UMdC Overview

Initial Draft 2017-08-29 by Bob Richmond. For discussion purposes.

Introduction

This note is aimed at readers familiar with Manuel de Codage (MdC) encoding of Egyptian as well as basic familiarity with Unicode. It is a description for discussion, not a tutorial or a specification.

UMdC is to be a coding manual for Ancient Egyptian in Unicode built on the basic 1071 hieroglyphs published in Unicode 5.2 (2009) and the 9 control characters for quadrat arrangements in the standardization pipeline for a future Unicode release. Version 1 aims to set out the fundamentals to help work move forward on developing and applying a new generation of resources and tools for working with digital forms of Egyptian.

The system described here is informed by practical issues of working with hieroglyphic in Unicode as well as various discussions on the topic over the last ten or more years. Some material is already available at https://github.com/HieroglyphsEverywhere/UMdC to share resources and enable discussions on what is essential for version 1 functionality and documentation.

An important feature of UMdC methodology is that hieroglyph control characters are visible and can be edited by hand just as in the original MdC system. We now use encodings like 𓀀𓐱𓁐𓐰𓏪, analogous to what in traditional MdC would be A1*B1:Z2.

Once the new control characters are standardized in Unicode and supported in software it can be expected that basic and casual use of hieroglyphic will normally work directly with the formatted quadrat forms e.g. 𓀀𓁐𓏪. Casual users should rarely encounter the control characters or need to be aware of UMdC. Experts will often need more than that so UMdC has longer term relevance.

An important feature of version 1 is a method for working with hieroglyphs not yet encoded in the basic 1071 Unicode set, including support for the extended Gardiner alphanumeric hieroglyph codes found in Hieroglyphica, JSesh and so forth. Apart from the fact this is useful for transitioning from traditional MdC transcriptions this feature helps support work going on to expand the hieroglyph repertoire in Unicode.

However, in other respects version 1 is minimalist and further extensions should be based on strong evidence of utility.

UMdC data files

A UMdC data file is a Unicode plain text file encoded with UTF-8 and named with a suffix ".umdc", e.g. EloquentPeasant.umdc. Unicode text files contain 0 or more lines of text and can be edited in simple text editors such as Windows Notepad, an HTML <textarea></textarea> input control or many other applications on all kinds of computer systems. The umdc suffix provides an indication that for best results the content should be viewed with a **UMdC compatible font** (or a composite font containing one) and is intended to be capable of being processed against UMdC guidelines by **UMdC-aware** software. Nevertheless, any valid Unicode text is valid UMdC in a technical sense.

Hieroglyphs in UMdC version 1 data

It is recommended that only the 1071 hieroglyphs already released in the Unicode Standard are used directly in UMdC data. These are guaranteed to be available in a UMdC compatible font. UMdC has ways of working with non-Unicode hieroglyphs (see below). Other techniques such as use of characters from the Unicode private areas or temporary character assignments in the SMP plane based on "what if" repertoire expansions are discouraged in UMdC unless guidelines are available. You should expect UMdC-aware applications to issue warnings if private zone etc. is detected. If this is a problem, the solution is to collaborate on establishing guidelines.

Version 1 does not define use of variation selectors for hieroglyph variants so VS are best avoided for the time being except for experimental purposes. As with private area characters if support is needed, guidelines will need to be written.

UMdC hieroglyphic is written horizontal left to right. There is no defined mechanism for vertical text in columns at version 1. There is a history of confusion in traditional MdC around encodings for vertical text so details will need to be fully documented before adding support to a future UMdC version.

Shaping of hieroglyph quadrats uses the proposed 9 control characters: 𓐰 𓐱 𓐲 𓐳 𓐴 𓐵 𓐶 𓐷 𓐸. Detained guidance on complex quadrat encoding is not yet available but the simple quadrats that form most texts can be modelled on traditional MdC equivalents.

Transliteration in UMdC

The MdC traditional transliteration characters ꜣ ꞽ y ꜥ w b p f m n r h ḥ ḫ ẖ z s š ḳ k g t ṯ d ḏ are available in Unicode except for ꞽ/Ꞽ Egyptological Yod. Transliterating yod as i or j works of course as does q as alternative to ḳ. A UMdC compatible font provides these transliteration characters in upper as well as lower case plus the LATIN SMALL LETTER I WITH SPIRITUS LENIS/ LATIN CAPITAL LETTER I WITH SPIRITUS LENIS characters intended for use as Egyptological Yod (and currently in the Unicode standardization pipeline). UMdC version 1 treats transliteration characters just like any other Unicode character – there are no specially defined conventions.

UMdC compatible fonts

As well as hieroglyph and transliteration characters, a UMdC compatible font must contain standard ASCII characters. It must not perform quadrat shaping using the 9 control characters. Additional characters such as ⟨ and ⟩ can be useful for editorial markup of hieroglyphic and other popular characters such as èéü are desirable so a list should be agreed. We can consider Greek, Coptic, Meroitic although it would be unfair to place a burden on font designers or software developers to have to support them so probably they should be listed but optional?

In any event so long as a font supports the UMdC required character list it is UMdC-compatible. Any number of other characters can be available.

Note that for umdc documents incorporating less commonplace characters or demanding character sets (e.g. Japanese, Chinese) a simple answer is to use a plain text editor such as BabelPad that supports composite fonts. Alternatively adopt a workflow using a standard word processor.

UMdC is a permissive system

UMdC is plain text so it can be used for many purposes. For example, if you are primarily targeting a spreadsheet or word processing document its usually a good idea to keep the data as plain as possible. At the other end of the spectrum if your data is to be inserted in an HTML web page you may want to incorporate HTML tags in your UMdC text so you might write:

```
<div class=”UMdC”>
   <p>|R(1),[1.1]𓊃𓐰𓀀𓐱𓏤𓊪𓏲𓃹𓐰𓈖𓇋𓈖𓐰𓊪𓐱𓏲𓃢𓐍𓐰𓋭𓏲𓈍𓐰𓈖𓀀𓂋𓐰𓈖𓆑𓇏𓏏𓐰𓏭𓀀𓊪𓐱𓏲𓐰𓈖𓇏 etc.</p>
</div>
```

In the situation where a hieroglyphic source is damaged or a hieroglyph transliteration is uncertain. MdC uses notation like -#b-G7-#e- and -r:V31A#13- to request quadrat shading. UMdC version 1 defines no shading mechanism. But one can write -#b-𓅆-#e- or 𓂋𓐰𓎢#13 and although applications in general have no way of shading it would be possible for a system such as JSesh to recognize the # conventions in UMdC data if MdC compatible shading were required. In this hypothetical instance conventions can be documented in guidelines for using UMdC with JSesh.

UMdC Extensions using tags

The extension mechanism for extending basic UMdC capabilities uses UMdC-specific XML-like tags. To support this mechanism UMdC-aware applications need to detect relevant tags and process the predefined XML character entities &amp; &lt; &gt; &apos; &quot; (& < > ‘ “). Normally it is unnecessary to use entities except in a situation where text can be confused with markup tags (specification can give details).

UMdC Version 1 just defines three extensions. One for comments and two for hieroglyphs not yet available in Unicode.

Comment Extension

Comments in Basic UMdC data can be written using an XML-like tag e.g.

```
<!--This is a comment -->
```

Multiline comments are valid e.g.

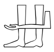
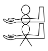
```
<!—
PapyrusWestcar.umdc
From JSesh 6.5.5 sample westcar.gly
-->
```

A UMdC-aware application typically recognizes this pattern and treats it as an XML style comment not primary text. Entities can be used if this special processing is not wanted.

Non-Unicode Hieroglyph extensions

Variants of Unicode hieroglyphs can be encoded using UMdC-specific <gv></gv> tags. For instance, hieroglyph A1A from the Hieroglyphica 2000 catalogue can be written <gv mdc=”A1A”>𓀁</gv>. A Hieroglyphica sign such as H10 (without an obvious variant in Unicode) can be expressed as <gv mdc=”H10”>?</gv>. 𓀭 mirrored (i.e.) can be expressed as <gv mdc=”A40\”>𓀭</gv>.

A UMdC-aware application may use the tag information. A simple example is an application that converts UMdC to traditional MdC which can simply pick up the mdc attribute value from the tag instead of looking up the Gardiner code for 𓀁 or 𓀭 as it would do for a normal hieroglyph.

Some geometric arrangements of two or more hieroglyphs such as Unicode D059 𓃁 ꜥb are already present in Unicode and some others can be expressed as combinations e.g. using the OVERLAY MIDDLE control 𓐶. Other combinations in Hieroglyphica that can’t use controls such as D298  and V72  can be encoded in UMdC using <ga></ga> tags rather than as variants: e.g. <ga mdc=”D298”>𓂝𓃀𓃀</ga> and <ga mdc=”V72”> 𓎛𓂝𓂝 </ga>.

As with <gv></gv> a UMdC application can use the mdc attribute to identify the variant.

This way of dealing with repertoire extensions is potentially expandable in various ways. For instance, an attribute could be added to indicate a hieroglyph is damaged in one way or another.

Concluding Remarks

I don’t think there is any doubt that some plain text system is essential for practical purposes.

UMdC as described can sit alongside other systems such as traditional MdC and complex XML systems such as that used by the Text Encoding Initiative.

I’ve kept the system simple for version 1 to leave opportunities for discussion on the best way to address issues like damaged or missing hieroglyphs, uncertainties about encoding individual hieroglyphs, damaged quadrats etc.

Clearly some functionality like use of red ink could be addressed using special tags for UMdC-aware applications in future.

There’s plenty more to be written but I’m interested in suggestions, questions, potential problems and feedback so version one can be wrapped up in the next few weeks and I can update the online samples and documents to match.

If you have any MdC/JSesh material you think might be useful for illustrative purposes please get in touch.

Bob Richmond