
Journal of Mongolia International University: Research and Innovation

Journal homepage: <https://jri.miu.edu.mn>, journal@miu.edu.mn

Research paper

Preserving the Visual Presentation Form of Digitized Traditional Mongolian Script While Waiting for Unicode Specifications to Stabilize

Jonathan Sande 

Lecturer, Computer Science Department of Mongolia International University

Corresponding Author: jonathan.sande@miu.edu.mn

Journal of Mongolia International University: Research and Innovation (2026), Vol. I, No. 1, pp. 50–58

doi: 10.66712/jmri.v1i1.5

Published online: 22 May 2026

© The Author(s). This is an Open Access article, under CC BY-NC 4.0.

Computer Science Department

A B S T R A C T

Purpose: Digital encoding for traditional Mongolian text has evolved over time. The two main strategies for encoding Mongolian text are graphetic and phonetic. Graphetic solutions encode the glyph shapes while phonetic solutions encode the letters and rely on contextual shaping rules to determine the letter shapes. Unicode takes the phonetic approach by encoding the letters of the traditional Mongolian alphabet, but it does not specify any shaping rules. Variation between fonts and variation as the specifications evolve over time cause Unicode text to be malformed when displayed in an environment different than it was created in. Attempts have been made to standardize the Unicode shaping rules. However, not all entities have agreed on a single standard yet.

Methods: In order to preserve the correct visual appearance of digitized traditional Mongolian text until an unambiguous specification is standardized and agreed upon by all parties, this paper proposes using a graphetic encoding model to store the glyph shapes. This text can then be converted back to Unicode on demand.

Results: An open source converter (`mongol_code`) was developed and published, which converts between Menksoft code and Unicode following the GB/T 25914-2023 specification. The converter achieved a 100% pass rate across 1,401 unit tests with 88.7% code coverage.

Conclusions: Because Menksoft code encodes both the glyph shapes and phonetic letters, it successfully preserves these elements during conversion to and from Unicode, which allows for long term storage while waiting for Unicode specifications to stabilize.

Keywords: traditional Mongolian script, Unicode, phonetic encoding, graphetic encoding, encoding conversion

Introduction

Although the encoding of digitized traditional Mongolian text has been around for decades, there is still not an internationally accepted specification for how every letter combination (ligature) should be encoded in the phonetic-based model of Unicode. This has led to frequent display errors when sharing text across applications that use different fonts and encodings. There is also the difficulty of maintaining text integrity over time as the specifications for the Unicode shaping rules have evolved. While waiting

for rendering ambiguities and international disagreements to be resolved, software developers need a way to preserve the visual presentation form of stored Mongolian text. This article recommends converting Unicode text to a glyph-based encoding model for storage. This text can then be converted back to Unicode when needed. Such a practice is a temporary measure and can be abandoned once the specifications for Unicode shaping rules are formalized and agreed upon by all parties.

This paper will assume that the reader has a background on the fundamentals of writing the

traditional Mongolian script. Other publications have adequately explained Mongolian writing features such as directionality, cursive joining, vowel harmony, free variation selectors (FVS), and the Mongolian vowel separator (MVS) (Kullmann & Tserenpil, 2015; Unicode Mongolian Working Group, 2019).

A number of models have been used to encode traditional Mongolian text over the years.

Among others, these include CMs encoding in Mongolia and Menksoft, Boljoo, and Saiyin encodings in Inner Mongolia (Batjargal et al., 2011). Traditional Mongolian was standardized in Unicode in 1999 with the model developed by Quejingzhabu at Inner Mongolia University (Teneg, 2025).

Unicode itself doesn't define the contextual shaping rules needed to form words in Mongolian. Because of this, Mongolian Unicode font makers have defined their own shaping rules, which often differ from the rules used by other font makers. The mix of differing Unicode fonts along with non-Unicode fonts has led to a situation where text written in one location would become malformed or illegible when copied to another location. Figure 1 is a screenshot from

Ishida (2020) showing how seven different fonts render a Unicode Mongolian Medial GA followed by the free variation selectors FVS1, FVS2, and FVS3. Horizontal rows should all be rendered the same, but as you can see, they are not.

Encoding Models

There are a number of ways that Mongolian has been encoded in the digital age. Broadly speaking, these fall into two different categories: graphetic and phonetic. The following sections will cover some of the main examples of each.

Graphetic Models

Graphetic encoding models assign a code for each distinct visual form that a unit of Mongolian writing may take. Depending on the architecture of a particular graphetic model, the form represented by a code could include a whole letter, such as the medial dotted form of a masculine GA, or it could be a sub-letter form, such as breaking the aforementioned GA into two separate dotted teeth. See Figure 2.

The abstract form itself is displayed with a glyph from the font and is generally independent of any pronunciation information or letter names. The graphetic encoding does not specify the presentation style. That means individual fonts can display a print style or a handwriting style and include contextual ligatures that make certain letter combinations flow more smoothly.

CMs Encoding

Classical Mongolian script (CMs) encoding was developed by Peter Cheung and has been widely used in Mongolia (Batjargal et al., 2011). This encoding reuses the Latin/ASCII codes to store the glyphs for traditional Mongolian. As an example, Figure 3 shows the word Mongol written in traditional script using CMs encoding. The second line adds space between each glyph to make it clear from the third line which Latin character each glyph is mapped to.

The primary advantage of CMs code is that there is no need to install a special Input Method Editor (IME) for Mongolian. A standard Latin keyboard is fine. All that is required to display the Mongolian text is to use a CMs font. In ASCII decimal, the string "Moehgo&" would be encoded as [77, 111, 101, 104, 103, 111, 38, 34].

When converting from Unicode to CMs code, much of the pronunciation data is lost. For example, Unicode medial O, U, OE, and UE are all mapped to a single CMs glyph. Thus, there is no way to convert accurately back to Unicode from CMs code without an external source of information. This means CMs code is not an ideal candidate for long-term presentation form storage.

ZCode

ZCode was developed by Chuck Namujila (Namujila, 2020). The current implementation (Z52) uses 52 codes from the Mongolian Unicode Block (U+1865—U+18AA). These codes are seen as temporary and the proposal is for Unicode to assign new code points for the ZCode forms. The representative glyphs are shown in Figure 4.

OpenType shaping rules can be applied in a font to make glyph combinations flow better, but the basic glyphs themselves are enough to write any Mongolian word.

The simplicity of ZCode would solve a lot of problems related to complex text shaping rules or duplicate encodings inherent in some of the other models. However, the Mongolian Working Group for Unicode did not express much hope that such a model would be adopted into Unicode (Anderson, 2020).

Like CMs encoding, converting from Unicode to ZCode is a lossy action since any phonetic information that the Unicode contains is lost in the conversion.

Menksoft Code

Menksoft encoding, created by the Menksoft company, uses the Unicode Private Use Area (PUA) for its codes (U+E234—U+E34F). Menksoft code is different than ZCode in that it encodes both the precise glyph shape and also the Mongolian letter information. For example, many of the glyphs for O and U are visually identical since they are written the same, so Menksoft has distinct codes for the glyphs of both letters. In some ways, this makes Menksoft code a fusion of the graphetic and phonetic models.

Figure 5 shows the full set of glyphs from the Menksoft Qagan font.

While visually duplicate glyphs can create problems for searching and sorting, it has the benefit that it allows for a non-lossy method of converting to Unicode and back. Because of this, Menksoft code is the encoding chosen by this research for long-term storage of the presentation form of traditional Mongolian text.

Phonetic Models

Phonetic encoding models of Mongolian focus on the letters or sounds of the language rather than the shapes that a particular letter may take in context. Unicode has gone this route (The Unicode Consortium, 2025). Table 1 shows the code values, representative forms, and names for Unicode Mongolian letters. Digits, punctuation, control characters and other scripts in the Mongolian Unicode block are omitted from the table.

Mongolian Unicode fonts use contextual rules to select the appropriate internal glyph when rendering text. However, because of the lack of clear specifications in the past on how those rules should be defined, different fonts did not always select the same presentation glyph for a certain string of

Mongolian Unicode letters and control characters. You've already seen the result of this in Figure 1.

The situation is improving as more work is being done to precisely define the contextual rules for displaying Mongolian Unicode. The sections below will describe recent efforts in China and Mongolia.

China: GB/T 25914-2023

China has published the document GB/T 25914-2023, whose title in English is “Information technology—Traditional Mongolian nominal characters, presentation characters and use rules of controlling characters” (State Administration for Market Regulation & Standardization Administration of the People's Republic of China, 2023). This recommended standard goes a long way in disambiguating the contextual rules for rendering Unicode strings in Mongolian. Unicode itself has not adopted this standard, but there is ongoing work on Unicode Technical Note (UTN) #57 that is incorporating the specifications of GB/T 25914-2023. Additionally, Microsoft's Mongolian Baiti font and Google's Noto Sans Mongolian font satisfy the specifications of GB/T 25914-2023 (Jiang, 2025).

GB/T 25914-2023 tells the precise presentation form that each Mongolian Unicode letter should take in context. This includes control characters like FVS1, FVS2, FVS3, FVS4, and MVS. It also includes fixed words and suffixes.

Mongolia: Refined Model

Officials in Mongolia have not published their own specification yet, and it is still to be determined whether they will approve the work on UTN #57 or take another route (Teneg, 2025).

However, the Bolorsoft company in Mongolia has developed what they call the refined

phonetic model, that is, a refinement of the Unicode phonetic model. The refined model cleans up unneeded historical variants, fixes positional mismatches, removes the need for some control characters, and splits out the feminine forms of QA and GA into KE and GE (Bolorsoft, 2019).

While this does make the encoding significantly simpler, there remains the compatibility issue for sharing text to environments that don't support the refined model encoding. The top line of Figure 6 shows the output of the word *mönkh*

(eternity) in refined model encoding but displayed with Google's Noto Sans Mongolian font. The second line uses standard Unicode encoding by replacing the refined model KE (U+1888) with QA (U+182C) for proper display with Noto Sans Mongolian.

The refined model has some excellent suggestions for improvements on the Unicode standard. However, it is not the purpose of this paper to advocate for one phonetic model over another. The differences between current encoding models merely illustrates the need to have a means of storing text without corrupting the visual presentation form.

Methodology

This section describes the process of how potentially volatile phonetic-based Unicode text can be converted to graphetic-based Menksoft code, stored long-term without danger of losing the presentation form, and then converted back to Unicode when the Unicode specifications have stabilized.

The process of creating a converter between Unicode and Menksoft code involved analyzing the font glyphs in Menksoft fonts and then writing the conversion algorithm to and from Unicode.

Menksoft Glyph and Code Analysis

Creating the code converter started with analyzing the glyph assigned to each code point. Currently, there is no published specification for Menksoft code. However, the glyph shapes and code points can be observed by opening a Menksoft font with a font editing software. This was shown in Figure 5 previously. The Menksoft glyphs range from code points U+E234 to U+E34F. These codes are all in the Private Use Area (PUA) of Unicode. Punctuation comes first followed by glyphs for each of the letters of the Mongolian alphabet. The progression of the letters follows the same order as Mongolian Unicode. However, glyph forms are encoded for each position in the word: isolate, initial, final, and medial. Since Menksoft encodes the letter in addition to the glyph shape, there are identical glyphs for letters that are written the same way. For example, see Figure 7 for a comparison of the A and E glyphs.

Each glyph has a one-to-one mapping for letters. That is, a single glyph never maps to two

letters, even for consonant-vowel pairs that overlap when in context. For example, see Figure 8 for the B + U combination.

Some glyph ranges for a single letter sometimes appear to have identical glyphs. However, there are often minor differences that make one glyph more appropriate for a given context. Sometimes the differences are not even visible at all in the print style font (Qagan/white font), but appear in a handwriting style font. Such a case is apparent in Figure 9. Both glyphs for N are in the initial position. However, the A “tooth” that follows it requires a slanting finish to N, while in the second example, the O that follows has a straight stem, so N must also finish with a straight stem.

In this way, each glyph was analyzed for the best appearance in context of other glyphs. The PUA code point was then given an appropriate name in the Dart source code corresponding to the form specifications in GB/T 25914-2023. Listing 1 shows the named variables for the N glyphs. All are static const int variables.

Listing 1: Named Menksoft codes

```
naStart = 0 xE2B1;
naIsol = 0 xE2B3 ;
naIsolFvs1 = 0 xE2B4 ;
naInitTooth = 0 xE2B1 ;
naInitStem = 0 xE2B3 ;
naInitFvs1Tooth = 0 xE2B2 ;
naInitFvs1Stem = 0 xE2B4 ;
naMediTooth = 0 xE2B8 ;
naMediStem = 0 xE2BA ;
naMediFvs1Tooth = 0 xE2B7 ;
naMediFvs1Stem = 0 xE2B9 ;
naMediFvs1Mvs = 0 xE2B6 ;
naMediFvs2Mvs = 0 xE2B5 ;
naMediNg = 0 xE2C0 ;
naMediFvs1Ng = 0 xE2BF ;
naFina = 0 xE2B5 ;
naFinaFvs1 = 0 xE2B6 ;
```

The order follows the GB/T 25914-2023 specification order rather than the order the glyphs appear in the font.

Encoder Implementation

Words in Mongolian do not affect the shape of adjacent words. Thus, each word can be processed independently.

Suffixes appear to be separate words because they start with a space. However, in both Unicode and Menksoft code, that space is a special character. In Unicode, it is encoded with an MVS connector between the word and the suffix. (In the past, this was handled by a Narrow No-Break Space (NNBSP) character, but this is now deprecated.) Therefore, when converting between Menksoft code and Unicode (in either direction), the first step is to extract out a single word with its suffixes.

The strategy for converting Menksoft code to Unicode is slightly different than converting Unicode to Menksoft code, so these will be described separately in the sections below.

Converting Menksoft Code to Unicode

When converting a Menksoft encoded word to Unicode, the first thing to check is whether it is a predefined code sequence. Many two-letter words have a set encoding that is defined in GB/T 25914-2023. In addition to two-letter words, suffixes also have a predefined encoding. Finally, the question particles uu/üü and the word buu (not) have a fixed encoding. If the word to be converted matches any of these hard-coded strings, the preset encoding can be returned early without contextually analyzing the word and converting character by character.

If no preset string was found, the Menksoft characters are converted one by one starting at the beginning of the input string. Since most Menksoft codes are grouped together by letter (N is an exception since it appears on both sides of NG in the encoding), it's possible to route the Menksoft code to a proper letter-handling function by a simple check of which range it falls in.

From there, the character is further subdivided and processed according to its position within the word (isolate, initial, medial, and final). Then, based on the glyph shape, formatting characters such as FVS, MVS or NIRUGU are added to the Unicode character. This is repeated for every Menksoft code, and in this way the Unicode string for the word is built up. Each word is in turn concatenated to form the fully converted text string.

Converting Unicode to Menksoft Code

The process of converting Unicode to Menksoft code contains a few differences than going the other direction. First of all, Menksoft code only encodes glyphs for what are sometimes referred to as Hudum Mongolian. This term is used to distinguish the traditional Mongolian script from other scripts that are derived from it but still included in the Unicode Mongolian block. These are Todo, Sibe, Manchu, Buryat, and extensions for Sanskrit and Tibetan (The Unicode Consortium, 2025). Because Menksoft code does not encode for these scripts, any word that contains Unicode characters for these scripts is not converted in the conversion process.

Part of the process of choosing the correct glyph for a particular Mongolian Unicode character is making sure that one glyph flows smoothly into the next glyph. This was demonstrated in Figure 9 earlier for NA and NO. The flow transition between glyphs can be characterized as a tooth, a stem, or round.

“Tooth” refers to any character that starts by slanting to the left (when viewed in vertical orientation). In addition to the A/E tooth glyph, the left slanting characters also include characters like T/D, R, and W. Since characters like these begin with a slant to the left, the preceding glyph needs to also end with a slant to the left. Conversely, characters that begin with a straight stem (such as CH, O/U, and B) call for a preceding character that also ends with a straight stem.

Another shaping consideration that must be taken into account is when the glyph shape below a character is “round”. These round characters include the feminine Q and G forms. The reason these are important is that M, L, and NG take special shapes when preceding feminine Q/G. Since it is the subsequent character that affects the shape of the preceding character, it is algorithmically easier to start at the end of the word and work to the front when doing the conversion. Before returning the final converted string, the glyph codes can be reversed to provide the string in the correct order.

Beyond the special considerations mentioned above, the conversion process is mainly a matter of choosing the correct glyph based on the location in the word, whether there is an FVS character involved, and sometimes also the gender of the word.

The MVS character has also taken a greater role after NNBSP was deprecated. Previously,

NNBSP was used to attach suffixes to the root word. However, this role has now been replaced with MVS. It is also permissible to attach several words together using MVS. Due to this, when the converter scans the input text, it must differentiate between the original usage of MVS (a vowel separator for words ending in A/E) and the suffix or word separator usage.

Strategies on Preserving Glyph Shapes

The purpose of Menksoft encoding in this proposal is to retain the visual appearance of the Mongolian script over time. The strategy then is to take user input or other Mongolian text input written in Unicode and convert it to Menksoft encoded data before storing it as a text file or in a database. When the user wishes to export that text out of the application at some future date, it is then converted back to Unicode. This is illustrated in Figure 10.

Conversion to Other Encoding Systems

Although the converter described in this paper only converts between GB/T 25914-2023 Unicode and Menksoft code, other phonetic and graphetic encoding systems can be added. The contextual rules required to convert between Unicode and Menksoft code are highly complex. (The GB/T 25914-2023 document is 106 pages long!) However, it is relatively easy to convert between different versions of Unicode or different graphetic encoding systems such as CMs or ZCode. The mongol_code converter can then act as a bridge to convert between any two phonetic and graphetic encodings once the relatively simpler Unicode-to-Unicode or Graphetic-to-Graphetic converters are written. The caveat is that phonetic information is not available when converting from certain graphetic encoding systems such as CMs code or ZCode to Unicode.

Results and Discussion

The mongol_code converter was written in the Dart language so that applications using the cross-platform Flutter framework can use it. The package and source code are available in public, open-source repositories (Author, 2025a, 2025b).

Figure 11 shows a simple example of how the mongol_code package can be used to convert from Menksoft code to Unicode and from Unicode to

Menksoft code. The sample here shows single word conversions, but the package is also capable of converting much longer strings of text.

Text Coverage

The project contains 1,401 unit tests. These include conversion tests for whole words, suffixes, punctuation, isolates, and control characters. One of the tests takes a word bank of 17,721 Menksoft encoded words and converts it to Unicode and back, ensuring that the final Menksoft code is identical to the original code. One hundred percent of the tests are passing.

Test coverage measures the percentage of the total number of lines of code (LOC) in the package that are hit when running the unit tests. Of the 2,884 LOC in mongol_code, the tests covered 88.7% of them.

Usage

The strategy of storing Mongolian text in Menksoft code is currently being tested in another project where maintaining the visual presentation form of every word over time is critical. However, more work is needed before publicly releasing that project.

Limitations

Although Menksoft code has the advantage of being stable over time and stores both the character shape and pronunciation, there are still some limitations:

- There are many ways that Menksoft code can be incorrectly encoded. Encoding the wrong letter will also result in the wrong Unicode when converted.
- Todo Mongolian and other scripts in the Unicode Mongolian block are not supported.
- It was not possible to obtain a large data set of real-world Mongolian text written with the GB/T 25914-2023 specifications, so Unicode-Menksoft-Unicode conversion tests were not run.
- Converting to Menksoft code doesn't solve problems like search engine indexing, sorting, searching for text in a document, or phishing attacks. Menksoft code is not a long-term solution for traditional Mongolian digitization, only a temporary

solution to preserve visual presentation form while other long-term solutions are worked out.

- Test coverage is not yet 100%, meaning there is potential for some conversions to fail or for regression bugs appear in the future.
- The package did not address certain rare edge cases described in GB/T 25914-2023.

Conclusion

Unicode is making progress in adopting precise standards for encoding every character variation. However, there is still not unanimous agreement across governments, companies, and font developers. The specifications continue to evolve. Until an unambiguous and universally agreed upon standard is adopted, app developers can ensure visual text fidelity over time by storing traditional Mongolian text using a graphetic model. Some graphetic models lose phonetic information when converted from Unicode and back, but Menksoft code retains this information. The `mongol_code` package accurately converts between Unicode and Menksoft code. Since

Menksoft code has other disadvantages (such as enabling multiple ways of writing the same word), this is seen as a temporary measure until the Unicode ambiguities are solved. That day will be highly welcomed.

References

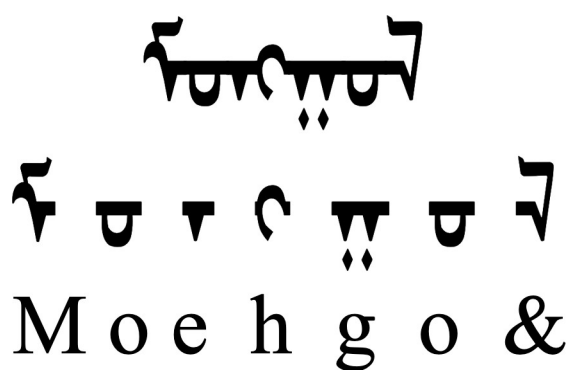
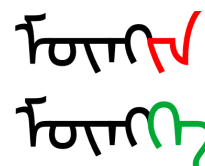
- Anderson, D. (2020, April 23). *Notes on informal meeting on Mongolian with Chuck Namujila* (Mongolian Working Group Document No. MWG4-8). Unicode Consortium. Retrieved December 30, 2025, from <https://www.unicode.org/mwg/mwg4docs/mwg4-8NotesOnInformalMeeting23April20-Anderson.pdf>
- Author. (2025a). [software package blinded for peer review].
- Author. (2025b). [source code repository blinded for peer review].
- Batjargal, B., Khaltarkhuu, G., Kimura, F., & Maeda, A. (2011). A study of traditional mongolian script encodings and rendering: Use of unicode in opentype fonts. *Int. J. Asian Lang. Process.*, 21(1), 23–44.
- Bolorsoft. (2019). *The refined phonetic model: The most convenient solution for recovering the current Mongolian encoding* (Report No. MWG/3-N7). Retrieved December 27, 2025, from <https://www.unicode.org/mwg/mwg3docs/mwg3-7RefinedPhoneticModelR.pdf>
- Ishida, R. (2020, September). Mongolian variant forms. *W3C Internationalization*. Retrieved December 26, 2025, from <https://r12a.github.io/mongolian-variants/mongolian-variant-forms.pdf>
- Jiang, K. (2025). *Progress report on UTN #57: Encoding and shaping of the Mongolian script* (Report No. L2/25-140). Retrieved December 27, 2025, from <https://www.unicode.org/L2/L2025/25140-report-utn57.pdf>
- Kullmann, R., & Tserenpil, D. (2015). *Mongolian grammar* (5th ed.). Kullnom Verlag.
- Namujila, C. (2020). *Preliminary proposal for encoding the mongolian usgiin solution in the smp of the ucs* (Report No. MWG/4-N4). Unicode Consortium. Retrieved December 26, 2025, from https://www.unicode.org/mwg/mwg4docs/mwg4-4PreliminaryProposal-ChuckN_0214.pdf
- State Administration for Market Regulation & Standardization Administration of the People's Republic of China. (2023). Information technology—traditional Mongolian nominal characters, presentation characters and use rules of controlling characters.
- Teneg, T. (2025). The digitisation odyssey of the mongolian script in unicode. *Inner Asia*, 27(2), 277–313.
- The Unicode Consortium. (2025). The Unicode standard, version 17.0 — Mongolian (u+1800–u+18af) [Accessed: 2025-12-27]. <https://www.unicode.org/charts/PDF/U1800.pdf>
- Unicode Mongolian Working Group. (2019). *Mongolian (section 13.5)* (Report No. MWG/3-2R). Unicode Consortium. Retrieved December 26, 2025, from <https://www.unicode.org/mwg/mwg3docs/mwg3-2UnicodeV12MongolianBlockR.pdf>

Table 1. Mongolian Unicode Basic Letters (18201842)

Hex	Ltr	Name	Hex	Ltr	Name
1820	ᠠ	A	1832	ᠲ	TA
1821	ᠡ	E	1833	ᠳ	DA
1822	ᠢ	I	1834	ᠴ	CHA
1823	ᠣ	O	1835	ᠵ	JA
1824	ᠤ	U	1836	ᠶ	YA
1825	ᠥ	OE	1837	ᠷ	RA
1826	ᠦ	UE	1838	ᠸ	WA
1827	ᠦ	EE	1839	ᠰ	FA
1828	ᠨ	NA	183A	ᠬ	KA
1829	ᠠᠩ	ANG	183B	ᠬᠠ	KHA
182A	ᠪ	BA	183C	ᠲᠠ	TSA
182B	ᠫ	PA	183D	ᠵᠠ	ZA
182C	ᠬᠠ	QA	183E	ᠬᠠᠠ	HAA
182D	ᠭᠠ	GA	183F	ᠵᠠ	ZRA
182E	ᠮᠠ	MA	1840	ᠯᠠ	LHA
182F	ᠯᠠ	LA	1841	ᠵᠠ	ZHI
1830	ᠰᠠ	SA	1842	ᠴᠠ	CHI
1831	ᠰᠠ	HA			

Table 2. Test Coverage Results for mongol_code.

Directory	Coverage (%)	Lines	Hit
menksoft/	85.5	1,566	1,339
shared/	100.0	5	5
unicode/	92.5	1,313	1,215
Total	88.7	2,884	2,559

**Figure 1.** Seven FVS1-3 variations on medial GA**Figure 2.** Masculine medial dotted GA as one graphetic code or two**Figure 3.** CMs mapping of glyphs to Latin codes to write “Mongol”**Figure 6.** Refined model encoding (top) rendered by Noto Sans Mongolian font

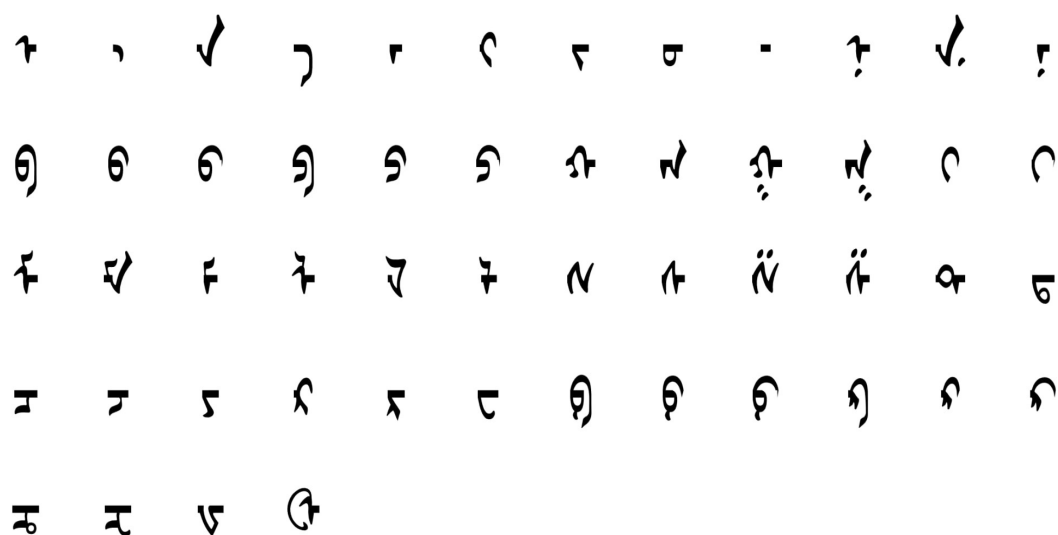


Figure 4. 52 Zcode glyphs

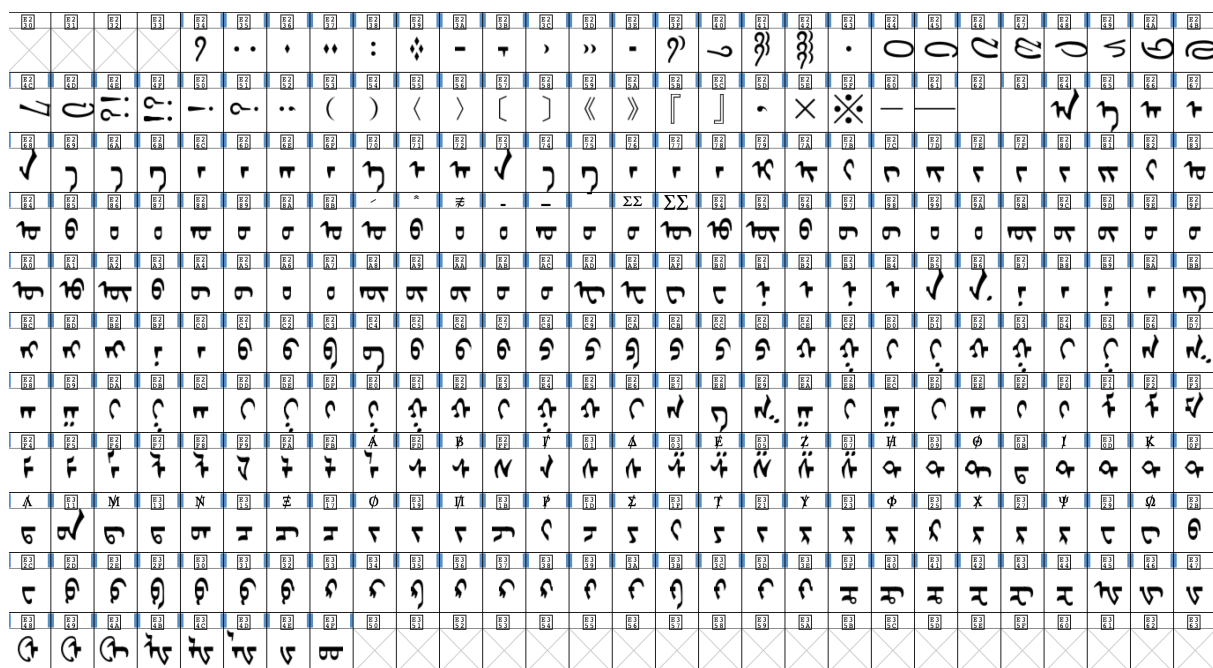


Figure 5. Menksoft glyphs in the PUA region