

⇒ Implementing Javanese ⇐

Norbert Lindenberg
Version 1, 2022-09-13

This document assists in implementing the Javanese script in fonts, rendering systems, keyboards, and other software by providing information that complements information in *The Unicode Standard*.




Contents

1	Reference materials	2
2	Script identification	2
3	Special characters	3
4	Conjunct forms	3
5	Encoding order of orthographic syllable components	6
6	Rendering	9
7	Keyboards	12
8	Line breaking	14
9	Acknowledgments	14

© 2022 Lindenberg Software LLC. Norbert Lindenberg, Lindenberg Software LLC, and the Unicode Consortium make no expressed or implied warranty of any kind, and assume no liability for errors or omissions. No liability is assumed for incidental and consequential damages in connection with or arising out of the use of the information or programs contained in or accompanying this technical note. The Unicode [Terms of Use](#) apply.

3 Special characters

The following characters are mentioned multiple times in this document:

- U+A9C0  JAVANESE PANGKON is the Javanese virama. This character is used both by itself, to represent the visible vowel killer, and as the first part of the character sequences used to encode the conjunct forms, or *pasangan*, of consonants, vocalic liquids, and independent vowels. This document uses PANGKON to denote the code point, and *pangkon* specifically for the visible vowel killer.
- U+A9BF  JAVANESE CONSONANT SIGN CAKRA is a medial consonant with complex rendering: In its normal form, it wraps around the bottom and left side of the base glyph or conjunct form it is attached to, and may form ligatures with other characters. However, in cases where this gets too complicated, a simplified form of *cakra* to the left of the base, , can be used instead.

4 Conjunct forms

The following table shows the base consonants, vocalic liquids, and independent vowels (together the letters of the Javanese script) and their conjunct forms.

Base character	Base form	Conjunct form
Consonants		
<i>ka</i>	ꦏꦲ	ꦏꦲꦩꦠꦺꦤ꧀
<i>ka sasak (qa)</i>	ꦏꦲꦱꦱꦏꦺꦴ	ꦏꦲꦱꦱꦏꦺꦴꦩꦠꦺꦤ꧀
<i>ka murda (kha)</i>	ꦏꦲꦩꦸꦂꦢ	ꦏꦲꦩꦸꦂꦢꦩꦠꦺꦤ꧀
<i>ga</i>	ꦒ	ꦒꦩꦠꦺꦤ꧀
<i>ga murda (gha)</i>	ꦒꦲꦩꦸꦂꦢ	ꦒꦲꦩꦸꦂꦢꦩꦠꦺꦤ꧀
<i>nga</i>	ꦒꦲ	ꦒꦲꦩꦠꦺꦤ꧀
<i>ca</i>	ꦕ	ꦕꦩꦠꦺꦤ꧀

Base character	Base form	Conjunct form
<i>ca murda (cha)</i>	ꦱꦩ	ꦱꦩꦠ
<i>ja</i>	ꦗ	ꦗꦠ
<i>nya murda (jnya)</i>	ꦗꦚ	ꦗꦚꦠ
<i>ja mahaprana (jha)</i>	ꦗꦲ	ꦗꦲꦠ
<i>nya</i>	ꦚꦤ	ꦗꦲꦚꦤꦠ
<i>tta</i>	ꦠꦠ	ꦠꦠꦠ
<i>tta mahaprana (ttha)</i>	ꦠꦠꦲ	ꦠꦠꦠꦠ
<i>dda</i>	ꦢꦢ	ꦢꦢꦠ
<i>dda mahaprana (ddha)</i>	ꦢꦢꦲ	ꦢꦢꦠꦠ
<i>na murda (nna)</i>	ꦚꦤꦤ	ꦚꦤꦤꦠ
<i>ta</i>	ꦠꦤ	ꦠꦠꦠꦠ
<i>ta murda (tha)</i>	ꦠꦲ	ꦠꦠꦠꦠ
<i>da</i>	ꦢꦤ	ꦢꦠꦠ
<i>da mahaprana (dha)</i>	ꦢꦠꦲ	ꦢꦠꦠꦠ
<i>na</i>	ꦚꦤ	ꦠꦠꦠꦠ
<i>pa</i>	ꦥ	ꦥꦠ
<i>pa murda (pha)</i>	ꦥꦲ	ꦥꦠꦠ
<i>ba</i>	ꦧꦤ	ꦧꦠꦠꦠꦠ
<i>ba murda (bha)</i>	ꦧꦲ	ꦧꦠꦠꦠ

Base character	Base form	Conjunct form
<i>ma</i>	ꦩ	ꦩꦺ
<i>ya</i>	ꦪ	ꦪꦺ
<i>ra</i>	ꦫ	ꦫꦺ
<i>ra agung</i>	ꦫꦸ	ꦫꦸꦺ
<i>la</i>	ꦭ	ꦭꦺ
<i>wa</i>	ꦮ	ꦮꦺ
<i>sa murda (sha)</i>	ꦱ	ꦱꦺ
<i>sa mahaprana (ssa)</i>	ꦱꦱ	ꦱꦱꦺ
<i>sa</i>	ꦱ	ꦱꦺ
<i>ha</i>	ꦲ	ꦲꦺ
Vocalic liquids		
<i>pa cerek (vocalic r)</i>	ꦱꦫ	ꦱꦫꦺ
<i>nga lelet (vocalic l)</i>	ꦱꦭ	ꦱꦭꦺ
<i>nga lelet raswadi (vocalic ll)</i>	ꦱꦭꦭ	ꦱꦭꦭꦺ
Independent vowels		
<i>a</i>	ꦲꦩ	ꦲꦩꦺ
<i>i kawi</i>	ꦲꦶ	ꦲꦶꦏꦮ
<i>i</i>	ꦲꦶ	ꦲꦶꦺ
<i>ii</i>	ꦲꦶꦶ	ꦲꦶꦶꦺ
<i>u</i>	ꦲꦸ	ꦲꦸꦺ

Base character	Base form	Conjunct form
<i>e</i>	ꦺ	ꦥꦺꦴ
<i>ai</i>	ꦲꦶ	ꦥꦲꦶꦴ
<i>o</i>	ꦺꦴ	ꦥꦺꦴꦴ

Conjunct forms are encoded by preceding the letter with PANGKON. The formation of conjunct forms can be prevented by inserting U+200C ZERO WIDTH NON-JOINER between PANGKON and the letter. The second conjunct forms of *nya* and *ba* are stylistic variants, which should be implemented as font features. The Indonesian standard SNI 9047:2021 shows these stylistic variants encoded as sequences of PANGKON, U+200D ZERO WIDTH JOINER, and consonant; this has the disadvantage, however, of breaking conjunct formation in fonts that do not specifically support the variant glyphs.

5 Encoding order of orthographic syllable components

Javanese, like other Brahmic scripts, has features where phonetic and visual order of characters within an orthographic syllable may differ: Dependent vowels and a medial consonant to the left of the base, and bindus and a final consonant above the base or a spacing conjunct form rather than above a right-side dependent vowel. For example, the components of the syllable ꦏꦺꦴꦂ are generally pronounced in the order ꦏꦺ *n*, ꦺꦴꦂ *o*, ꦴꦂ *r* → *nor*. The script's encoding uses a primarily phonetic ordering. In addition, the conjunct forms of letters are encoded as sequences of PANGKON and the respective letter; such sequences should generally not be broken up. Characters and conjunct forms within an orthographic syllable should be encoded in the relative order shown in the following table. The *encoding* column uses the syntax defined in the section A.2 Extended BNF of The Unicode Standard. The *count* column states how many characters of each class occur in real-life orthographic syllables.

Class	Characters	Encoding	Count
<i>consonant, vocalic liquid, independent vowel, number, generic base</i>	<p> ᮘ ᮙ ᮚ ᮛ ᮜ ᮝ ᮞ ᮟ ᮠ ᮡ ᮢ ᮣ ᮤ ᮥ ᮦ ᮧ ᮨ ᮩ ᮪ ᮫ ᮬ ᮭ ᮮ ᮯ ᮰ ᮱ ᮲ ᮳ ᮴ ᮵ ᮶ ᮷ ᮸ ᮹ ᮺ ᮻ ᮼ ᮽ ᮾ ᮿ ᯀ ᯁ ᯂ ᯃ ᯄ ᯅ ᯆ ᯇ ᯈ ᯉ ᯊ ᯋ ᯌ ᯍ ᯎ ᯏ ᯐ ᯑ ᯒ ᯓ ᯔ ᯕ ᯖ ᯗ ᯘ ᯙ ᯚ ᯛ ᯜ ᯝ ᯞ ᯟ ᯠ ᯡ ᯢ ᯣ ᯤ ᯥ ᯦ ᯧ ᯨ ᯩ ᯪ ᯫ ᯬ ᯭ ᯮ ᯯ ᯰ ᯱ ᯲ ᯳ ᯴ ᯵ ᯶ ᯷ ᯸ ᯹ ᯺ ᯻ ᯼ ᯽ ᯾ ᯿ ᰀ ᰁ ᰂ ᰃ ᰄ ᰅ ᰆ ᰇ ᰈ ᰉ ᰊ ᰋ ᰌ ᰍ ᰎ ᰏ ᰐ ᰑ ᰒ ᰓ ᰔ ᰕ ᰖ ᰗ ᰘ ᰙ ᰚ ᰛ ᰜ ᰝ ᰞ ᰟ ᰠ ᰡ ᰢ ᰣ ᰤ ᰥ ᰦ ᰧ ᰨ ᰩ ᰪ ᰫ ᰬ ᰭ ᰮ ᰯ ᰰ ᰱ ᰲ ᰳ ᰴ ᰵ ᰶ ᰷ ᰸ ᰹ ᰺ ᰻ ᰼ ᰽ ᰾ ᰿ ᱀ ᱁ ᱂ ᱃ ᱄ ᱅ ᱆ ᱇ ᱈ ᱉ ᱊ ᱋ ᱌ ᱍ ᱎ ᱏ ᱐ ᱑ ᱒ ᱓ ᱔ ᱕ ᱖ ᱗ ᱘ ᱙ ᱚ ᱛ ᱜ ᱝ ᱞ ᱟ ᱠ ᱡ ᱢ ᱣ ᱤ ᱥ ᱦ ᱧ ᱨ ᱩ ᱪ ᱫ ᱬ ᱭ ᱮ ᱯ ᱰ ᱱ ᱲ ᱳ ᱴ ᱵ ᱶ ᱷ ᱸ ᱹ ᱺ ᱻ ᱼ ᱽ ᱾ ᱿ ᳀ ᳁ ᳂ ᳃ ᳄ ᳅ ᳆ ᳇ ᳈ ᳉ ᳊ ᳋ ᳌ ᳍ ᳎ ᳏ ᳐ ᳑ ᳒ ᳓ ᳔ ᳕ ᳖ ᳗ ᳘ ᳙ ᳚ ᳛ ᳜ ᳝ ᳞ ᳟ ᳠ ᳡ ᳢ ᳣ ᳤ ᳥ ᳦ ᳧ ᳨ ᳩ ᳪ ᳫ ᳬ ᳭ ᳮ ᳯ ᳰ ᳱ ᳲ ᳳ ᳴ ᳵ ᳶ ᳷ ᳸ ᳹ ᳺ ᳻ ᳼ ᳽ ᳾ ᳿ ᴀ ᴁ ᴂ ᴃ ᴄ ᴅ ᴆ ᴇ ᴈ ᴉ ᴊ ᴋ ᴌ ᴍ ᴎ ᴏ ᴐ ᴑ ᴒ ᴓ ᴔ ᴕ ᴖ ᴗ ᴘ ᴙ ᴚ ᴛ ᴜ ᴝ ᴞ ᴟ ᴠ ᴡ ᴢ ᴣ ᴤ ᴥ ᴦ ᴧ ᴨ ᴩ ᴪ ᴫ ᴬ ᴭ ᴮ ᴯ ᴰ ᴱ ᴲ ᴳ ᴴ ᴵ ᴶ ᴷ ᴸ ᴹ ᴺ ᴻ ᴼ ᴽ ᴾ ᴿ ᵀ ᵁ ᵂ ᵃ ᵄ ᵅ ᵆ ᵇ ᵈ ᵉ ᵊ ᵋ ᵌ ᵍ ᵎ ᵏ ᵐ ᵑ ᵒ ᵓ ᵔ ᵕ ᵖ ᵗ ᵘ ᵙ ᵚ ᵛ ᵜ ᵝ ᵞ ᵟ ᵠ ᵡ ᵢ ᵣ ᵤ ᵥ ᵦ ᵧ ᵨ ᵩ ᵪ ᵫ ᵬ ᵭ ᵮ ᵯ ᵰ ᵱ ᵲ ᵳ ᵴ ᵵ ᵶ ᵷ ᵸ ᵹ ᵺ ᵻ ᵼ ᵽ ᵾ ᵿ ᶀ ᶁ ᶂ ᶃ ᶄ ᶅ ᶆ ᶇ ᶈ ᶉ ᶊ ᶋ ᶌ ᶍ ᶎ ᶏ ᶐ ᶑ ᶒ ᶓ ᶔ ᶕ ᶖ ᶗ ᶘ ᶙ ᶚ ᶛ ᶜ ᶝ ᶞ ᶟ ᶠ ᶡ ᶢ ᶣ ᶤ ᶥ ᶦ ᶧ ᶨ ᶩ ᶪ ᶫ ᶬ ᶭ ᶮ ᶯ ᶰ ᶱ ᶲ ᶳ ᶴ ᶵ ᶶ ᶷ ᶸ ᶹ ᶺ ᶻ ᶼ ᶽ ᶾ ᶿ ᷀ ᷁ ᷂ ᷃ ᷄ ᷅ ᷆ ᷇ ᷈ ᷉ ᷊ ᷋ ᷌ ᷍ ᷎ ᷏ ᷐ ᷑ ᷒ ᷓ ᷔ ᷕ ᷖ ᷗ ᷘ ᷙ ᷚ ᷛ ᷜ ᷝ ᷞ ᷟ ᷠ ᷡ ᷢ ᷣ ᷤ ᷥ ᷦ ᷧ ᷨ ᷩ ᷪ ᷫ ᷬ ᷭ ᷮ ᷯ ᷰ ᷱ ᷲ ᷳ ᷴ ᷵ ᷶ ᷷ ᷸ ᷹ ᷺ ᷻ ᷼ ᷽ ᷾ ᷿ Ḁ ḁ Ḃ ḃ Ḅ ḅ Ḇ ḇ Ḉ ḉ Ḋ ḋ Ḍ ḍ Ḏ ḏ Ḑ ḑ Ḓ ḓ Ḕ ḕ Ḗ ḗ Ḙ ḙ Ḛ ḛ Ḝ ḝ Ḟ ḟ Ḡ ḡ Ḣ ḣ Ḥ ḥ Ḧ ḧ Ḩ ḩ Ḫ ḫ Ḭ ḭ Ḯ ḯ Ḱ ḱ Ḳ ḳ Ḵ ḵ Ḷ ḷ Ḹ ḹ Ḻ ḻ Ḽ ḽ Ḿ ḿ Ṁ ṁ Ṃ ṃ Ṅ ṅ Ṇ ṇ Ṉ ṉ Ṋ ṋ Ṍ ṍ Ṏ ṏ Ṑ ṑ Ṓ ṓ Ṕ ṕ Ṗ ṗ Ṙ ṙ Ṛ ṛ Ṝ ṝ Ṟ ṟ Ṡ ṡ Ṣ ṣ Ṥ ṥ Ṧ ṧ Ṩ ṩ Ṫ ṫ Ṭ ṭ Ṯ ṯ Ṱ ṱ Ṳ ṳ Ṵ ṵ Ṷ ṷ Ṹ ṹ Ṻ ṻ Ṽ ṽ Ṿ ṿ Ṱ ṱ Ṳ ṳ Ṵ ṵ Ṷ ṷ Ṹ ṹ Ṻ ṻ Ṽ ṽ Ṿ ṿ </p>	[U+A984..U+A9B2, U+A9D0..U+A9D9, U+25CC]	1
<i>nukta</i>	◌̣	U+A9B3	0 to 1
<i>conjunct form</i>	<p> ᮘᮙ ᮙᮚ ᮚᮛ ᮛᮜ ᮜᮝ ᮝᮞ ᮞᮟ ᮟᮠ ᮠᮡ ᮡᮢ ᮢᮣ ᮣᮤ ᮤᮥ ᮥᮦ ᮦᮧ ᮧᮨ ᮨᮩ ᮩ᮪ ᮪᮫ ᮫ᮬ ᮬᮭ ᮭᮮ ᮮᮯ ᮯ᮰ ᮰᮱ ᮱᮲ ᮲᮳ ᮳᮴ ᮴᮵ ᮵᮶ ᮶᮷ ᮷᮸ ᮸᮹ ᮹ᮺ ᮺᮻ ᮻᮼ ᮼᮽ ᮽᮾ ᮾᮿ ᯀᯁ ᯁᯂ ᯂᯃ ᯃᯄ ᯄᯅ ᯅᯆ ᯆᯇ ᯇᯈ ᯈᯉ ᯉᯊ ᯊᯋ ᯋᯌ ᯌᯍ ᯍᯎ ᯎᯏ ᯏᯐ ᯐᯑ ᯑᯒ ᯒᯓ ᯓᯔ ᯔᯕ ᯕᯖ ᯖᯗ ᯗᯘ ᯘᯙ ᯙᯚ ᯚᯛ ᯛᯜ ᯜᯝ ᯝᯞ ᯞᯟ ᯟᯠ ᯠᯡ ᯡᯢ ᯢᯣ ᯣᯤ ᯤᯥ ᯥ᯦ ᯦ᯧ ᯧᯨ ᯨᯩ ᯩᯪ ᯪᯫ ᯫᯬ ᯬᯭ ᯭᯮ ᯮᯯ ᯯᯰ ᯰᯱ ᯱ᯲ ᯲᯳ ᯳᯴ ᯴᯵ ᯵᯶ ᯶᯷ ᯷᯸ ᯸᯹ ᯹᯺ ᯺᯻ ᯻᯼ ᯼᯽ ᯽᯾ ᯾᯿ ᰀᰁ ᰁᰂ ᰂᰃ ᰃᰄ ᰄᰅ ᰅᰆ ᰆᰇ ᰇᰈ ᰈᰉ ᰉᰊ ᰊᰋ ᰋᰌ ᰌᰍ ᰍᰎ ᰎᰏ ᰏᰐ ᰐᰑ ᰑᰒ ᰒᰓ ᰓᰔ ᰔᰕ ᰕᰖ ᰖᰗ ᰗᰘ ᰘᰙ ᰙᰚ ᰚᰛ ᰛᰜ ᰜᰝ ᰝᰞ ᰞᰟ ᰟᰠ ᰠᰡ ᰡᰢ ᰢᰣ ᰣᰤ ᰤᰥ ᰥᰦ ᰦᰧ ᰧᰨ ᰨᰩ ᰩᰪ ᰪᰫ ᰫᰬ ᰬᰭ ᰭᰮ ᰮᰯ ᰯᰰ ᰰᰱ ᰱᰲ ᰲᰳ ᰳᰴ ᰴᰵ ᰵᰶ ᰶ᰷ ᰷᰸ ᰸᰹ ᰹᰺ ᰺᰻ ᰻᰼ ᰼᰽ ᰾᰿ ᱀᱁ ᱁᱂ ᱂᱃ ᱃᱄ ᱄᱅ ᱅᱆ ᱆᱇ ᱇᱈ ᱈᱉ ᱉᱊ ᱊᱋ ᱋᱌ ᱌ᱍ ᱍᱎ ᱎᱏ ᱏ᱐ ᱐᱑ ᱑᱒ ᱒᱓ ᱓᱔ ᱔᱕ ᱕᱖ ᱖᱗ ᱗᱘ ᱘᱙ ᱙ᱚ ᱚᱛ ᱛᱜ ᱜᱝ ᱝᱞ ᱞᱟ ᱟᱠ ᱠᱡ ᱡᱢ ᱢᱣ ᱣᱤ ᱤᱥ ᱥᱦ ᱦᱧ ᱧᱨ ᱨᱩ ᱩᱪ ᱪᱫ ᱫᱬ ᱬᱭ ᱭᱮ ᱮᱯ ᱯᱰ ᱰᱱ ᱱᱲ ᱲᱳ ᱳᱴ ᱴᱵ ᱵᱶ ᱶᱷ ᱷᱸ ᱸᱹ ᱹᱺ ᱺᱻ ᱻᱼ ᱼᱽ ᱾᱿ ᳀᳁ ᳁᳂ ᳂᳃ ᳃᳄ ᳄᳅ ᳅᳆ ᳆᳇ ᳇᳈ ᳈᳉ ᳉᳊ ᳊᳋ ᳋᳌ ᳌᳍ ᳍᳎ ᳎᳏ ᳏᳐ ᳐᳑ ᳑᳒ ᳒᳓ ᳓᳔ ᳔᳕ ᳕᳖ ᳖᳗ ᳗᳘ ᳘᳙ ᳙᳚ ᳚᳛ ᳜᳛ ᳜᳝ ᳝᳞ ᳞᳟ ᳟᳠ ᳠᳡ ᳡᳢ ᳢᳣ ᳣᳤ ᳤᳥ ᳥᳦ ᳦᳧ ᳧᳨ ᳨ᳩ ᳩᳪ ᳪᳫ ᳫᳬ ᳬ᳭ ᳭ᳮ ᳮᳯ ᳯᳰ ᳰᳱ ᳱᳲ ᳲᳳ ᳳ᳴ ᳴ᳵ ᳵᳶ ᳶ᳷ ᳷᳸ ᳸᳹ ᳹ᳺ ᳺ᳻ ᳻᳼ ᳼᳽ ᳾᳿ ᴀᴁ ᴁᴂ ᴂᴃ ᴃᴄ ᴄᴅ ᴅᴆ ᴆᴇ ᴇᴈ ᴈᴉ ᴉᴊ ᴊᴋ ᴋᴌ ᴌᴍ ᴍᴎ ᴎᴏ ᴏᴐ ᴐᴑ ᴑᴒ ᴒᴓ ᴓᴔ ᴔᴕ ᴕᴖ ᴖᴗ ᴗᴘ ᴘᴙ ᴙᴚ ᴚᴛ ᴛᴜ ᴜᴝ ᴝᴞ ᴞᴟ ᴠᴡ ᴡᴢ ᴢᴣ ᴣᴤ ᴤᴥ ᴥᴦ ᴦᴧ ᴧᴨ ᴨᴩ ᴩᴪ ᴪᴫ ᴫᴬ ᴬᴭ ᴭᴮ ᴮᴯ ᴯᴰ ᴰᴱ ᴱᴲ ᴲᴳ ᴳᴴ ᴴᴵ ᴵᴶ ᴶᴷ ᴷᴸ ᴸᴹ ᴹᴺ ᴺᴻ ᴻᴼ ᴼᴽ ᴾᴿ ᵀᵁ ᵁᵂ ᵂᵃ ᵃᵄ ᵄᵅ ᵅᵆ ᵆᵇ ᵇᵈ ᵈᵉ ᵉᵊ ᵊᵋ ᵋᵌ ᵌᵍ ᵍᵎ ᵎᵏ ᵏᵐ ᵑᵒ ᵒᵓ ᵓᵔ ᵔᵕ ᵕᵖ ᵖᵗ ᵗᵘ ᵘᵙ ᵙᵚ ᵚᵛ ᵛᵜ ᵜᵝ ᵞᵟ ᵠᵡ ᵡᵢ ᵢᵣ ᵣᵤ ᵤᵥ ᵥᵦ ᵦᵧ ᵧᵨ ᵨᵩ ᵩᵪ ᵫᵫ ᵬᵭ ᵭᵮ ᵮᵯ ᵯᵰ ᵰᵱ ᵱᵲ ᵲᵳ ᵳᵴ ᵴᵵ ᵵᵶ ᵶᵷ ᵷᵸ ᵸᵹ ᵹᵺ ᵺᵻ ᵻᵼ ᵽᵾ ᵿᶀ ᶁᶂ ᶂᶃ ᶃᶄ ᶄᶅ ᶅᶆ ᶆᶇ ᶇᶈ ᶈᶉ ᶉᶊ ᶊᶋ ᶋᶌ ᶌᶍ ᶍᶎ ᶎᶏ ᶏᶐ ᶑᶒ ᶒᶓ ᶓᶔ ᶔᶕ ᶕᶖ ᶖᶗ ᶗᶘ ᶘᶙ ᶙᶚ ᶚᶛ ᶛᶜ ᶜᶝ ᶞᶟ ᶠᶡ ᶢᶣ ᶣᶤ ᶤᶥ ᶥᶦ ᶦᶧ ᶧᶨ ᶨᶩ ᶪᶪ ᶫᶫ ᶬᶭ ᶭᶮ ᶮᶯ ᶯᶰ ᶱᶱ ᶲᶲ ᶳᶳ ᶴᶴ ᶵᶵ ᶶᶶ ᶷᶷ ᶸᶸ ᶹᶹ ᶺᶺ ᶻᶻ ᶼᶼ ᶽᶽ ᶾᶾ ᶿᶿ ᷀᷁ ᷂᷁ ᷂᷃ ᷃᷄ ᷄᷅ ᷅᷆ ᷆᷇ ᷇᷈ ᷈᷉ ᷊᷉ ᷊᷋ ᷋᷌ ᷌᷍ ᷎᷍ ᷎᷏ ᷐᷏ ᷑᷑ ᷒᷒ ᷓᷓ ᷔᷔ ᷕᷕ ᷖᷖ ᷗᷗ ᷘᷘ ᷙᷙ ᷚᷚ ᷛᷛ ᷜᷜ ᷝᷝ ᷞᷞ ᷟᷟ ᷠᷠ ᷡᷡ ᷢᷢ ᷣᷣ ᷤᷤ ᷥᷥ ᷦᷦ ᷧᷧ ᷨᷨ ᷩᷩ ᷪᷪ ᷫᷫ ᷬᷬ ᷭᷭ ᷮᷮ ᷯᷯ ᷰᷰ ᷱᷱ ᷲᷲ ᷳᷳ ᷴᷴ ᷵᷵ ᷶᷶ ᷷᷷ ᷸᷸ ᷹᷹ ᷺᷺ ᷻᷻ ᷼᷼ ᷽᷽ ᷾᷾ ᷿᷿ ḀḀ ḁḁ ḂḂ ḃḃ ḄḄ ḅḅ ḆḆ ḇḇ ḈḈ ḉḉ ḊḊ ḋḋ ḌḌ ḍḍ ḎḎ ḏḏ ḐḐ ḑḑ ḒḒ ḓḓ ḔḔ ḕḕ ḖḖ ḗḗ ḘḘ ḙḙ ḚḚ ḛḛ ḜḜ ḝḝ ḞḞ ḟḟ ḠḠ ḡḡ ḢḢ ḣḣ ḤḤ ḥḥ ḦḦ ḧḧ ḨḨ ḩḩ ḪḪ ḫḫ ḬḬ ḭḭ ḮḮ ḯḯ ḰḰ ḱḱ ḲḲ ḳḳ ḴḴ ḵḵ ḶḶ ḷḷ ḸḸ ḹḹ ḺḺ ḻḻ ḼḼ ḽḽ ḾḾ ḿḿ ṀṀ ṁṁ ṂṂ ṃṃ ṄṄ ṅṅ ṆṆ ṇṇ ṈṈ ṉṉ ṊṊ ṋṋ ṌṌ ṍṍ ṎṎ ṏṏ ṐṐ ṑṑ ṒṒ ṓṓ ṔṔ ṕṕ ṖṖ ṗṗ ṘṘ ṙṙ ṚṚ ṛṛ ṜṜ ṝṝ ṞṞ ṟṟ ṠṠ ṡṡ ṢṢ ṣṣ ṤṤ ṥṥ ṦṦ ṧṧ ṨṨ ṩṩ ṪṪ ṫṫ ṬṬ ṭṭ ṮṮ ṯṯ ṰṰ ṱṱ ṲṲ ṳṳ ṴṴ ṵṵ ṶṶ ṷṷ ṸṸ ṹṹ ṺṺ ṻṻ ṼṼ ṽṽ ṾṾ ṿṿ ṰṰ ṱṱ ṲṲ ṳṳ ṴṴ ṵṵ ṶṶ ṷṷ ṸṸ ṹṹ ṺṺ ṻṻ ṼṼ ṽṽ ṾṾ ṿṿ </p>	U+A9C0 [U+A984..U+A9B2]	0 to 2
<i>nukta for conjunct form</i>	◌̣	U+A9B3	0 to 1
<i>virama</i>	ᮧ	U+A9C0	0 to 1
<i>bottom-left or bottom medial consonant</i>	ᮧᮙ	[U+A9BF, U+A9BD]	0 to 1
<i>bottom-right medial consonant</i>	ᮧᮚ	U+A9BE	0 to 1
<i>left-side dependent vowel</i>	ᮧᮙᮙ	[U+A9BA..U+A9BB]	0 to 1
<i>top dependent vowel</i>	ᮧᮙᮙ	[U+A9B6..U+A9B7, U+A9BC]	0 to 1

Class	Characters	Encoding	Count
<i>bottom dependent vowel</i>	◌◌◌◌	[U+A9B8..U+A9B9]	0 to 1
<i>right-side dependent vowel</i>	◌◌◌◌	[U+A9B4..U+A9B5]	0 to 1
<i>bindu</i>	◌◌◌◌	[U+A980..U+A981]	0 to 1
<i>visarga</i>	◌◌◌◌	U+A983	0 to 1
<i>final consonant</i>	◌◌◌◌	U+A982	0 to 1

If a *virama* (PANGKON) is used without an immediately following letter, it represents the visible vowel killer *pangkon*, which ends the orthographic syllable, and no character shown later in the table should follow. Instead, it may be followed by punctuation, by a line break, by U+200C ZERO WIDTH NON-JOINER to force an orthographic syllable break, or by space when using Western-style word separation.

A *nukta* should only be used after the base character or after a spacing conjunct form, not after a below-base conjunct form.

Some documents in the Kawi and Sanskrit languages use U+A982 ◌ JAVANESE SIGN LAYAR, which normally is a final consonant, as a syllable-initial consonant, also known as *repha*. The syllable ◌◌◌◌ is then pronounced in the order ◌ r, ◌ n, ◌◌ o → *rno*. This does not affect the encoded representation, which keeps treating LAYAR as a final consonant.

The encoding order shown above is the one resulting from the OpenType Universal Shaping Engine's default interpretation of the Unicode character data for Javanese characters, with existing overrides that move U+A9BF ◌ JAVANESE CONSONANT SIGN CAKRA into the same group as U+A9BD ◌ JAVANESE CONSONANT SIGN KERET, and U+A9BE ◌ JAVANESE CONSONANT SIGN PENGKAL out of that group. The engine inserts dotted circles into character sequences with out-of-order marks, so that OpenType fonts only need to deal with correctly ordered characters. Fonts based on technologies other than OpenType, such as Apple Advanced Typography or Graphite, should themselves insert dotted circles into character sequences with out-of-order marks.

Keyboards and other text-generating software should ensure that typed or generated text conforms to this encoding order. Spelling checkers and other text-validating software should use this encoding order in their reference data.

Note that the encoding order shown above differs from that documented in the Unicode Standard up to version 14.0, which was both incomplete and incorrect. The Unicode Technical Committee has removed the information on syllable structure from version 15.0 of the standard.

6 Rendering

The following steps must be taken, in the order given, for each orthographic syllable to achieve orthographically correct rendering:

1. Combine each sequence of PANGKON and an immediately following letter to a conjunct form. However, where a conjunct form would collide with below-base parts of the base, or where too many conjunct forms are stacked, it may be acceptable to show *pangkon* and the letter separately, in which case the letter becomes a base itself.

ꦥꦶ + ꦏꦏ → ꦏꦏꦶ -ka

ꦥꦶ + ꦫꦶ → ꦫꦶꦂ -rě

ꦥꦶ + ꦱꦤ → ꦱꦤꦶ -a

ꦥꦶ + ꦏꦏ + ꦥꦶ + ꦏꦏ → ꦏꦏꦶꦏꦏ or ꦥꦶꦏꦏꦏ -kla

2. Move all pre-base vowels (ꦏꦺ and ꦏꦶꦺ) before the base.

ꦏꦏ + ꦏꦺ → ꦏꦺꦏꦏ ke

3. Convert any *cakra* (ꦏꦿ) to its left-side variant (ꦏꦺ) where necessary to prevent collisions with other glyphs.

ꦏꦏ + ꦏꦿ + ꦥꦶ → ꦏꦏ + ꦏꦺ + ꦥꦶ -krya

4. Move any *cakra* converted to its left-side variant before its base glyph (but after pre-base vowels).

ꦏꦿ + ◌ + ꦏꦿꦶ → ꦏꦿꦶ *krya*

For OpenType fonts using the Universal Shaping Engine, step 1 needs to be implemented in the font; step 2 should be implemented by the shaping engine based on Unicode character data; steps 3 and 4 are implemented by applying the “pref” feature to convert to the left-side form of *cakra*, which should cause the shaping engine to reorder the glyph.

Several additional steps may be taken to improve typography and to support stylistic preferences:

- Several glyphs take on connecting forms when below-base marks attach to them. The combinations may be implemented as ligatures – see the paragraph “Ligatures versus glyph positioning” below.

ꦏꦿꦸ + ꦏꦸ → ꦏꦸꦸ *-ku*

ꦏꦿꦸꦱ + ꦏꦸ → ꦏꦸꦸꦱ *-quu*

ꦏꦿꦸꦠꦺ + ꦏꦸꦠꦺ → ꦏꦸꦸꦠꦺ *-trě*

ꦏꦿꦸꦶ + ꦏꦸꦶ → ꦏꦸꦸꦶ *-lya*

- *Cakra* usually wraps around the bottom and left side of the base glyph or conjunct form it is attached to. When attached to a spacing conjunct form, it only wraps around that, not around the base glyph. *Cakra* also forms ligatures with an attached dependent vowel U+A9B8 ꦏꦸꦶ JAVANESE VOWEL SIGN SUKU. When it gets too complicated, however, falling back to the left-side form of *cakra* is fine. Combinations of the left-side form of *cakra* with spacing conjunct forms are not known, so it is not clear whether in such a combination the *cakra* should be positioned to the left of the base glyph or to the left of the spacing conjunct form.

ꦏꦿ + ꦏꦿ → ꦏꦿꦿ *kra*

ꦏꦿ + ꦏꦿꦶ → ꦏꦿꦶꦶ *nra*

ꦏꦿ + ꦏꦿꦸ + ꦏꦿꦸ → ꦏꦿꦸꦸꦸ *ndra*

ꦏꦿ + ꦏꦿꦸꦱ + ꦏꦸꦱ → ꦏꦿꦸꦸꦱꦸ *nru*

ꦏꦿ + ꦏꦿꦸꦱ + ꦏꦿꦸꦱ + ꦏꦸꦱ → ꦏꦿꦸꦸꦱꦸꦱꦸ *ndru*

ꦏꦫ + ꦏꦫ + ꦏꦫ + ꦏꦫ → ꦏꦏꦫꦫ *kkru*

ꦏꦫ + ꦏꦫ + ꦏꦫ → ꦏꦏꦫꦫ *npra*

ꦏꦫ + ꦏꦫ + ꦏꦫ + ꦏꦫ → ꦏꦏꦫꦫꦫ *ntrya*

- Glyphs that sit below below-base glyphs may be reduced in height, and below-base vowels may be attached at a higher-than-usual position, to reduce overall line height.

ꦏꦫ + ꦏꦫ → ꦏꦫꦫꦫ *-tya*

ꦏꦫ + ꦏꦫ + ꦏꦫ → ꦏꦫꦫꦫꦫ *-tyu*

ꦏꦫ + ꦏꦫ → ꦏꦫꦫꦫ *-kla*

- Above-base marks are commonly not stacked, but displayed side-by-side or even inside one another, which can be accomplished by creating combination glyphs. When determining the set of supported combinations, it is reasonable to assume that U+A980 ◌ꦲ JAVANESE SIGN PANYANGGA and U+A9B3 ◌ꦲ JAVANESE SIGN CECAK TELU do not co-occur, and that at most one of the vowels U+A9B6 ◌ꦲ JAVANESE VOWEL SIGN WULU, U+A9B7 ◌ꦲ JAVANESE VOWEL SIGN WULU MELIK, or U+A9BC ◌ꦲ JAVANESE VOWEL SIGN PEPET occurs. U+A981 ◌ꦲ JAVANESE SIGN CECAK and U+A982 ◌ꦲ JAVANESE SIGN LAYAR only co-occur when *layar* is used as *repha* in Kawi or Sanskrit.

ꦏꦫ + ꦏꦫ → ꦏꦫꦫꦫ *-ěr*

ꦏꦫ + ꦏꦫ → ꦏꦫꦫꦫ *-ěng*

ꦏꦫ + ꦏꦫ → ꦏꦫꦫꦫ *(nukta)-i*

ꦏꦫ + ꦏꦫ + ꦏꦫ + ꦏꦫ + ꦏꦫ → ꦏꦏꦫꦫꦫꦫꦫ *nfri*

- Above-base marks are positioned above base glyphs or right-side conjunct forms, not above right-side vowels. In OpenType, this means that right-side vowels must be treated as marks, so that they can be ignored when positioning the above-base marks. Because of a compatibility feature in the Universal Shaping Engine, this causes their width to be set to 0, so that it must be added back later using the “dist” feature.

ꦭꦺꦴꦩ꧀ + ◌ꦠ + ◌ꦺꦴꦩ꧀ → ꦭꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ *om*

ꦒꦺꦴꦩ꧀ + ꦒꦺꦴꦩ꧀ + ◌ꦠ + ◌ꦺꦴꦩ꧀ → ꦒꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ *nor*

ꦒꦺꦴꦩ꧀ + ◌ꦠ + ꦒꦺꦴꦩ꧀ + ◌ꦠ + ◌ꦺꦴꦩ꧀ → ꦒꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ *for*

- Fonts may enable the selection of stylistic glyph variants via features such as stylistic sets.

ꦒꦺꦴꦩ꧀ → ꦒꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ *ba*

ꦒꦺꦴꦩ꧀ → ꦒꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ *nya*

- Orthographic syllables often need to be spaced apart to avoid collisions between below-base glyphs.

ꦭꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ → ꦭꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ *om swastyastu*

Ligatures versus glyph positioning. The Javanese script has numerous glyphs that may, depending on the design of a typeface, connect to each other. For example, the conjunct form ꦧꦺꦴꦩ꧀ *wa* may connect to the right-most stems of many base consonants, as in ꦏꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ *kwa*. If followed by a vowel ꦸꦩ꧀ *u*, it may also connect to that: ꦏꦺꦴꦩ꧀ꦠꦺꦴꦩ꧀ꦸꦩ꧀ *kwu*. For best rendering results, all such combinations of connecting glyphs would be implemented as ligatures. However, these combinations can easily number in the hundreds, possibly over a thousand, and substantially increase the size of a font. The alternative is to not include such ligatures, or only the most commonly used ones, and instead rely on positioning the connecting glyphs appropriately. This may however result in slight offsets in connecting lines, as renderers usually first rasterize the glyphs separately, then position them relative to each other, and both steps involve rounding of coordinates. Font developers should consider the trade-off carefully.

7 Keyboards

Key issues in the development of keyboards include which characters to support, how to arrange them on a physical keyboard or on screen, how to let the user interact with the keyboard, whether and how to support predictive input, and how to ensure the correct encoding order of orthographic syllable components. This section discusses some of these issues, comparing the Indonesian standard SNI 9048:2021, which covers physical and

virtual (on-screen) keyboards for Javanese as well as for Sundanese and Balinese, and the (on-screen only) keyboard in the Aksara Jawa app of Lindenberg Software.

Supported characters. When selecting characters to support, a common practice is to select only the characters needed for a given language, not all characters in a script. The modern Javanese language, for example, requires only 20 of the 36 consonants encoded in Unicode. On the other hand, a language-based approach requires identifying the character set used for each language (there does not seem to be much information available on Madurese or Sasak) and then creating separate keyboards for each. As the currently encoded Javanese character set is not that large, both SNI 9048 and the app’s keyboard provide the complete character set.

One major difference between SNI 9048 and the app’s keyboard is how they support conjunct forms. SNI 9048 does not specify dedicated keys for them, so users have to understand that they have to type first PANGKON, then the letter whose conjunct form they need. Experience with the Aksara Bali app of Lindenberg Software, which has the same limitation, has shown that a fair number of users do not understand this, and complaints about the “lack of support” for conjunct forms was the primary cause for customer care issues as well as negative reviews for this app. The Aksara Jawa app therefore provides separate layers with conjunct forms, one of which is shown below. This solution has proven much easier to understand.



Keyboard layout. SNI 9048 specifies both physical and virtual keyboard layouts. Both are similar and follow the English/Indonesian layout for physical keyboards, with two layers of five rows. For the physical keyboards, this may take advantage of muscle memory. The Aksara Jawa app, on the other hand, takes advantage of on-screen keyboards’s ability to show the keys of all layers, and uses three layers for base characters and two for conjunct forms, each with four rows to better fit on phone screens.

Encoding order of orthographic syllable components. SNI 9048 does not discuss the encoding order of orthographic syllable components. This likely means that correct ordering is left up to users, who will have to understand the encoding order and edit text until it is free of dotted circles. The keyboard of the Aksara Jawa app relieves them of this by automatically reordering characters within an orthographic syllable. This is possible because the keyboard API on iOS lets keyboards read and edit the text surrounding the current insertion point. The input method APIs on Android, macOS, and Windows provide similar capabilities. Smart keyboards should take advantage of them to help users avoid incorrect text and dotted circles.

8 Line breaking

Lines of Javanese text can be broken at any orthographic syllable boundary. The Unicode Line Breaking Algorithm and implementations based on it, such as the Internationalization Classes for Unicode (ICU) library, do not yet provide this style of line breaking. Instead, two errors commonly occur: Either lines are broken only at punctuation, resulting in text overflowing the space available to it, or lines are broken at Unicode grapheme cluster boundaries, resulting in broken conjunct forms, as PANGKON is treated as the end of a grapheme cluster.

A [proposal to correct the Unicode Line Breaking Algorithm](#) has been submitted to the Unicode Technical Committee, which has [requested](#) that this proposal be [implemented in ICU](#). Owners of other implementations should update them based on the proposal.

9 Acknowledgments

I would like to thank Aditya Bayu Perdana, who explained many details of Javanese typography to me and who designed the font used in this document (and in the Aksara Jawa app). I also thank Bayu (again), Evelyn Teo, Fadhl Haqq, Liang Hai, Marc Durdin, Martin Hosken, Muthu Nedumaran, and Simon Cozens for feedback on drafts of this document.

