Noto Sans Tagalog v3.0

•••

Documentation & Information

FREDRICK R. BRENNAN

2 August 2020

Differences between Noto Sans Tagalog

Noto Sans Tagalog v3.0 improves upon v2.0 in the following ways:

- It is a one-axis (weight) variable font with an `avar` table
- It includes glyph for “ra”, a missing letter, at U+170D
  - Also archaic “ra” at U+171F
- It includes a *pamudpod*
- It includes alternate hollow *kudlit* enabled in various ways (`salt`; `cv02`; VS1)

Overview

Tagalog?

Noto Sans Tagalog is a font for a script known to *The Unicode Standard* as “Tagalog”. However, this name is not what the script is known by to Filipinos; Filipinos know it as *baybayin* or *alibata*. *Baybayin* is the name preferred by Philippine scholars, deriving from the Tagalog verb root *baybay*, meaning “to spell”.

The name “Tagalog” was kept as Noto Sans Tagalog is part of the much larger Noto family, and as all the others are named after their respective Unicode blocks, so too should Noto Sans Tagalog. However, the alternate name of “Noto Sans Baybayin” should be understood just as well by your OS to refer to the font, as it’s in the name table.

Creation notes

Process in a nutshell

This was primarily an engineering project and learning experience for me, Fred Brennan, the author of this font. This was my first Google Web Font.

It is a requirement of Google that all Google Fonts be built from UFO with `fontmake`. This is unfortunate as it leads many to use Glyphs to make Google Fonts, which is not what Google wants to promote (I'd hope). I worked out a deal with Google where I'd use FontForge to make *precursors*, and then compile the final font with `fontmake`. So that's what I did.

I drew the glyphs in FontForge saved as an SFD "precursor file", and exported to UFO "source" file. UFO "source", combined by `fontmake` with "design space file" generated with help of AFDKO library and RoboFont example. In August 2019 I made a tutorial about my process.

Different strokes for different blokes

This font makes use of the excellent Expand Stroke work contributed to FontForge by Skef Iterum. Mr. Iterum rewrote the entire Expand Stroke system FontForge had up to 2019, well known to be a buggy mess before his blessed intervention, and his work first appeared in the Pi Day 2020 release (`20200314`).

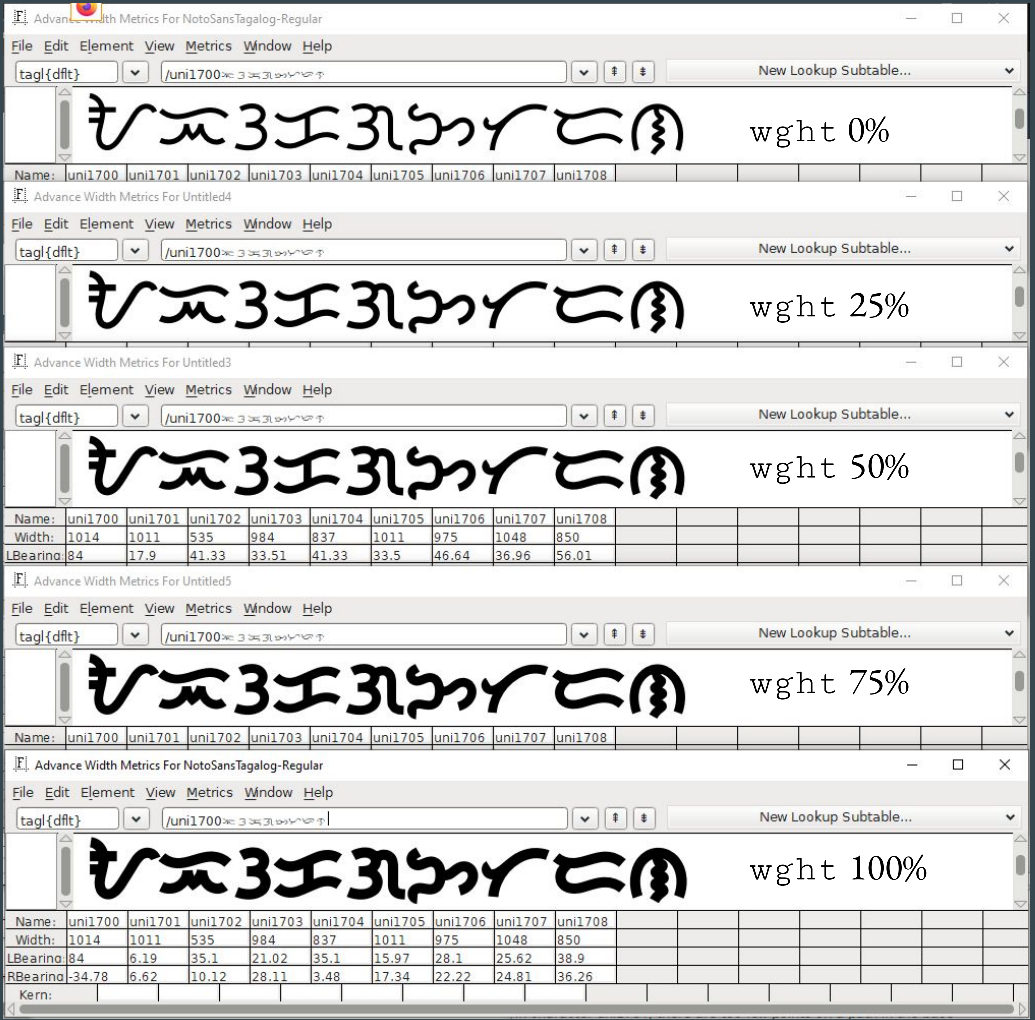
Much thought went into automating making different width strokes interpolation compatible; the final font uses some of this work. However, every glyph in every master was tweaked so much by hand that this work is not distributed as part of the build tools at this time. The question of speeding up the making of glyphs interpolation compatible is a question I continue to consider; it is possible I will release a Python extension for FontForge some day based on the gears that working on Noto Sans Tagalog put in motion in my mind.

OpenType Variations

Weight axis

Noto Sans Tagalog v3.0 is the first variable font for *baybayin*. It is a one-axis font with an `avar` table which improves the design of several characters in the bold weight.

For those systems which do not yet support variable fonts, the standard build script outputs four fonts: a font equal to the current "Noto Sans Tagalog" (`wght` 400); a font equal to what would be "Noto Sans Tagalog Bold" (`wght` 700); and then then two further fonts with `wght` 500 and 600.


Advance Width Metrics For NotoSansTagalog-Regular
File Edit Element View Metrics Window Help
tagl{dflt}
/uni1700
New Lookup Subtable...
wght 0%
Advance Width Metrics For Untitled4
wght 25%
Advance Width Metrics For Untitled3
wght 50%
Advance Width Metrics For Untitled5
wght 75%
Advance Width Metrics For NotoSansTagalog-Regular
wght 100%

Name:	uni1700	uni1701	uni1702	uni1703	uni1704	uni1705	uni1706	uni1707	uni1708

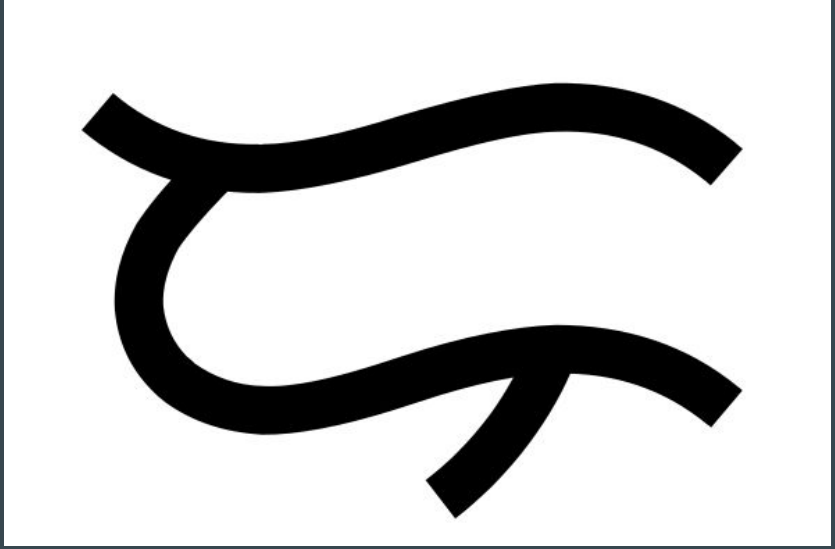
Name:	uni1700	uni1701	uni1702	uni1703	uni1704	uni1705	uni1706	uni1707	uni1708
Width:	1014	1011	535	984	837	1011	975	1048	850
LBearing:	84	17.9	41.33	33.51	41.33	33.5	46.64	36.96	56.01

Name:	uni1700	uni1701	uni1702	uni1703	uni1704	uni1705	uni1706	uni1707	uni1708

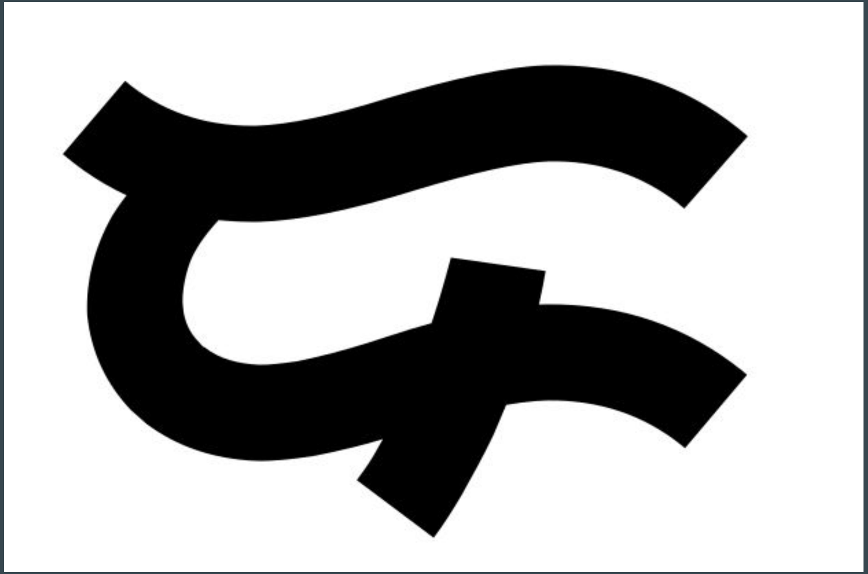
Name:	uni1700	uni1701	uni1702	uni1703	uni1704	uni1705	uni1706	uni1707	uni1708
Width:	1014	1011	535	984	837	1011	975	1048	850
LBearing:	84	6.19	35.1	21.02	35.1	15.97	28.1	25.62	38.9
RBearing:	-34.78	6.62	10.12	28.11	3.48	17.34	22.22	24.81	36.26
Kern:									

avar — U+170D stroke position and crossing behavior

At and above wght 750, U+170D's differentiating stroke begins to cross over so as to be more visible at small point sizes.



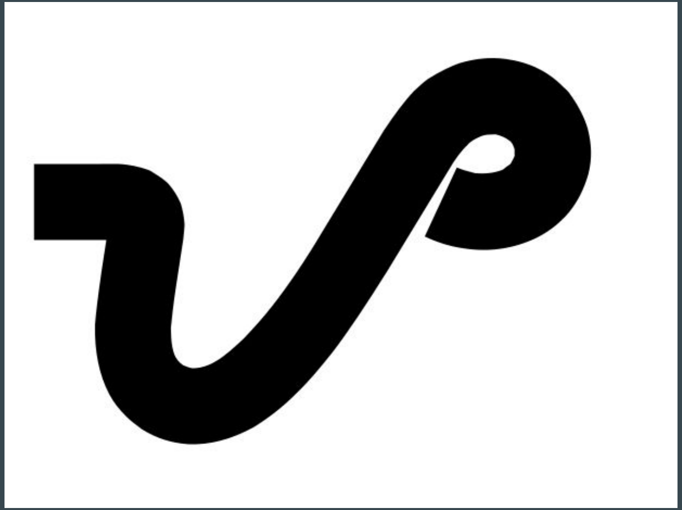
wght 0%

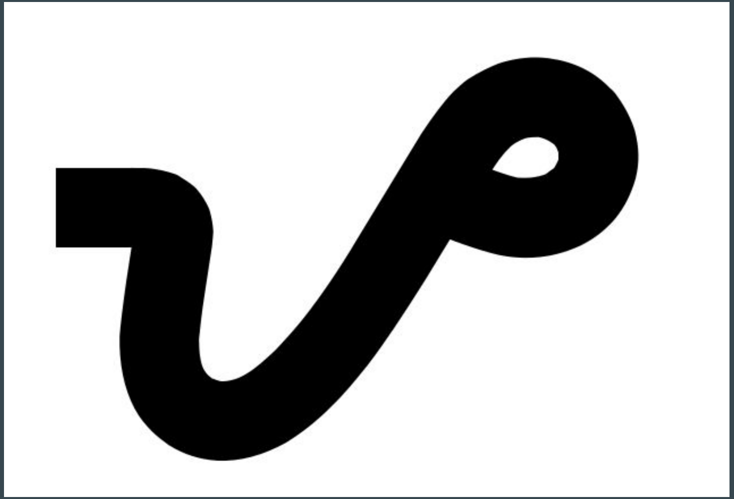


wght 100%

`avar` — U+170C gap closing

At and above `wght` 75%, the gap (*simulated at left*) closes in “ya”. The tiny gap looks quite ugly at small point sizes, so is better closed.





Unicode considerations

COVID-19 delays

Oh, the dreaded topic of our era. Yes, indeed, even font documentation, this holiest topic unconcerned with the fleshy problems of men, cannot escape it.

I was commissioned to work on this font after a successful Unicode proposal: L2/19-258R, entitled *The baybayin "ra"*, and a complaint I made on the Noto bug tracker based on it. (Okay, yes, it certainly helped that Dave Crossland can put up with me in small doses.)

My proposal was accepted, and the missing letter "ra" will be encoded in Unicode 14.0. It, along with its archaic brother, is in the pipeline to be encoded.

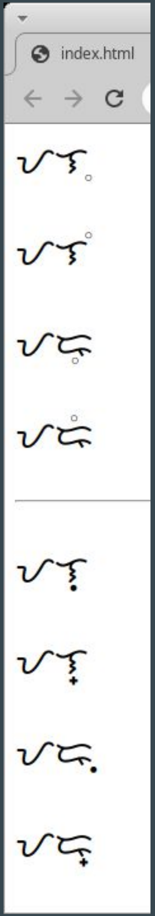
However, until it finally lands, there's a bug you should be aware of, detailed on next slide.

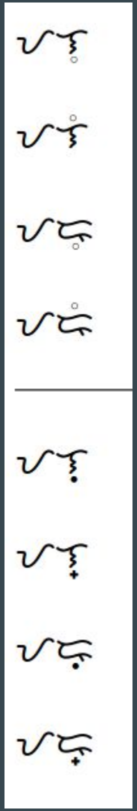
Chrome text layout bugs and you

Chrome (*at left*), for reasons known only to its developers, (and perhaps not even to them! 😉) seems to render all scripts in different “runs”.

So, if it encounters, let’s say, Latin letter Q followed immediately by Hebrew letter Qoph (ק), even if the font contains glyphs for both, it will feed HarfBuzz the text in two runs.

Chrome seems to consider unencoded characters in a script of their own. As such, they are not rendered as part of the `tagl` script’s run. As such, the broken *kudlit* placing. Firefox (*at right*), for its part, works fine.





Chrome text layout bugs and you (part 2)

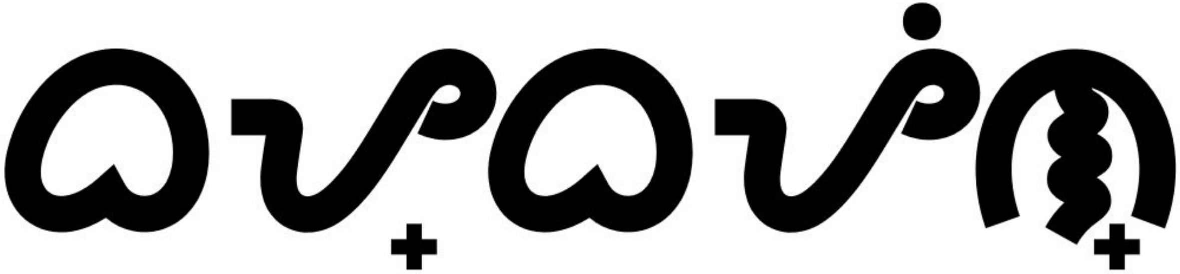
Is this a bug? Perhaps perspective is everything. However, as soon as Unicode updates, whenever that might be, (it is my understanding that without COVID-19, it would have already happened, or else happened by March 2021; but with COVID-19, we're looking at September 2021 at earliest,) Noto Sans Tagalog will not need to be reissued, nor a new version issued, to fix this, everything will "just work".

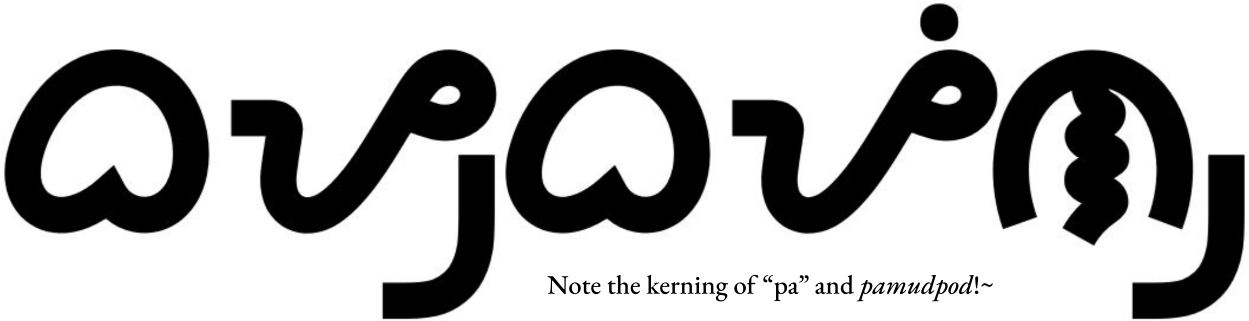
Because Chrome does it in two runs, there is nothing I can do. No `ccmp` will save the day, no `liga` nor `ss01` nor `calt`. The first glyph in the run is the mark, the OpenType Layout code I wrote doesn't even get to see the preceding "ra" as it's in a different run. 😔

Pamudpod

Pamudpod: alternate *virama*

The *pamudpod* originates in the Hanuno'o script. It is a native version of the Spanish cross-shaped *virama*, and just like it, it silences a vowel. At top, *virama*. Bottom, *pamudpod*.





Note the kerning of "pa" and *pamudpod*!~

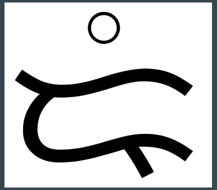
Hollow *kudlit*

“E” is “i” and “o” is “u”

Traditionally, the *baybayin* script does not differentiate “e” from “i” nor “o” from “u”. This is a native feature of the Old Tagalog language, which only had three vowels: “a”, “i” and “u”. Gradually, “i” became expressible in IPA as [ɛ ~ i] and “u” as [o ~ u]. The Spanish, upon arrival, affected the phonology and vocabulary of Tagalog (though spared 99% of its grammar) with their imported words, and “e” split from “i” and “o” from “u”, becoming truly distinct phonemes; the distinctions now also impact native words: *upo* is not *opo*.

Norman de los Santos, author of many *baybayin* fonts, recommends the use of a hollow *kudlit* for “e” and “o”. Such *kudlit* is not in Unicode; someday I may propose it. I considered doing so to accompany this font, but I believe more evidence of its acceptance by the community is needed. However, I offer in Noto Sans Tagalog v3.0 three ways to get hollow *kudlit*, even though individual characters for them do not exist at present.

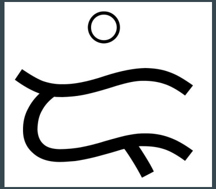
Hollow *kudlit*—Stylistic Alternates (`salt`)



A challenge Unicode proposers will no doubt face as regards hollow *kudlit* is that many *baybayin* fonts make all *kudlit* hollow, with no intended sound change. However, when a font includes both, it is meant that the hollow form of the *kudlit* above correspond to “e” and hollow *kudlit* below correspond to “o”.

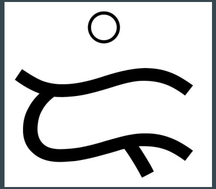
As some fonts already make all *kudlit* hollow, I offer the option in Noto Sans Tagalog v3.0 via OpenType feature Stylistic Alternates (`salt`).

Hollow *kudlit*—Character Variant (`cv02`)



Hollow *kudlit* can also easily be considered a mere character variant. So, I offer it as one as well, enabled via OpenType Character Variant 2, `cv02`.

Hollow *kudlit*—Variation Selector 1



Unicode offers something called a “variation selector”. VS1—U+FE00, is marked by the Unicode Consortium as having a CJK purpose. So, as the Tagalog block is never going to become ideographic, we can use it to encode a hollow *kudlit* in plaintext.

Thus, the glyph at top right can be encoded as: U+170D; U+1712; U+FE00.

OpenType roundup

Other new OpenType features Noto Sans Tagalog v3 understands

- `hist`: Makes “ra” (U+170D) into archaic “ra”. (U+171F)
- `salt`: Besides mentioned uses, also turns *virama* 45 degrees, making them “X”-shaped.
- `kern`: Although we have few kerns, some characters, like *pamudpod*, benefit.
- `mkmk`: For the *double kudlit*.†
- `calt`
  - For the *pahantig*. ZWNJ + [U+1712 U+1713] = *pahantig*.†
  - For the *padalawa*. ZWNJ + [U+1712 U+1713] + [U+1712 U+1713] = *padalawa*.†
    - VS1 may be provided as well, to either first or second.
- `cv01`: Turns *virama* 45 degrees, but not *kudlit* hollow.
- `cv02`: Turns *kudlit* hollow, but not *virama* 45 degrees.

† See de los Santos (2014), p. 27.