ISO/IEC JTC1/SC2/WG2 N3061
L2/06-
2006-03-27
L2/06-129

Universal Multiple-Octet Coded Character Set
International Organization for Standardization
Organisation Internationale de Normalisation

Doc Type: **Working Group Document**
Title: **Supporting Discussion for the Encoding of seven additional Myanmar Characters in the UCS**
Source: **Martin Hosken**
Status: **Individual contribution**
Action: **For consideration by JTC1/SC2/WG2 and UTC**
Date: **2006-03-27**

This document provides further supporting information to N3043R (L2/06-077R) in response to considerable discussion on the topic within the UTC.

Transcoding

If N3043R is adopted, it will necessitate the transcoding of existing Unicode text. In the normal course of events this would be considered prohibitively expensive and would require an alternative solution to be found. But due to the low adoption of Unicode within Myanmar and for the script, with perhaps only 1 commercial solution on the market, transcoding is a possibility at this point.

Extent of the Problem

A quick scan of the internet has shown that there are a few documents that are stored in conformant Unicode 4.1. There are a vast majority of texts either stored in legacy or in non-conformant Unicode that will need to be transcoded regardless of which model Unicode ends up with. Almost all of those conformant documents have been created by the very implementors that are working on implementations of Myanmar Unicode and are wanting to see these seven additional characters added. For the most part these documents have been transcoded from existing legacy documents, and so can be transcoded again with a modified transcoder. Alternatively the content authors have stated that they are happy to transcode. Specifically, the authors of the my.wikipedia and the Judson Bible pages have both stated that they are happy to transcode.

There is one constraint on all this transcoding option and that is that the final encoding for Myanmar Burmese must be resolved quickly. The decision made in April will be key to this. The Myanmar computing community is ready to distribute Unicode based implementations now and they are only holding back to allow ISO to make the necessary additions to the UCS. If these do not happen,, they will have to go ahead with what will probably be a variety of solutions handling edge cases in difference and possibly incompatible ways, and the opportunity for transcoding relatively small amounts of data will have passed. In addition, an unchanged or undecided encoding model will make it very difficult to encode the minority extensions needed for the Myanmar script. After this, there will be no possibility of change and transcoding will not be an option. This is the final call on this topic. Delaying a decision is not an option as far as the Myanmar computing community is concerned.

An Approach to Transcoding

The following Perl based code fragment suggests an approach to transcoding and shows the complexity of what is involved:

```
s/\x{1039}\x{200C}/\x{103A}/og;        # asat
s/\x{1039}\x{101A}/\x{103B}/og;        # medial ya
s/\x{1039}\x{101B}/\x{103C}/og;        # medial ra
s/\x{1039}\x{101D}/\x{103D}/og;        # medial wa
s/\x{1039}\x{101F}/\x{103E}/og;        # medial ha
s/\x{101E}\x{1039}\x{101E}/\x{103F}/og;# great sa
```

```
if ($type eq 'UTN#11')                # discussion of this to follow
{
        s/\x{1004}\x{1039}/\x{1004}\x{103A}\x{1039}/og;
        s/\x{200D}//og;               # remove kinzi blocker
}
elsif ($type eq 'NLP')
{
        s/\x{1004}\x{1039}\x{200D}/\x{1004}\x{103A}\x{1039}/og;
}
elsif ($type eq 'Kai')                # from what I can glean
{
        s/\x{1004}\x{1039}/\x{1004}\x{103A}\x{1039}/og;
        s/\x{1004}\x{200B}\x{1039}/\x{1004}\x{1039}/og;
}
# tall a rules
s/([\x1001}\x{1002}\x{1004}\x{1012}\x{1015}\x{101D}]\x{1031}?)\x102C/$1\x{102B}/og;
s/([\x{1002}\x{1012}]\x{1039}[\x{1000}-\x{1021}]\x{103D}?\x{103F}?\x{1031}?)\x102C/$1\x{102B}/og;
```

Caveat emptor: due to time constraints this code has not been tested. The conversion commands are executed in this order so that the contexts in later rules are simplified. In addition, in comparison to the complexity of conversion from a legacy glyph based encoding, this transcoding is like a walk in the park.

One of the confusions with the current Myanmar model is the handling of Kinzi. There are three different approaches that I know of to handling kinzi. “UTN#11” is the approach taken in UTN#11. “NLP” is the approach taken by the Myanmar Language Commission and Natural Language Processing Research Lab and “Kai” is the approach taken by Ka'onohi Kai of Xenotypetech in his commercial implementation. This list is by no means complete. There are other solutions to the Kinzi problem according to the particular implementor. Not all are listed here.

Glyphs	***UTN#11***	***NLP***	***Kai***
◌ိ◌်	U+1004 U+1039 U+101D U+1031	U+1004 U+1039 **U+200D** U+101D U+1031	U+1004 U+1039 U+101D U+1031
◌◌	U+1004 **U+200D** U+1039 U+101D U+1031	U+1004 U+1039 U+101D U+1031	U+1004 **U+200B** U+1039 U+101D U+1031

This highlights one of the existing problems with the Myanmar model in that there is no standard on how to handle the Kinzi issue. Unicode has failed to give a normative answer to this question and so different solutions have emerged. So even without the changes proposed in N3043R, somebody is going to have to transcode their data! N3043R at least resolves this confusion.

Alternative Solutions

Various approaches have been suggested for resolving the problems in different ways with an aim to not requiring transcoding.

The Problems

N3043R states the primary problem that requires the disunification in its Rationale for medial disunification. Sgaw Karen shows it most clearly in that the association between base character and medial form is incompatibly different from that used in Burmese. It is not just that in Sgaw Karen a character has a different medial form, it is that that medial form is identical to another character's medial form. Thus if you render text using the default rendering, it will be displayed with the wrong medial occurring, thus making the text illegible. Therefore it is necessary to disunify the medials from their base characters. This core problem is often overlooked in the discussions. It is central and all solutions must address this primary issue.

In addition to the medials, there is the issue of tall -a. In Burmese U+102C may take two forms depending on context. In minority languages, either the context may change or only one form (the tall form, not the nominal form) is used.

Within the Burmese language there are still outstanding issues regarding encoding:

- How do we handle stacked nga (U+1004) as opposed to rendering a kinzi? (Known as the kinzi problem, with 3 existing solutions presented above)

- How do we handle stacked ra (U+101B), ha (U+101F), etc. as found in old Burmese which take a subjoined base form rather than a medial form?
- How do we handle stacked sa (U+101E) when default ligation causes the creation of great sa (*U+103F)?
- How do we handle superscript repha, as found in old Mon and Sanskrit?

See UTN#11 for more details on these issues.

These problems are soluble within the current encoding, but there is no unity as to their solution at this time and resolving this will require at least some transcoding of data. In addition, such solutions are all based around ZWNJ (U+200C) and ZWJ (U+200D) which means that these characters are no longer ignorable as they have tended to be considered.

Language Tagging

A commonly preferred alternative solution is to say that minority language text must be tagged with its language to get correct rendering.

Firstly, the Unicode standard has a stated requirement that:

> *Plain text must contain enough information to permit the text to be rendered legibly, and nothing more.*

As it stands the Sgaw Karen issue fails this requirement with the current encoding approach, due to not merely a different glyph appearing, but that wrong glyph has a different meaning in the language and so may not appear.

Secondly, language tagging is a useful feature that should be encouraged in systems. But it is unreasonable to expect that all systems will support language tagging. It is not a pre-requisite for using Unicode. Requiring the use of language tagging adds an order of magnitude to the complexity of implementation comparable to that of requiring smart font rendering. Unicode is stated as being a plain text encoding, it has no means for transferring language tags and so while in process data will be correctly rendered, data passed as plain text will lose its tagging and be rendered wrongly. Therefore, language tagging may only be used to resolve fine tune rendering that moves text from being merely legible to being rendered correctly.

Different Fonts

Some suggest that changing font should solve the problem. This is an almost identical issue to that of language tagging. Plain text does not carry font information and so legible rendering is lost on plain text transfer.

Add New Base Characters for medials

This approach adds new base characters to account for the different medial forms of the base character.

Looking at what is needed for some of the key minority extensions, I believe it will be necessary to add variant copies (i.e. duplicates!) of the following characters: U+1014 (na), U+1019 (ma), U+101A (ya), U+101C (la twice: Mon and Sgaw) and U+101D (wa).

While this may not seem a lot of characters, it must be noted that the additional characters will cause considerable confusion as to when they should be used and when the standard Myanmar characters should be used. They will look identical and will all need to be handled by all processes identically, except when following a virama. Each additional character will, in effect, add to the confusion rather than reduce it. And since the subjoined or medialised form of a letter is not a normative property of a character (in fact it is not specified by the standard at all, only by convention), as far as the standard is concerned, these characters will be identical. In many cases where identical letters have been encoded, one of them has had to be deprecated soon afterwards due to spelling ambiguities even when constrained by language.

Some examples of the problems of having such similar characters would cause are:

- Inability to make a joint Burmese/Sgaw Karen keyboard: having two ya keys, for example, is too horrific to imagine

- Increased confusibility causing security problems.

This approach only addresses the question of medials and other solutions would need to be found for the other issues, *at the same time.*

A variation of this approach can get the number of addition characters down by only adding variant copies for the problematic characters and then using true medial characters for the others. But this is even more untenable than extra base characters approach since it mixes models, particularly where one model is new.

ZWJ & ZWNJ Profusion

There are many ways of encoding without adding new codes. One is using variant selector another that seems to be favoured with Myanmar is to scatter ZWJ or ZWNJ around the virama (U+1039) to try to straighten out the variety of ways in which characters can combine when the vowel is killed. They key relationships that have been exposed are:

- visible asat with following character as full form.
- medialisation where the following character changes shape significantly
- subjoining as used for syllable chaining
- kinzi superjoining

If the virama (U+1039) is used as the core of all these relationships, then the relationships need to be marked somehow and most of the confusion has been based around how to use ZWNJ and perhaps ZWJ in combination with virama to indicate a particular relationship. Here we show various approaches to solving this problem.

Relationship	***Existing + NLP/UTN#11***	***N3043R***
visible asat	U+1039 U+200C	U+103A
medialisation	U+1039	separate code
subjoining	U+1039 (U+200D)	U+1039
kinzi	U+1004 U+1039 (U+200D) or U+1004 U+200D U+1039	U+1004 U+103A U+1039

The parentheses show the possible addition of a code in a particular context.

N3043R therefore provides a clean unambiguous solution to marking the various relationships and does away with all the ZWJ & ZWNJ issues completely. This fits better with how ZWJ and ZWNJ were originally designed to be used, to indicate a particular contextual form of a character is to be used when that context is not present. Thus marking a non-final form at the end of a word, for example or an initial form word medially. This is not how it was used in the Myanmar script and therefore has been problematic since its inception.

Addressing Tall -a

An alternative approach to handling tall -a is to keep U+102C with its current meaning, i.e. that it takes a tall form in certain contexts, and to encode a new minority tall -a character that does not change shape.

The question arises for those language which do have two forms of -a, what rules should apply. Mon and older Burmese texts can have different rendering rules regarding tall -a. Here, language tagging could be used, although is far from preferable since it requires out of band information. So an approach that doesn't require such information is going to have the advantage in this area.

The Burmese people consider the medials and tall -a to be separate letters. For example, they do not learn that great ya (◌) has any relation to ra (U+101B ◌ရ) unless they happen to study Burmese to University level.

People comment that they want the ability for a system to handle the choice of the appropriate form of -a automatically. First, the standard Burmese keyboard has the two forms of -a on it. But should such a system

be required, it is possible to add such rules into a keyboard method to do this. A sophisticated keyboard method is already needed to handle the desire to type U+1031 (ေ) before its consonant, even though it is stored afterwards, and this is a much harder problem. So adding contextual choice of -a to such a keyboard is no worse than adding rendering rules to the font system to pick the appropriate -a form, not to say the cost of implementing language tagging everywhere and integrating that with rendering.

Addressing Kinzi

The advantage which the proposed model brings to kinzi is that now kinzi is a unique, unambiguous sequence. There is no longer any need to check for a blocking code to see whether kinzi should not be rendered. This greatly facilitates other processes such as sorting as well.

An alternative approach would be to encode kinzi as its own combining character. The problem with this is that a combining character must be stored after the base character to which it applies. But kinzi is actually the final of the previous syllable and so is to be considered before the consonant on which it renders. Thus a combining character would greatly increase complexity in analysis processes, particularly sorting which can currently be done without any preprocessing.

Particular Issues

This section tries to lay out a clear set of semi-formal solutions to the various issues and what their impact would be.

Asat

Current scenario: asat = `U+1039 U+200C`.

Proposed scenario: asat = `U+103A`.

Current Scenario

Advantages:

- No need to transcode
- No need to add another character

Disadvantages:

- Retains the need for ZWNJ which some processes strip
- Turns a context marker (ZWNJ) into a necessary letter and mysteriously turns two invisible letters into a visible diacritic.
- Linguistically a virama after a vowel is meaningless but is necessary for rendering the asat on a vowel

Proposed Scenario

Advantages:

- Makes a very common character explicit
- Greatly simplifies virama (U+1039) into having a single function (subjoining)
- Allows use of asat for unambiguously marking kinzi and repha
- More accurately reflects use of asat on vowels.

Disadvantages:

- Requires a simple transcode.
- Adds a character

Medials

Current scenario: virama (U+1039) + base (x)

Proposed scenario: explicit character code (y)

Current Scenario

Advantages:

- No need to transcode.
- No need to add another character.

Disadvantages:

- Relationship betewen base and medial form is fixed
- Cannot add more medial forms without adding more base characters, with consequent added confusion.
- Ambiguity with kinzi abounds
- Cannot support repha
- Does not support subjoined forms that differ from medial forms. They are one concept.

Proposed Scenario

Advantages:

- Separates medialisation from subjoining and greatly simplifies and disambiguates the model
- Disambiguates kinzi
- Is simply extensible
- Models the insiders' understanding of their script
- Allows support for subjoined forms of medial base characters.

Disadvantages:

- Adds new characters so we have to fight for some unknown reason
- Requires simple transcoding
- Loses an implicit link from medial to base character

Conclusion

Various alternative solutions have been presented to address the problems listed. And given that this is about giving meaning to numbers there are many creative ways that that can be done. Others may think of more ways of providing a solution. The question then is which solution is better?

We have seen that even without a change to the standard, in order to resolve kinzi and the other Burmese issues, most data is probably going to have to be transcoded anyway. We also saw that this isn't a big problem just at the moment, although in the next 6 months, I envisage an explosion in the use of Unicode within Myanmar. With different and potentially incompatible solutions to the various problems. The fact that the Myanmar implementors themselves recognize the disadvantage of such incompatibilities, and that they support the proposed changes which avoid them, should be taken very seriously by the UTC and WG2.

In addition to the arguments presented in N3043R I would like to suggest a few other advantages of this model change:

- It more closely reflects the insider's understanding of their script. This should not be underestimated.

An encoding that more naturally reflects the understanding that people have of their script is always to be preferred over one that doesn't (all other things being equal).

- It provides a clean, natural solution to the problems listed.
- It relieves ZWJ & ZWNJ of their fixup function within the script.
- It is what the language computing community want and they are willing to put in the work.

I suggest therefore that readers not place the need for no transcoding above all other consideration when the users of the script do not place it there at this time.