Considerations concerning Egyptian Hieroglyphs extension

Author: Michel Suignard

Date: October 24, 2022

Summary:

The following represent some early considerations concerning the encoding of extension to the Egyptian Hieroglyphs repertoire. They expose some current issues concerning compound signs, encoding principles, and propose some solutions. The author is looking for recommendation and advice from UTC.

Productive font rendering

Egyptian Hieroglyphs typically involve compound signs made of atomic signs grouped in cluster, either as quadrat or simply stacked or overlaid. Many of these compound signs may be rendered without the need to create atomic glyphs for these compound signs, using glyphs of various sizes for the same sign and allowing multiple placements of these signs. These complex renderings are made using specific text engine and underlaying font technology (such as OpenType) without much of a need for atomic glyphs. This is usually called 'productive' rendering. Its best example is its use in 'quadrat' and insertion rendering.

Example of quadrat (source [1]):

Examples of corner insertion (source [2]):

Or middle insertion

In order to do this, the font needs to contain multiple size of the same glyph (like the sign X1: half round bread). And typically, exact placement of the inserted pieces cannot be guaranteed; quotes from [2]:

> For example, there should be primitives to distinguish such spatial arrangements as overlay and corner insertions, as satisfactory transcriptions cannot be obtained without them, but we would not aim to fine-tune scaling of signs or the distances between them. This also means that we cannot expect Unicode to replace tools such as JSesh, as the power of such tools is beyond the capabilities of common font technology. For faithful transcriptions of hieroglyphic texts that are to be included in printed publications, JSesh and other such tools will still be needed in the future.

Typically, every sign that is a candidate for insertion may need to be duplicated in many sizes in the font to create an aesthetically pleasant rendering. With the extension getting in many thousands of base signs and the limit of 64K glyphs per font imposed by OpenType, it is easy to see the potential issue on hand. At the same time, creating an atomic glyph for each compound sign (itself encoded as a sequence of code points) may also run into the same 64K glyph limit. It is however encouraging that this 64K glyph could be lifted soon.

There is also the special case of some containers that can be defined either as a single sign (container) or enclosure elements (start and end) and use the same principle as cartouche. Both concepts can be productive. However, they also require having multiple sizes in the font for the various signs.

Finally, there is the case of overlay where productive rendering will not in most case be satisfactory. Signs that are overlaid typically 'fuse' together; they are not simply superimposed. This means that most of these cases need to be treated as a ligature/substitution using for example the OpenType GSUB tables.

An important consideration concerning overlay is that most are not functionally different from the functions represented by the sequences of their element. Therefore, if the overlay represents the functions of the components, it should be productive, if not, it should be atomic. But from a point of view of font production, it is likely that only 'known' overlays will be correctly rendered (using GSUB tables for example), while others will just be 'best effort'.

As a result, one of the few examples of an overlay which has a specific function:  (phonemogram *šmʿ*), compared to its elements  (phonemogram *nḫb*) and  (phonemogram ʿ) is proposed as an atomic character. However all other identified overlays should still use ligature/substitution in the font.

Note: one could argue that even these overlays with distinctive functions could also be encoded as sequences if they are guaranteed to be represented as single glyphs. However, there are bad precedent in the context of combining sequences, so having them encoded as atomic units guarantees a correct rendering by all fonts.

Compound signs and their definition in Unicode code points

Compound signs (i.e. signs made themselves of multiple simpler signs) can be encoded either atomically (single code point) or a sequence of code points. Given the productive way on which hieroglyphs tend to be created and discovered, it is not practical to have an atomic representation for all combinations.

However, it would be extremely valuable to have sequences defined for typical compound signs found in collections like those represented by Hieroglyphica and JSesh. Because many of these compound signs could have multiple sequence definitions, it could lead to confusion, and having a recommended sequence exposed in a technical report would be extremely useful.

The author would also advocate that for most (if not all) cases where a situation of overlay/insertion/arrangement is claimed for compound signs that have been verified by Egyptologists, a sequence should be provided. At minimum, it would assure that we are encoding the elements necessary to encode these sequences.

Compound signs that can use either enclosure or insertion

We always had two solutions to encode some compound signs that use base signs that look like a combination of enclosure signs, especially  and  . We can either use the enclosure signs such as  , , and for those 2 cases above and a bit more for all permutation of the hwt sign.

While the enclosure signs can productively accommodate enclosure of multiple groups of signs (which would require font ligature/substitution for middle insertion), they have their own challenge when used in different text orientation as explained in [2]:

> **middle insertion versus pairs of delimiters**
>
> It is tempting to disallow use of middle insertion (…) where the same appearance can also be achieved using an enclosure as above. We would hesitate to impose that restriction at this time however, as middle insertion has a number of advantages over enclosures.

> First, the delimiters of an enclosure are interpreted according to text direction, that is, the delimiters are rotated by a quarter turn if the direction is vertical. Connected to this, we assume that the implementation in say OpenType does not require remembering the opening delimiter until the moment when the closing shape is being drawn. However, it is generally undesirable that a simple composition like  would turn into  upon a change of text direction, which would unavoidably be the case if pairs of delimiters are used. A related issue is the existence of 'vertical' cartouches in horizontal text, …, where use of the middle insertion, rather than a pair of delimiters, would prevent an undesirable normalization to a horizontal cartouche. Second, we would not expect nested enclosures to be implemented, …, as this would require considerable effort to handle a case that is very rare. However, corner and middle insertion would normally be available within enclosures, ...
>
> Allowing say  to be expressed using middle insertion, next to encoding as enclosure, does not require additional effort, beyond the implementation of middle insertion in general. In practice, most users may prefer middle insertion over enclosure if only one small group is enclosed. If several groups are enclosed, then middle insertion is not applicable.
>
> Next to O6  , we would need graphical variants with the square in the other three corners. We will leave this to the work on the sign list that is going on in parallel.

Another challenge of the enclosures is that because they are primarily designed to create cartouche effects such as in:  , their height tends to be taller than typical compound signs with contained object. The following picture shows a compound sign embedded in a cartouche which shows the different vertical size of the cartouche and the compound sign  :

Note: Interestingly, document [2] says that such embedding using enclosure is not going to be supported, but if insertion is used for the bird enclosed in a hwt as recommended (because the contained sign is a single group), there is no enclosure embedding, and it can be easily implemented.

The text in the Unicode 15.0 Core Specification [4] mentions the same single group restriction for the hwt container:

> Note that when inserting into the HWT enclosure, only a single group of one or more signs can be inserted. If a sequence of groups is to be enclosed into the HWT, the enclosure controls should be used, as described later in this section under "Enclosure Controls."

In the same vein, the core specification shows an example of a single group insertion:

 13196 <13193, 13433, 13437, 133CF, 13430, 131FF, 13438>

In practice, compound signs which contain multiple groups of signs are extremely rare (excluding of course cartouches). A vast majority of them are contained in a hwt element. The IFAO sign collection does not contain any such multi-group contained signs and Valeurs Phonétiques has only a few that could be classified as such:

A first approach reading of the various documents could give the impression that enclosure signs and middle insertion are both common solution for contained elements, but most inserted elements are single groups, including cases where the hwt sign is the base container.

In conclusion, more emphasis should be put on the use of middle insertion for contained signs and we should make sure that it can be used in all scenarios involving inserted single groups. In other words, the insertion model is the generic case, and the enclosure model is the exception.

Definition of a container, or can a container be defined as a sequence of code points

Most containers are simple atomic signs, consider for example ; but others such as the hwt sign (O6), exist in rotated/mirrored form; for O6 you could have: , , and . Another case is the enclosure with battlements (O13). Compound signs exist with this form, such as , or . However, the second compound sign uses a mirrored version of O13 which is not proposed for encoding. But if we mirror the base, to create the compound sign we now have to mirror the base character to indicate the glyph in which the content is actually inserted. Something like (using middle insertion which is only an approximation of the sign):

: 13267, 13440, 13439, 133BA> or

(The text engine 'understands' that the mirrored operator affects the previous character and the sequence of two code points <13267, 13440> defines the container, making unnecessary delimiters like 13437 and 13438 .

For the hwt (O6), things get more challenging because we may have to combine mirroring and rotation; so the base definition becomes more complex. And again the Unicode 15.0 core spec does not specific the exact syntax of combining rotation and mirroring.

For example, assuming that the hwt sign is defined as follows: U+13257: , the other versions are defined as follows (assumption is that the container would be square):

: mirror: <13257, 13340>

: rotation 180°: <13257, FE01>

: rotation 180° + mirroring: <13257, FE01, 13340>

Some examples:

: <13257, FE01, 13439, 132DE>

: <13257, FE01, 13340, 13439, 132F9>

(In that example, the base container is defined by a sequence of 3 code points: <13257, FE01, 13340>)

At this point, it looks like only O6 and O13 mentioned above have this feature, so while the number of these required sequences is limited it would be wise to document these cases because it may not be obvious to the casual readers that containers can be specified as sequences of code points.

Additional consideration concerning container sizes

It is not totally clear how OpenType adapts to variable sized content. Ideally for the hwt sign, to have decent rendering of inserted signs you would need a narrow and square container, even maybe a middle-sized version.

Again L2/21-248 [2] says:

> One slight complication is that some common signs such as 𓍿 and 𓎆 may need to be included in a font in a second, larger version, such as 𓍾 and 𓎇, for use with middle insertion. This is easy to realize within the existing implementations of Ancient Egyptian control characters in OpenType.

This should be mentioned in the core spec. Now some experts want the base container to have an average width (often either too narrow or too wide for many signs), but if the technology allows some variable sizing, the concern is moot.

Token versus abstract signs and encoding strategy

When exploring the Egyptian hieroglyphic universe, we are facing multiple considerations:

1) The paleographic forms are found in written or carved evidence and are typically referred as tokens. While there may be some limited consolidation to avoid duplicates, the representation of these forms tends to be close to the actual signs found on papyrus or stone.
2) Above that, you will find many more abstract forms used in publication. They tend to represent function or classes, but they do sometimes graphically deviate from the tokens from which they are derived.
3) Many publications when describing evidence found on actual buildings or papyrus navigate between actual representation of the tokens and abstract representation of the same elements.
4) Therefore, there are vast amount of material that mix these signs with no clear delimitation.

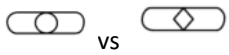
When Unicode encoding decision is to be made, one has to look at both abstract signs and tokens. At minimum, abstract signs should have a relation to some tokens. This has resulted in some encoding decision:

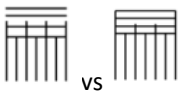
- A sign should not be encoded unless it is related to well documented token(s), and preferably not to single evidence (hapax).
- However a sign may be encoded if it is necessary as a base for many compound signs even if the sign is not itself associated with a token.
- If the number of compound signs using that base is limited, it may be better to encode them as atomic signs. For example (HG/JSesh O339):

- If a compound sign is not associated with any token, it is not necessary to create base signs that would be otherwise required to represent it with a sequence.
- Alternatively, if a contained object does not exist as non-contained, it may make more sense to make it atomic, example:

The main tension point is that some signs are represented by two or more abstract signs with the same function, and decision should be made to either have only one code point or multiple code points. In one case (single code point) it would be easier to search and identify the functions, but you don't get to represent existing publications adequately. In the second case (multiple code points), it is more difficult to identify the functions, but more existing publications can be represented.

Some examples (X4, O33):





At present, the encoding proposal is favoring consolidation of these abstract signs, meaning that many existing publications will require alternate forms to be correctly represented (for example, all Dendara publications by Sylvie Cauville). To facilitate this, such common substitution/variants are currently documented in new data set.

Transliteration

Because the function in context is an important attribute of Egyptian hieroglyphs it is an important requirement to be able to provide that data to existing and newly proposed hieroglyphs.

That functional description includes transliteration in Latin script of the original hieroglyphic writing system. The current data is based on the MdC version as documented in [3]. The advantage of the MdC version is that it is pure ASCII but uses casing in an odd way. Another popular option is the Gardiner convention based on the same reference [3]. This is the one in use in the Valeurs Phonétiques (VP) values documented in the database. Some examples:
Hw.t-nTr versus *ḥwt-nṯr*

Following is an excerpt from [3]:

Conventional Transliteration Schemes													
Glyph	Brugsch	Erman	Budge	Erman & Grapow	Gardiner	Edel	Manuel de Codage	Hodge	Schenkel	Hannig; Allen	Hoch	Schneider	Conventional Egyptological pronunciation
	1889	1894	1910	1926–1953	1957	1955[1]	1988	1990	1991	1995; 2000	1997	2003	
	ꜣ	ꜣ	a	ꜣ	ꜣ	ꜣ	A	ꜣ	ꜣ	ꜣ	ꜣ	ɹ	/ɑ, ɑː/
	ꜥ	i	á	i, j	i	j	i	ʔ	i	j	i	i	/i, iː, j/
	″	ï	i	j	y	j	y	y	i	j	y	i	/iː/
	ꜥꜥ	y	i	j	y	jj, j	y	y	y	y	y	y	/iː/
	ꜥ	ꜥ	ā	ꜥ	ꜥ	ꜥ	a	ꜥ	ꜥ	ꜥ	ꜥ	ɑ́	/ɑː/
	w	w	u	w	w	w	w	w	w	w	w	w	/w, uː/

	b	b	b	b	b	b	b	b	b	b	b	b	/b/
	p	p	p	p	p	p	p	p	p	p	p	p	/p/
	f	f	f	f	f	f	f	f	f	f	f	f	/f/
	m	m	m	m	m	m	m	m	m	m	m	m	/m/
	n	n	n	n	n	n	n	n	n	n	n	n	/n/
	r, l	r	r, l	r	r	r	r	r	r	r	r	l	/r/
	h	h	h	h	h	h	h	h	h	h	h	h	/h/
	ḥ	ḥ	ḥ	ḥ	ḥ	ḥ	H	ḥ	ḥ	ḥ	ḥ	ḥ	/ħ, h/
	ḫ	ḫ	x, kh	ḫ	ḫ	ḫ	x	x	ḫ	ḫ	ḫ	ḫ	/x/
	ḫ	ḫ	x, kh	ḫ	ḫ	ḫ	X	x	ḫ	ḫ	ḫ	ḫ	/ç/
	s	s	s	s	s	z	s, z	z	s	z	s	s	/z, s/
	s	s	s	ś	s	s	s	s	ś	s	s	ś	/s/
	š	š	ś, sh	š	š	š	S	š	š	š	š	š	/ʃ/
	ḳ	ḳ	q	ḳ	ḳ	q	q	q	ḳ	q	q	ḳ	/k, q/

You can usually convert from one version to another but there are some limits because newer versions disunify some readings, so the sooner we get a stable standard, the better.

Note: Related to that, we have a bit of a challenge to represent some of these transliteration in the code charts (if we want to do that). The MdC version being pure ASCII is not an issue. But any other conventions would have code points outside the Myriad Pro font used for rendering in the code chart for annotation, resulting in some ransom note effect (when a code point is outside the scope for annotation text). Currently the few occurrences in the existing blocks use the Gardiner convention and we are lucky that only few instances create the ransom note effect. Example:

13300 EGYPTIAN HIEROGLYPH S040
- phonogram 'wꜣs'

13301 EGYPTIAN HIEROGLYPH S041
- phonogram 'ḏꜣm'

Character names

For Egyptian hieroglyphs the naming convention is a challenge. Originally, the most common convention was the Gardiner 1953 style: groups A to Z, excluding J, and adding Aa for unclassified, Ff for Hieratic transcription, and Nn for nomes. A is for example the group for Human, and the first element is A1.

Many other naming conventions have been derived from the original Gardiner, using the same naming convention, however they deviate in their extensions, affecting different values signs for the same sign. This makes that naming convention unstable.

Unicode for the existing block uses a variant of that naming convention by using one or two letters A to Z, minus J, NU and NL for the nomes and AA for unclassified (Ff is not recognized as a separate group). Then it uses a 3-digit convention for element of each category, therefore Gardiner A1 becomes A001. The convention is called Unikemet. This convention has not been widely adopted outside Unicode.

The issue with that convention arises when trying to cover the 10,000+ signs that could be identified in extensions; some groups become so large to be unmanageable. For example, the extended A group contains more than 1000 members. In addition, the apparent name collision still exists. For example Unikemet A069  refers to A71 in Hierogyphica and Jsesh, and Hieroglyphica/JSesh A69  corresponds to a not yet encoded character. At minimum the Unikemet convention would have to be extended with another digit (such as A001 becomes A0001).

An alternative is to use a naming convention on a taxonomy that subdivide these Gardiner groups in smaller sub-groups. The only well-established taxonomy doing that is the one created by IFAO (Institut Français d'Archéologie Orientale). The author has created a naming convention following that taxonomy which still uses the A-Z, AA convention but subdivides down to another level with subgroups like A-01, A-02,..B-01, B-02. And each subgroup uses a 3 digits convention to identify their member. For example, Gardiner A1 becomes A-010-001. Most of the subgroups have 50 or less members, and only very few exceed 100, and none reaches 200. While reluctant at the beginning, a small group of Egyptologists have in fact help the author to improve the classification by moving signs among categories.

The last reasonable alternative is to use the code point as done for other large collections like CJK. But all meaning is lost in the name which may be extreme in this case.

Finally, the current database in use to categorize the totality of these signs (over 10,000 not all candidates for encoding) uses an index based on Private Use Area codes, starting at U+F0000 and currently ending at U+F470C. The index scheme reserves holes between each sub-groups to allow for extension near current members. The PUA index was originally ordered based on an initial IFAO classification, but with ongoing refinement, there is only a loose relationship between the PUA code order and the current IFAO based taxonomy. One feature of the PUA code is its guaranteed stability since inception; the IFAO based names are still being worked on and would only become stable when the first major extension is proposed for encoding. In summary, possible names convention for the proposed sign  :

Naming convention	**Syntax**	**Examples EGYPTIAN HIEROGLYPH [EXTENDED]**
Unikemet extension	[A-IKZ, AA, NU, NL]ddd	A0071
IFAO based	[A-IKZ,AA]-dd-ddd	A-01-003
Code point	hhhhh	13460
PUA	hhhhh	F0002

Note that in the IFAO based notation, there is no correlation between the Unikemet or Gardiner number and the values reflected in the subgroup number or the index in the subgroup (as illustrated above in A071 being represented by A-01-003).

Ancillary information and what to put in the code chart

In all cases, a lot of information has been collected when establishing the collection of candidates for Egyptian Hieroglyph extensions, including extensive sources, discussion point about the merit for encoding, and more importantly sign description and function in context. For example for the sign mentioned above  :

- Sources: IFAO (1,3); Dendara VIII 5,13
- Sign description: Man, seated on heel, right knee raised, right arm raised, left arm in front of body.
- Function in context: Classifier human being (rmn.w)

It would not be practical to include all these pieces of information into the code chart. Many code points have many sources, and the sign description can be verbose for complex signs. However, the function in context is extremely useful and typically concise.

The amount of ancillary information on the code chart also depends on the naming convention. A code point-based naming convention does not favor putting any information in the code chart, but instead a model similar to CJK, where all info is pushed to a database. Note that this would demand some tooling modification in the code chart tool to suppress the creation of a useless name list in the code chart for those blocks.

Any other model conveying some information in the character name should probably also have at least the function in context which is already shown for some Egyptian hieroglyphs in the encoded block as shown above in the discussion about transliteration. Example:

13460  EGYPTIAN HIEROGLYPH A-01-003

• classifier human being (rmn.w)

Note that while it is a longer term to develop a database for Egyptian Hieroglyphs, the first encoding proposal will probably only have a text file with some tabulated information as described above (catalog name, description, and function in context). The sources information is still very preliminary and subject to many discussions.

References

[1] Cluster model for Egyptian Hieroglyphic Quadrats, Andrew Glass, L2/20-176
https://www.unicode.org/L2/L2020/20176-hierogyph-cluster.pdf

[2] Additional control characters for Ancient Egyptian hieroglyphic texts, Andrew Glass et al.
Dec 22,2021 L2/21-248 https://www.unicode.org/L2/L2021/21248-egyptian-controls.pdf

[3] Transliteration of Ancient Egyptian, Wikipedia, accessed Oct 5 2022
https://en.wikipedia.org/wiki/Transliteration_of_Ancient_Egyptian

[4] Egyptian Hieroglyphs, Unicode 15.0 core specification
https://www.unicode.org/versions/Unicode15.0.0/ch11.pdf#G26607