approach is taken, the text will be rendered as something like لَـٰٓـَٔـَٔـَةً (acceptable) with normal Unicode-complying software, while Koranic software can create a ligature.

Acknowledgments

The author wishes to thank Thomas Milo, Adil Allawi, and the Arabeyes free software community for fruitful discussions. Some samples are taken from papers and documents by Thomas Milo and Jonathan Kew.