

Proposed Draft: Unicode Technical Report  
UNICODE ARABIC MARK ORDERING ALGORITHM

Authors	Roozbeh Pournader ( <a href="mailto:roozbeh@unicode.org">roozbeh@unicode.org</a> ), Bob Hallissy ( <a href="mailto:bob_hallissy@sil.org">bob_hallissy@sil.org</a> ), Lorna Evans ( <a href="mailto:lorna_evans@sil.org">lorna_evans@sil.org</a> )
Date	2017-07-26
This version	
Previous version	<a href="http://www.unicode.org/L2/L2014/14127-arabic-marks-order.pdf">http://www.unicode.org/L2/L2014/14127-arabic-marks-order.pdf</a>
Latest version	
Revision	3

### Summary

*The Unicode Arabic Mark Ordering Algorithm (UAOA) describes an algorithm for determining correct rendering of all canonically equivalent sequences the same way.*

### Status

This document, based on [L2/14-127 Standardizing the order of Arabic combining marks](#) by Roozbeh Pournader, is intended to clarify the rationale behind the algorithm, make the algorithm easier to understand, and provide additional examples of its use.

We have been asked to make this a UTR. If UTC chooses, it could become a UTS or UAX later, depending on the level of acceptance.

## Contents

- 1. [Overview](#)
- 2. [Background](#)
- 3. [UAOA Details](#)
  - 3.1 [Modifier Combining Marks \(MCM\)](#)
  - 3.2 [UAOA](#)
- 4. [Demonstrating UAOA](#)
  - 4.1 [Artificial Test Case](#)
  - 4.2 [Exceptions](#)
  - 4.3 [Example Cases](#)
- 5. [Further Explanations](#)
  - 5.1 [Use of NFD and not NFC](#)
  - 5.2 [Sukun alternate forms](#)
  - 5.3 [Shadda](#)
  - 5.4 [Kasra and kasra-like characters](#)
  - 5.5 [Maddah and Small High Madda](#)
  - 5.6 [Dotted circles](#)
  - 5.7 [Other Characters](#)
  - 5.8 [Other uses for UAOA](#)
  - 5.9 [Best combining classes for yet-to-be-encoded combining marks in Arabic](#)
- 6. [References](#)

## 1. Overview

The combining classes of Arabic combining characters in Unicode are different than combining classes in most other scripts. They are a mixture of special classes for specific marks plus two more generalized classes for all the other marks. This has resulted in problems for sequences with multiple combining marks since Unicode 2.0.

UAOA, described herein, is the proposed implementation to deal with the problems of special combining classes. This algorithm is the simplest algorithm that keeps user expectations, stays within the stability requirements of Unicode, correctly renders data in normalized forms, renders all canonically equivalent sequences the same way.

## 2. Background

Rules and recommendations for the correct display of combining marks are discussed in a number of places in the Unicode standard, including sections 5.13, 7.9, and 9.2 [8]. Some general principles include:

- Canonically equivalent sequences should display the same
- Combining marks from the same combining class are normally displayed using the *inside-out rule*, i.e., from the base outward

- Combining marks from different combining classes (other than ccc=0) may be re-ordered with respect to each other if that helps to achieve the desired display.

In Unicode, the Arabic script combining marks include eleven different non-zero canonical combining class values, as shown in Table 1. When a combining character sequence includes marks from more than one of these classes, the rendering system has to determine a sequence (i.e. a display order) in which to position these marks on the base character.

While it might be tempting to just use NFC or NFD, neither of these will yield what Arabic readers expect. For one example that will be easily understood by all readers of Arabic script, given a combining character sequence including a shadda (ccc=33) and damma (ccc=31), NFC and NFD will move the *damma* before the shadda at which point the default inside-out rule would place the *shadda* above the *damma*, which is incorrect.

A further complication is that while some cases are obvious to readers of languages written with Arabic script and thus will likely get the same display from various rendering implementations, many of the combining marks, especially those with ccc=220 and ccc=230, are not commonly understood, and different rendering implementations have made different decisions regarding display order, resulting in inconsistent behavior between one system and another.

UAOA defines a method to reorder Arabic combining marks in such a way that, when subsequently rendered according to the inside-out rule, it achieves a number of goals:

- Display combining marks in the order in which most users would expect
- Provide for overriding the display order for exceptional cases
- Display canonically equivalent sequences in the same way.

Canonical Combining Class (ccc) value:	Combining marks in this class
0	Combining grapheme joiner
27	fathatan, open fathatan
28	dammatan, open dammatan
29	kasratan, open kasratan
30	fatha, small fatha
31	damma, small damma
32	kasra, small kasra
33	shadda

34	sukun
35	Superscript alef
220	All other below combining marks
230	All other above combining marks

**Table 1: Canonical combining class values for marks used in Arabic script**

### 3. UAOA details

The algorithm starts by reordering combining marks according to one of the Unicode Normal forms, and then make adjustments by moving certain marks closer to the base.

#### 3.1 Modifier Combining Marks (MCM)

For use by this algorithm, we define a group of combining marks called “Modifier Combining Marks” (MCM). These are combining characters that are normally used to modify the base character before them, and because of that, should normally be rendered closer to the base character than *tashkil* (supplementary diacritics, including vowels). The MCM characters are not officially classified as *ijam* (consonant pointing/nukta, etc) in the Unicode Standard, but they are usually very close to *ijam* in users’ perception of them.

The suggested characters to include in the MCM are:

- U+0654 ARABIC HAMZA ABOVE
- U+0655 ARABIC HAMZA BELOW
- U+0658 ARABIC MARK NOON GHUNNA
- U+06DC ARABIC SMALL HIGH SEEN
- U+06E3 ARABIC SMALL LOW SEEN
- U+06E7 ARABIC SMALL HIGH YEH
- U+06E8 ARABIC SMALL HIGH NOON
- U+08F3 ARABIC SMALL HIGH WAW

After an appropriate review period, the list of characters included in the MCM will be considered “fixed.” Adding a character to the MCM that is already in Unicode could change the rendering of data that assumes the implementation of UAOA. If new characters are encoded in Unicode, those new characters could be added to the MCM at the time of encoding if needed.

## 3.2 UAOA

**Input:** A Combining Character Sequence (D56) containing one or more Arabic combining marks.

**Output:** A canonically equivalent Combining Character Sequence reordered for rendering using inside-out stacking.

Steps:

1. Normalize the input to NFD
2. Within the result, for each maximal-length substring, *S*, of non-Starter (D107) characters, re-order as follows:
  - a. Move any shadda characters (ccc=33) to the beginning of *S*.
  - b. If a sequence of ccc=230 characters begins with any MCM characters, move those MCM to the beginning of *S* (before any characters with ccc=33).
  - c. If a sequence of ccc=220 characters begins with any MCM characters, move those MCM to the beginning of *S* (before any MCM with ccc=230).

## 4. Demonstrating UAOA

### 4.1 Artificial Test Case

This is a very artificial test case, just to demonstrate the algorithm:



## 4.2 Exceptions

The default order of the UAOA may not be perfect, and someone may want to modify the order of some marks. When the UAOA does not result in a preferred order, U+034F COMBINING GRAPHEME JOINER (CGJ) should be used to request a different order.

## 4.3 Example Cases (rationale for including characters in the MCM)

The following examples demonstrate why each of these characters should be included in the MCM.

U+0654 ARABIC HAMZA ABOVE and U+0655 ARABIC HAMZA BELOW

TUS [8], Chapter 9, page 387 discusses the use of combining hamza above and below.

فَأُولَئِكَ

Example 1 [6, page 9, end of line 5]

In Example 1, UAOA puts a *damma* over a *hamza above*. If an orthography needs to put the *hamza above* over the *damma*, the text should be encoded as <damma, CGJ, hamza above>.

UAOA puts the *kasra* below a *hamza below*. If an orthography needs to put the *hamza below* under the *fatha*, the text should be encoded as <fatha, CGJ, hamza below>.

#### U+0658 ARABIC MARK NOON GHUNNA

Regarding this mark Kew ([1] page 2) says “The ARABIC NASALIZATION MARK is considered equivalent to a “nukta”, *as it is a modifier that binds tightly to the underlying letter;*” (italics added to express emphasis). “ARABIC NASALIZATION MARK” was later renamed to “ARABIC MARK NOON GHUNNA”.

#### U+06DC ARABIC SMALL HIGH SEEN and U+06E3 ARABIC SMALL LOW SEEN

SMALL HIGH SEEN was put in MCM because most Quranic orthographies use the character as an MCM only. The orthographies that use the character the other way will need to use a CGJ in those cases.

Example 2a [4]

Example 2b [4]

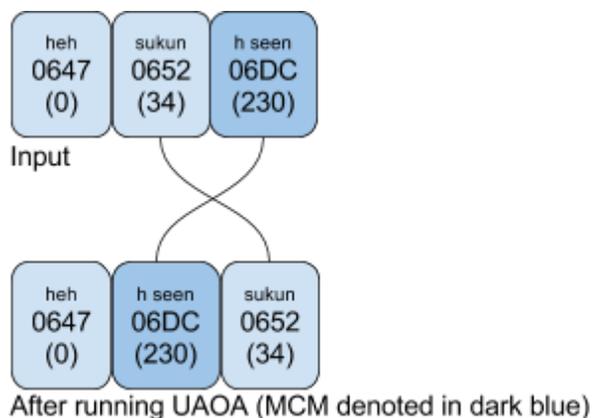
In Example 2a, the *small high seen* is rendered below the *sukun*, while in Example 2b, it is rendered over it. The examples are indeed from the same document (Al-Hilâlî and Khân 1996), just two pages away. The *small high seen* has different roles: in Example 2a it is a hint that the base letter, *sad*, should be pronounced as if it was a *seen*; in Example 2b, it is a pause-related hint.

Example 2a (characters and properties):

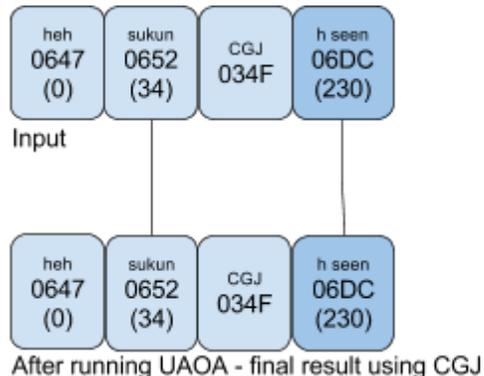
sad 0635 (0)	h seen 06DC (230)	sukun 0652 (34)
--------------------	-------------------------	-----------------------

Running UAOA on this string does not result in any changes.

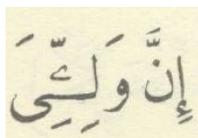
Example 2b (characters and properties):



Running UAOA on the string in Example 2b resulted in an undesired change. It puts a *sukun* over a *seen above*. If an orthography needs to put the *seen above* over the *sukun*, the text should be encoded as <sukun, CGJ, seen above>.



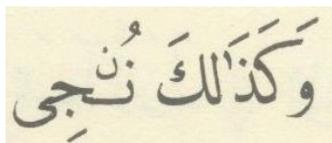
U+06E7 ARABIC SMALL HIGH YEH



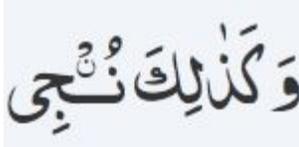
Example 3 [3, page 9, line 11]

In Example 3, UAOA puts a *shadda* over a *small high yeh*. If an orthography needs to put the *small high yeh* over the *shadda*, the text should be encoded as <shadda, CGJ, small high yeh>.

U+06E8 ARABIC SMALL HIGH NOON

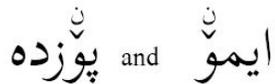


Example 4a [3, page 10, (21.88)]



Example 4b [7]

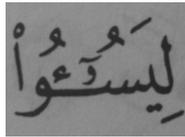
Example 4a demonstrates a common reading with just a small high noon. However some readings, such as in Example 4b, have a *sukun* over a *small high noon*. UAOA puts a *sukun* over a *small high noon*. If an orthography needs to put *small high noon* over *sukun*, the text should be encoded as <sukun, CGJ, small high noon>.



Example 4c

Example 4c shows a practical orthography that uses *small high noon* for nasalization. It is theoretically possible for a vowel to appear above the small high noon in this practical orthography.

U+08F3 ARABIC SMALL HIGH WAW



Example 5 [5, page 4]

In Example 5, UAOA puts a *maddah* over a *small high waw*. This indicates the *small high waw* should stay closer to the base than *maddah*.

## 5. Further Explanations

### 5.1 Use of NFD and not NFC

NFD assures sequences such as <superscript alef, madda> always results in the same ordering, independent of the base letter. If the algorithm used NFC, the sequence <alef, superscript alef, madda> would have resulted in a different order than <lam, superscript alef, madda> (since NFC composes <alef, madda> to <alef-with-madda-above>).

### 5.2 Sukun alternate forms

There are three *sukun-like* shapes encoded at U+06DF..U+06E1 that are used in some Quranic orthographies to denote different things which are not always a *sukun*. Fortunately, their canonical combining class is 230, so the ordering in the presence of other combining

marks is not affected. Unfortunately, since the combining class for the *sukun* is 34, they will *not* be treated like a normal *sukun* in all cases. This does not create a real problem, as users who create the data simply have more flexibility with the alternate *sukun* characters than with the normal *sukun*. UAOA should not make them equivalent, as that would make the algorithm unnecessarily complex and make the usage of CGJ more frequent.

### 5.3 Shadda

Some may question why *shadda* has a special place in the UAOA. The combining class for shadda is higher than most vowels (ccc=33); however, it should be closer to the base than the vowels.

### 5.4 Kasra and kasra-like characters

UAOA is able to handle the special ligation of *kasra* and *kasra-like* characters which are ligated with a *shadda* or *hamza* in some styles and appear just below them instead of below the base letter; they still logically follow the *shadda* or *hamza*.

### 5.5 Maddah and Small High Madda

In an early stage in the development of this document there was discussion about the difference in the identity of U+0653 ARABIC MADDAH ABOVE and U+06E4 ARABIC SMALL HIGH MADDHA and if the normal madda should belong in MCM. After discussion with Arabic script experts, a decision was made to not include either of them in the MCM, and consider the two characters size variants of each other.

### 5.6 Dotted circles

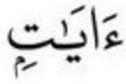
Some rendering engines will insert a dotted circle for what it understands to be an invalid sequence. This is a problem in Arabic script because something that appears invalid may actually be valid text in some lesser known orthography of a minority language or in the Quran. (For example, Microsoft Windows's text rendering engine, described in Microsoft Typography 2014 [2], appears to disallow combination of certain Quranic marks that are known to appear with each other in the Quran.)

Such spell-checking processes are best implemented at a higher level than a rendering engine. Also, a dotted circle insertion algorithm that would display all canonically equivalent sequences the same way is hard to design and the result may be counter-intuitive for its users.

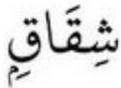
However, the algorithm could easily be extended for implementations that may wish to insert such marks (such as the one described Microsoft Typography 2014 [2]) by applying the algorithm first and then inserting the dotted circles.

## 5.7 Other characters

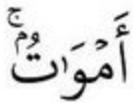
Some might ask why *meem above* (ccc=230) and *meem below* (ccc=220), and other similar characters, are not in the UAOA.



Example 6a [6, page 11]



Example 6b [6, page 21]



Example 6c [6, page 19]

Examples 6a-6c show that the *meem* is normally kept after *fatha*, *kasra* or *damma*.

## 5.8 Other uses for UAOA

The algorithm is very useful for backspacing, when there is no external information available about the original order in which the text was entered.

## 5.9 Best combining classes for yet-to-be-encoded combining marks in Arabic

When new combining marks are encoded, 220 should be used for below marks and 230 for above marks. In the special cases where an alternative version of the basic *tashkil* is encoded, the same combining class as the *tashkil* could be used, but extreme care should be taken.

## 6. References

1. Kew, Jonathan, 2002. *Bidi committee consensus on Arabic additions from L2/01-425* (<http://www.unicode.org/cgi-bin/GetMatchingDocs.pl?L2/02-061> accessed 1 May 2017)
2. Microsoft Typography 2014. "Developing OpenType Fonts for Arabic Script." <http://www.microsoft.com/typography/OpenTypeDev/arabic/intro.htm> (accessed 27 Jul 2017).

3. Milo, Thomas. 2005. *Annotations to the printing of the 1924 Azhar Qur'an* (U+0670, U+06D6..U+06DB, U+06DD..U+06DF, U+06E0..U+06ED) (<http://www.unicode.org/cgi-bin/GetMatchingDocs.pl?L2/05-151> accessed 1 May 2017).
4. Muhammad Taqī-ud-Dīn Al-Hilālī and Muhammad Muhsin Khān (translators) 1417 AH (=1996 CE). *The Noble Qur'an: English Translation of the meanings and commentary*. King Fahd Complex For The Printing of The Holy Qur'an. ISBN 9960-770-15-X.
5. Pournader, Roozbeh. 2009. Proposal to encode four combining Arabic characters for Koranic use. (<http://www.unicode.org/cgi-bin/GetMatchingDocs.pl?L2/09-419> accessed 2 May 2017)
6. Quran example. Al-Baqarah. <https://archive.org/stream/quran-pdf/002%20-%20Al-Baqarah> (accessed 27 Jul 2017).
7. Quran example. <http://www.dailyayat.com/al-ambiya/21/88> (accessed 27 Jul 2017).
8. The Unicode Consortium. 2016 *The Unicode Standard, Version 9.0.0*, (Mountain View, CA: The Unicode Consortium, 2016. (<http://www.unicode.org/versions/Unicode9.0.0/>))



