

Kurinto

Font Folio



Kurinto Font Folio – User’s Guide

If you use *Microsoft Word* to publish PDF documents (like this one) that include text in non-European languages, you are likely to encounter many pitfalls:

- costly fonts,
- publishing restrictions,
- irregular line heights,
- poorly implemented styles and kerning,
- enormous PDF file sizes,
- spurious layout changes, and
- font format incompatibility.



If your document uses text with non-European characters, you are also likely to encounter:

- missing characters (which often look like □□□, □□□, ◆◆◆, □□□, or □□□),
- random font changes, and
- discordant font styles.

These pitfalls prevent many authors from producing a presentable (or even usable) PDF file. Kurinto fonts are designed to address all these issues.

The [Kurinto Font Folio](#) is a large collection of free fonts that include most of the characters in every human language. The fonts are designed to work together to create typographically cohesive digital publications. Academic publishing is an ideal use of these fonts.

Kurinto helps authors publish their work without studying font technology or using advanced typographic techniques. While the fonts can be useful in many situations, they are specifically intended for authors who use word processors (such as *Microsoft Word*) to publish multi-language text to digital documents (such as PDF).



Kurinto is and always will be free. The fonts are available to anyone at no charge and licensed under the *SIL Open Font License Version 1.1*. Kurinto fonts may be used, studied, copied, merged, modified, and redistributed. You may download and embed these fonts in digital documents, use them in commercial projects (including mobile apps), and bundle them (with or without modification) for re-distribution under the terms of the *SIL Open Font License Version 1.1*.

This document provides guidance for users of Kurinto fonts. It also has some rationale for design choices and limited information for font developers. To access Kurinto fonts, visit [Kurinto.com](#). For additional developer resources, visit [Github.com/ClintGoss/Kurinto](#).

I hope you find Kurinto useful!

— Clint Goss [clint@goss.com], as of July 25, 2020

Copyright ©2017–2020, Clinton F. Goss (clint@goss.com, www.goss.com) with Reserved Font Name Kurinto.

This Font Software is licensed under the *SIL Open Font License Version 1.1*, which is copied [below](#) and is also available with an FAQ at: <http://scripts.sil.org/OFL>.

END OF TERMS AND CONDITIONS

Excel®, Microsoft®, OpenType®, Outlook®, and Windows®, are registered trademarks of Microsoft Corporation in the United States and/or other countries. Cambria™, Calibri™, and Georgia™ are trademarks of the Microsoft group of companies. TrueType™ is a trademark of Apple Inc. PostScript® and Adobe PDF® are registered trademarks of Adobe Systems Incorporated. Times®, Palatino®, and Helvetica® are registered trademarks of Linotype-Hell AG and/or its subsidiaries. Arial™, Courier New™, and Times New Roman™ are trademarks of The Monotype Corporation. Monotype Garamond® is a trademark of Monotype Typography, Ltd which may be registered in certain jurisdictions. Unicode® is a registered trademark of Unicode, Inc. in the United States and other countries. Kurinto™ is a trademark of Clinton F. Goss in the United States and other countries. All other product names, trademarks, and registered trademarks are the property of their respective owners.

Note that “open source” is not a trademark (see the discussion at <https://opensource.org/pressreleases/certified-open-source.php>).

Legal Disclaimer

I do not have formal training in intellectual property law or any other branch of law. Nothing in this document should be considered authoritative legal advice.

Conventions and Typesetting

Font names such as **KURINTO TEXT WIDE TB** are typeset in **KURINTO BOOK BOLD SMCAPS**.

The default body text of this document is 12pt **KURINTO TEXT VERSION 2.195**. The typeface does change in some sections to demonstrate that font family, but all the fonts are the same version.

Headings such as *Conventions and Typesetting* are set in **KURINTO SERI ITALIC**. Software applications such as *Microsoft Word* are also typeset in **KURINTO SERI ITALIC**.

File names such as `ReadMe.txt` are typeset in **KURINTO MONO NARROW**.

Character indexes (code points) in *The Unicode Standard* are identified by the prefix “U+” followed by four to six hexadecimal (base 16) digits, all typeset in 11pt **KURINTO MONO**.

Names of Unicode characters are typeset in **KURINTO BOOK SMALL CAPITALS**. For example: the code point for the character Θ GREEK CAPITAL LETTER THETA is U+0398. Ranges of Unicode code points are shown as, for example, U+13000–U+1342F.

Terms shown in **red bold** are my own inventions for this project.

Names and abbreviations of licenses such as the *Open Font License* and the *OFL* are set in the Italic style of the current font.

The font used in the stylized keystrokes, such as **K**, use **KURINTO MONO**. This font is also used for the license text of the *Open Font License* shown on pages 13 and 14.

Commands in popup context menus are shown as **Popup Command**.

Overview

Most font packages do not have (or need) user manuals. However, Kurinto is larger and more complex than most font packages. It is a folio of many typefaces, each with several variants and styles.

If you wish to get started quickly, feel free to use the *Quick Start Guide* and start using the fonts. There is no need to consult the rest of this User's Guide, unless you want more information on how the fonts are structured and how they address the various issues and pitfalls of authoring documents for digital publication.

This User Guide also serves a secondary purpose: it offers an example (and a test) of the of the scenario that Kurinto addresses: publishing Word documents to PDF. To serve this secondary goal, some of the included non-Latin examples are extended further than might otherwise be appropriate (for example, the IPA (International Phonetic Alphabet) [transcription of “Kurinto” given below](#)). Also, the .docx file for this User Guide is included in release packages.

What is Kurinto?

Kurinto fonts are designed for authors who write primarily in European languages (i.e. using Latin letters), but who also include text that uses other, non-European, writing systems. Kurinto fonts may be useful if you need to, for example, reliably cut/paste text from Wikipedia and have it “work” — be readable, searchable, not disturb the document's format, not unduly increase the size of the PDF file, and match the style of the text that surrounds it.



A common publishing path for academic authors is to use *Microsoft Word* to create PDF files.¹ This path is fraught with pitfalls. Recent advances in font technology have laid the groundwork for representing all the world's writing systems, but I found it daunting to assemble a set of fonts for the many languages I needed. After wrestling for years to assemble a coordinated set of full-featured fonts, I felt it was a worthwhile task to expand the project and release it for general use ... Kurinto is the result of that effort.

Each of the five primary typefaces – **KURINTO TEXT**, **BOOK**, **SANS**, **SERI**, and **MONO** – contains most of the 137,994 characters in the Unicode Version 12.1 standard and covers all 150 modern and historic writing systems.² These typefaces also contain 30,097 characters in 77 auxiliary writing systems and character collections that are not currently part of the Unicode standard.

The eight secondary typefaces – **KURINTO ARTE**, **CODE**, **CURV**, **NEWS**, **OLDE**, **PLOT**, **ROMA**, and **TYPE** – are more specialized. They are designed for headings or text that stands out from the main body text.

¹ Microsoft estimates that 1.2 *billion* people use Word.

² *The Unicode Standard* uses the term “script” for a collection of character that make up a writing system. Rather than “script”, this document uses the more descriptive term “writing system”.

These fonts have limited coverage of the Unicode standard, but still have all the characters needed for many writing systems.

The “metric compatible” typefaces serve a special function: They let you replace popular fonts with open-source fonts without changing the layout of your document. If you use popular fonts such as Arial, Cambria, Calibri, Courier, Garamond, Georgia, or Times New Roman, you can directly replace those fonts with the corresponding metric compatible font – **KURINTO ARIA**, **BRIA**, **CALI**, **CMOD**, **CNEW**, **GARA**, **GRGA**, **TMOD**, and **TROM**. Each character in a metric compatible Kurinto font takes the same space (horizontal and vertical) as the original font.

To accommodate every human language, the typefaces are segmented into several font “variants”. Most writing systems in current use are included in the primary (or “Main”) font variant. The font for the bold version of **KURINTO TEXT** that includes the Main variant is called **KURINTO TEXT BOLD**. Historical languages (e.g. Phoenician) and constructed languages (e.g. Klingon) are included in the “Aux” variant – **KURINTO TEXT AUX BOLD**. Asian writing systems with very large character sets or composition rules are included in specific font variants – for example Japanese (**KURINTO TEXT JP BOLD**), Traditional Chinese (**KURINTO TEXT TC BOLD**), and Tibetan (**KURINTO TEXT TB BOLD**). There are also variants for certain large, specialized character sets, including **MUSIC** and **UFI** (a combination of several Unicode Font Initiatives including Medieval, Cyrillic, and Runic characters).

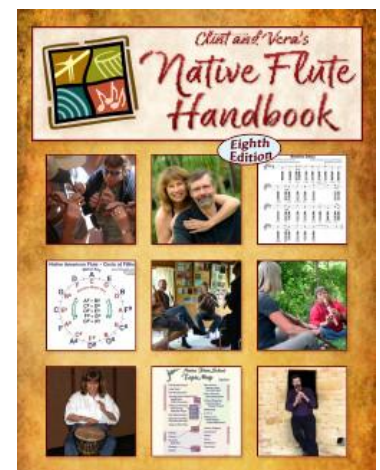
History

As part of my avocation playing the Native American flute, I published a book on the subject in 2014. It was authored in *Microsoft Word* and the beautifully printed copies were distributed privately.

By the third edition (320 pages), demand for the book exceeded the abilities of my laser printer (and my patience). I changed distribution from paper to a free PDF download ...

... and promptly descended into digital-publishing hell.

Despite my training and experience in technology³, I wrestled the issues, traps, annoyances, and pitfalls related to using fonts. The [Native Flute Handbook](#) is primarily written in English, but has Cherokee-language (ᎠᎵᎠᎵᎠᎵ)⁴ benedictions, Canadian Aboriginal Syllabics (ᓄᓄᓄᓄ), Hebrew (קורִינְטוֹ) lyrics, Rumi poetry in Arabic (كورينتو), and writings of Confucius (【第八章】 【一節】子曰、興於詩。【二節】立於禮。【三節】成於樂。)⁵ in an archaic form of Chinese.



³ A Ph.D. in Computer Science and 30 years of consulting experience.

⁴ Character samples of various character sets shown in this paragraph are transliterations of the word “Kurinto” into that language.

⁵ Sourced from *The Analects*, Volume 4, Book 8, Chapter 8. The quotation is based on the annotated and expanded text of William Edward Soothill (1861–1935), *The Analects of Confucius*, published by the author, Yokohama, 1910. See also

I was able to develop an initial set of fonts that served my needs, which were released as Kurinto v1.07 in 2016. I then began to realize that many authors – especially those producing academic publications – shared the same issues and could benefit from this project. In the three years between the seventh and eighth edition of the Native Flute Handbook, I took a **major** side trip and developed version 2 of Kurinto, which this User’s Guide describes.

Use Case

It's 1988, and you are writing your dissertation on *Invertebrate Courting Rituals*. Most of the thesis is in English, but a body of work has been done in Czech and Russian, and some sections need to be properly excerpted and cited.

You have obtained a document authoring system which, after much effort, handles the various accents over letters when writing Czech. However, copying text or citations written in Russian looks like this:⁶

²áÕ ÛîÔØ àÐÕÖÐîââî áÒÐÑÐÔÝëÛØ Ø àÐÕÝëÛØ Ò áÒÐÕÛ ÔÐáâÐØÝáâÒÕ Ø
ßàÐÒÐâ. ¾ÝØ ÝÐÕÕÛÕÝë àÐ × äÛÐÛ Ø áÐÕÕáâî Ø ÔÐÛÕÝë ßáââßÐâì Ò
ÐâÝÐèÕÝØ ÒâãÓ ÔâãÓÐ Ò ÔâãÕ ÑâÐâââÒÐ.

You seek expert advice, and are told about mojibake, code pages, metadata, screen fonts, printer fonts, and the need to properly select from among the thousands of possible code pages. You sigh, take a photo of the text with your SLR camera, paste the photo into your document with rubber cement (looks a bit shabby), and get your degree.

Fast forward 30 years. You’re writing your magnum opus: *Everything and Anything About Invertebrate Courting Rituals*. More work has been done in Asia, and there are key references in several more languages. Surely, the language / character issues must have been addressed in the last three decades ...

You cut-and-paste text from various sources into your document, and your document shows them like this:

Mating behavior in *cladocera* has been described in great detail by several authors over the last two centuries (see [1890] and [1969]). The coupling process, which can last as long as 8–10 min, involves the male grasping the female’s legs within her carapace with his first thoracic legs and the long setae of his first antennae.

You seek expert advice, and are told about Unicode, glyphs, code-point coverage, and OpenType advanced typography features. You sigh, take a photo of the text with your iPhone, and drag-and-

James Legge (1815–1897), *The Chinese Classics, Volume 1*, Confucian Analects (論語), London Missionary Society, Hongkong, 1861.

⁶ See [Mojibake](#) if you’re looking for a technical explanation of this text.

drop the photo into your book. Your document grows by a megabyte for each image, the text is fuzzy, and it cannot be searched. And once again ... it looks a bit shabby.

Why does an expert in invertebrate courting rituals need to understand the arcane details of font technology? (Any more than a type designer should need to learn about invertebrate courting rituals?)

Caveats

Here are some situations where Kurinto might **not** be ideal:

- If your primary writing system is not the pan-European “Latin” writing system – i.e. you primarily use Chinese, Devanagari, Arabic, Bengali, Cyrillic, Kana, etc. – then Kurinto may not be the best choice. There are some aspects of the way I implement multiple writing systems in single, large fonts that makes them less ideal than a font dedicated to those specific writing systems.
- Kurinto fonts are not specifically design for use on the Web or mobile devices. The **CORE** variant of the fonts can serve as Web fonts, but there are purpose-built web fonts that may serve you better (see [fonts.Google.com](https://fonts.google.com)).
- Kurinto fonts are large. Except for the **CORE** variant of fonts, most of the font files range from 5 to 30 megabytes – 10 to 60 times larger than most font files. This can stress older systems. I have not developed definitive minimum system requirements, but you probably want to be running a 64-bit operating system on a multi-core processor with at least 16 GB of physical memory. I expect these issues will become mute over the next decade.

What is Unicode?

Unicode aims to **Unify** the **encode**ings of all the world's character sets into a single Standard.⁷

Prior to Unicode, the “special” characters – those outside of the 95 most-used characters in Western culture – were handled by schemes that were arcane and unreliable. Since most of the world’s hundreds of thousands of characters are outside the 95 Basic Latin characters⁸, there are a lot of “special” characters. And there was a **lot** of arcane and unreliable work going on to represent all those “special” characters.

The Unicode Standard replaced all those arcane schemes with a carefully designed and reliable system for representing all the world’s characters. Although it is not without controversy, and despite being designed by committee, Unicode had succeeded: Nowadays, those old arcane schemes are rarely used. Most everyone uses Unicode.



⁷ Excerpted from <https://perldoc.perl.org/perlunicode.html>.

⁸ The 95 printable characters of the ASCII set. See https://en.wikipedia.org/wiki/ASCII#Printable_characters.

For an introduction to Unicode, see <https://en.wikipedia.org/wiki/Unicode>. For a full discussion of all aspects of Unicode, see <http://www.unicode.org>.

About the Name

“Kurinto” (koo-IN-toh; with various IPA transcriptions: /kuːɹɪnrou/; [kʰuːɹɪnrou]⁹; and kʊəntəʊ¹⁰) is how I hear my name – “Clint” – pronounced when we visit Japan. Native speakers in Japanese pronounce the word with equal emphasis on each syllable – like “Koor-In-Toh” – but my Western-trained brain *hears* it as “koo-IN-toh” so that is my preferred pronunciation.

The term “font folio” is newly invented (as far as I know) for this project. A font folio is a collection of typefaces (font families) that are designed to work together in a coordinated way. The fonts in the folio have coordinated naming, styles, metrics, embedding permissions, Panose settings, and typographic standards. The font families in the folio may fall into multiple classifications (e.g. serif, sans-serif, calligraphic, monospaced, etc.). In some circles, the term “font superfamily” is used for this concept.¹¹

About the Kurinto Font Folio License

Kurinto is free software. You may use, study, copy, merge, embed, modify, redistribute, and sell modified and unmodified copies of Kurinto under the terms of the *SIL Open Font License Version 1.1* (“the OFL”).



The *OFL* is maintained by SIL International. It attempts to be a compromise between the values of the free software and typeface design communities. It is used for almost all open source font projects, including those by Adobe, Google, and Mozilla.¹² The *OFL* is copied on pages 13–14 and is also available at <http://scripts.sil.org/OFL>.

Additional information is available in these text files that are part of every distribution package:

- **Contributors.txt** – Rosters of source fonts used in Kurinto, font designers of those source fonts, and general acknowledgements.
- **Copyrights.txt** – Copyright legends and licenses that apply to Kurinto and the source fonts that are incorporated in Kurinto.
- **FAQ-Kurinto.txt** – A vestigial Frequently Asked Questions file that refers to the [FAQ section in this User Guide](#).
- **FAQ-OFL.txt** – Frequently Asked Questions about the *SIL Open Font License Version 1.1*. The latest version of this may be obtained at <https://scripts.sil.org/OFL-FAQ> web.
- **Fontlog.txt** – A log of revisions made to Kurinto. If you use Kurinto fonts in a derivative work (i.e. create your own fonts based on Kurinto), please add to this file and distribute it along with your derived fonts.

⁹ See https://en.wiktionary.org/wiki/Wiktionary:IPA_pronunciation_key.

¹⁰ Provided by www.ToPhonetics.com on December 17, 2018.

¹¹ See https://en.wikipedia.org/wiki/Font_superfamily.

¹² Description from <https://choosealicense.com/licenses/ofl-1.1/>, retrieved March 3, 2020, CC-BY-3.0-UNP.

- **License.txt** – Copyright legend for Kurinto and the text of the *SIL Open Font License Version 1.1* (which is also copied [below](#)).
- **Map.txt** – A roster of the writing systems that make up each Kurinto font and which specific source files contributed each of those writing systems. The writing systems are identified by script codes that are described in the [Script Codes](#) section of this User Guide and listed in the `Scripts.txt` file.
- **OFL.txt** – The text of the *SIL Open Font License Version 1.1* (which is also copied [below](#)).
- **Panose.txt** – A list of Kurinto fonts in the release package together with their Panose values.
- **Patents.txt** – Patent legends that apply to the source fonts that are incorporated in Kurinto.
- **ReadMe.txt** – Overview information about the Kurinto Font Folio and the definitive roster of fonts in the particular release package.
- **Scripts.txt** – A list of the script codes used to identify writings systems. These script codes are an expansion of the Unicode scripts. They are used in the `Map.txt` file to identify which source fonts contributed to each writing system in each font.
- **Trademarks.txt** – Trademark legends that apply to Kurinto and to the source fonts that are incorporated in Kurinto.

Visit <https://www.Kurinto.com/> or contact Clint Goss (clint@goss.com) for more information. In particular, extensive licensing information on the various source fonts contained in Kurinto at <https://Kurinto.com/resources.htm>.

For practical examples of what this license lets you do (*which is a lot!*), see the *Open Font License* FAQ at https://scripts.sil.org/OFL-FAQ_web.



Motivation

The intent of the open-source approach is to foster a community of ongoing development and refinement. Kurinto was built on many instances of this open-source approach: *Linux*, *Apache*, *Firefox*, *WordPress*, *BIND*, *LibreOffice*, *FontUtils*, *FontTools*, and many (many) fonts released under licenses compatible with the *OFL*.

The more people that you have access to, the better the selection of ideas at your disposal, and the more you can recombine into something brand new. This is the key to cultural evolution.

Someone thinks of something. Someone else adapts it to a brand-new environment. Someone incrementally improves on it. Copying, recombining, and learning from each other.

This is what humans are good at. And this is the key to our intelligence.

— Michael Muthukrishna, London School of Economics

To promote ongoing development and collaboration, Kurinto is copyrighted and licensed under the *OFL*, which contain key permissions that promote sharing and continued development. This license

is a Free Cultural license (www.FreedomDefined.org) which contains “Attribution” and “ShareAlike” conditions.

When distributing a release package (original or modified) or any item that is covered by the License, the license requires you to give appropriate credit (“Attribution”) and provide a link to the license. You may do this by simply prominently displaying the text contained in the `License.txt` file, including the link to www.Kurinto.com. You should also describe any modifications that have been made to the release package. You may do so in any reasonable manner, but not in any way that suggests that any of the authors (see the `Authors.txt` file) endorse you or your use.

If you modify or build upon the material in the Kurinto Font Folio release package, you must distribute your contributions under the same license as the original (“ShareAlike”). Also, you may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

The Pan-Unicode Debate

Pan-Unicode fonts such as many of the fonts in Kurinto include many writing systems in each font file. There has been a long-standing debate about the need for pan-Unicode fonts, rather than a well-curated collection of language-specific fonts.¹³ The issues are roughly:

- Glyphs from different scripts are often not aesthetically integrated across script blocks;
- Font designers are rarely schooled in the typographic traditions of multiple cultures, so glyphs in different writing system may vary widely in technical or aesthetic quality; and
- Differences between cultures that use the same writing system lead to situations where text rendered adequately in one culture may appear odd or inappropriate in another culture. Arabic script for both Arabic language text in Egypt and Urdu in Pakistan is a prime example.

In the world of typographic design, these arguments are certainly valid. However, given the goals and audience for Kurinto, I believe that it comes down to the aphorism:¹⁴

The Enemy of the Good is the Perfect

I expect that Kurinto would be of most use to authors writing primarily in a European language, but who *occasionally* include text in other (secondary) writing systems. These authors are, themselves, unlikely to be schooled in those secondary writing systems.

¹³ See, for example, Ed Trager, *Why We Don't Really Need Pan-Unicode Fonts Anymore*, <http://unifont.org/fontguide/>, January 10, 2008, retrieved January 15, 2018.

¹⁴ Commonly attributed to Voltaire, based on a publication in 1770, often attributed to the anti-extremism principles of Confucius and Aristotle.

In this setting, authors often have a choice between:

1. A “perfect” rendering of the secondary text using diverse, language specific fonts – which often (in my experience) invokes many of the project-ending [pitfalls](#) of typesetting, and
2. A “good” rendering of the secondary text that smooths the path to publication.

The philosophy of Kurinto is the second approach.

The Under-Served

Well-designed fonts for many of the underserved writing systems have been slow in coming. Rough estimates place the number of typefaces at 60,000, containing 300,000 fonts.¹⁵ The vast majority of those serve European languages. As of early 2020, there was a single font that served native speakers of Hanfi Rohingya. Since the 360 million native English speakers¹⁶ have a choice of 60,000 typefaces, one might expect that the 1.8 million native speakers of Hanfi Rohingya would have, using the same proportions, about 300 typefaces. However, as of early 2020, they had one typeface.

Disenfranchising is often taken to mean the inability to vote. However, I would say that participation in on the world stage is dependent on a far more basic right – the fundamental right to communicate. Without a way to digitally express in your language, that is essentially impossible in today’s world. No E-mail, no Google Translate, no Wikipedia, ...

One of the goals of Kurinto is counter these biases by providing easier access to fonts for underserved writing systems. How important is this? Mike Jacobs provided an eloquent argument for the value of efforts in this area:¹⁷

Support for sophisticated typography belongs among the important things, and not relegated to a frill. The role of typography is not to prettify text, but to articulate it. That it does so in an aesthetic way—utilizing all the art it can draw from its own heritage, the heritage of manuscript tradition, and individual creative vision—should not disguise the expressive and organizational relationship of typography to text. A typographic culture, such as the one in which you engage as you read this article, is a system of visual indicators that helps readers navigate text and helps writers express their ideas. In the 550 years since Gutenberg developed metal type casting at Mainz, the printed Latin script has developed a particularly rich typographic culture, using romans, italics, bold type, smallcaps, ligatures, swash forms, etc., to organize and articulate the texts of hundreds of languages around the world. Other scripts have developed equally complex and adaptive systems, some more complex and sophisticated, while others are only just beginning their typographic journey. Typography is part of how the human race expresses itself, individually and

¹⁵ Estimates by Thomas Phinney, April 24, 2019, from <https://www.quora.com/How-many-fonts-are-there-in-the-world>.

¹⁶ See <https://www.babbel.com/en/magazine/how-many-people-speak-english-and-where-is-it-spoken>.

¹⁷ See <https://docs.microsoft.com/en-us/typography/opentype/processing-part2>, retrieved April 29, 2020.

collectively. Sadly, whenever support for a particular aspect of a typographic culture is limited, texts are inevitably rendered less expressive of the ideas they contain than they might be. As if we were forced to speak in a monotone, we cannot fully articulate what we need to say.

Acknowledgements

Contributors.txt contains detailed and extensive acknowledgements of the designers who contributed to this effort. In addition, I would like to personally thank these people and organizations who contributed to directly to the development of Kurinto:

Chris Ludwig of [Fundamental AV](#) constructed the servers used to render the Kurinto “pipeline” process, which is used to build each version of the Kurinto fonts. This project would not have been possible without the speed and reliability of those systems.

Tarrin Wills and Odd Einar Haugen of the [Medieval Unicode Font Initiative](#), who provided direct access to the MUFI database.

Erwin Denissen and Bhikkhu Pesala, for their responsiveness to requests regarding the FontCreator project.

PREAMBLE

The goals of the Open Font License (OFL) are to stimulate worldwide development of collaborative font projects, to support the font creation efforts of academic and linguistic communities, and to provide a free and open framework in which fonts may be shared and improved in partnership with others.

The OFL allows the licensed fonts to be used, studied, modified and redistributed freely as long as they are not sold by themselves. The fonts, including any derivative works, can be bundled, embedded, redistributed and/or sold with any software provided that any reserved names are not used by derivative works. The fonts and derivatives, however, cannot be released under any other type of license. The requirement for fonts to remain under this license does not apply to any document created using the fonts or their derivatives.

DEFINITIONS

"Font Software" refers to the set of files released by the Copyright Holder(s) under this license and clearly marked as such. This may include source files, build scripts and documentation.

"Reserved Font Name" refers to any names specified as such after the copyright statement(s).

"Original Version" refers to the collection of Font Software components as distributed by the Copyright Holder(s).

"Modified Version" refers to any derivative made by adding to, deleting, or substituting -- in part or in whole -- any of the components of the Original Version, by changing formats or by porting the Font Software to a new environment.

"Author" refers to any designer, engineer, programmer, technical writer or other person who contributed to the Font Software.

PERMISSION & CONDITIONS

Permission is hereby granted, free of charge, to any person obtaining a copy of the Font Software, to use, study, copy, merge, embed, modify, redistribute, and sell modified and unmodified copies of the Font Software, subject to the following conditions:

1) Neither the Font Software nor any of its individual components, in Original or Modified Versions, may be sold by itself.

2) Original or Modified Versions of the Font Software may be bundled, redistributed and/or sold with any software, provided that each copy contains the above copyright notice and this license. These can be included either as stand-alone text files, human-readable headers or in the appropriate machine-readable metadata fields within text or binary files as long as those fields can be easily viewed by the user.

3) No Modified Version of the Font Software may use the Reserved Font Name(s) unless explicit written permission is granted by the corresponding Copyright Holder. This restriction only applies to the primary font name as presented to the users.

4) The name(s) of the Copyright Holder(s) or the Author(s) of the Font Software shall not be used to promote, endorse or advertise any Modified Version, except to acknowledge the contribution(s) of the Copyright Holder(s) and the Author(s) or with their explicit written permission.

5) The Font Software, modified or unmodified, in part or in whole, must be distributed entirely under this license, and must not be distributed under any other license. The requirement for fonts to remain under this license does not apply to any document created using the Font Software.

TERMINATION

This license becomes null and void if any of the above conditions are not met.

DISCLAIMER

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

The Release Packages

A Kurinto release package is a .ZIP file that can be downloaded from <https://Kurinto.com>. The .ZIP file contains all the files of that release. A .ZIP file has a name such as:

Kurinto_v1.111_Full_20190602_0920.zip

The **blue** characters indicate the version of the release package. This will match the version number embedded in every font in that release package since all fonts are built from scratch for a release. Even if a particular font has not been altered since a prior release, it will carry this new version number.

The **dark red** characters indicate the “flavor” of the package (see below).

The **date stamp** in purple gives the date and time (24-hour clock) on my system that the release was assembled. Note that this date does not match the date stamps embedded in the font files.

Release Package “Flavor”

Each version of Kurinto has three choices of release packages: “Lite”, “Full”, and the “Developer” (or “Dev”). The three release packages have substantially different sizes.

Lite

The Lite package has a basic set of fonts to get started, without having to download and install the rather large package of the Full version. This includes the primary typefaces **KURINTO TEXT**, **KURINTO BOOK**, **KURINTO SANS**, **KURINTO SERI**, and **KURINTO MONO**.

Each typeface has Normal and Narrow width fonts, Regular and Bold weights, and the Main and Core variants.

Full

The Full package has all released fonts. This includes the secondary typefaces that are currently available, together with all the width (from Ultra-Narrow to Ultra-Wide), weights (from Ultra-Light to Ultra-Black), and all variants available (Core, Main, Aux, HK, JP, KM, KR, SC, TB, TC, TW, CJK, Music, and UFI).

It also has PDF documents in the /Doc subdirectory that show all the glyphs of each of the fonts in US-Letter format.

कूरंटो
കൂറിന്റേ
കുറിന്തോ
ବୁରୀନଟେ
庫里特
쿠린토
クリント
Куринто
Kurinto
Κούριντο
קורינטו
קורינט
کورینتو
کتنا
ካፋቶ
કુરુન્ટો
કુરિનાહો
ಕುರಿಂಟೊ
კურიენტო
കുറിന്തോ
කුරින්ටෝ
கேயுரிண்டோ

Developer

The full Developer release (with the file tag “Dev”) has all the files of the Full release, with these additions:

- A parallel set of character map PDF files in display format (30”×18”) with additional information on each glyph.
- The source files and configuration files that control the generation of the Code Charts by the Unibook application.
- Validation reports by various font validation tools on each of the fonts.

Directory Structure

This section outlines the directory structure of the distribution package.

Top-Level Directory

The file in the top-level directory that relate the authorship, copyrights, licensing, and other legal aspects of this Font Software are outlined above in [About the Kurinto Font Folio License](#). Of particular use is the ReadMe.txt file, which has a list of all fonts contained in the particular release package.

In addition to those files, the top-level directory contains:

- **Kurinto_QuickStart.pdf** – A one-page *how-to-get-started-in-a-hurry* document.
- **Kurinto_UserGuide.pdf** – This document.

/Doc

Documentation in PDF format. These files are licensed under the same license as this document and the fonts themselves. These files include:

- **Kurinto_QuickStart.pdf**: A single-page document for getting set up with minimum coaching.
- **Kurinto_SpecimenBook.pdf**: Samples of all the fonts.
- **Kurinto_UserGuide.pdf** – This document.
- **Kurinto_CodeChart_XXX.pdf** – A set of documents that show all non-Unicode code points provided in Kurinto fonts. There is a document for each of the font variants. All the code point ranges are in the three Private Use Areas (PUAs) provided by the Unicode standard.
- **LanguageBook_Typeface.pdf**: Text that covers hundreds of languages. One document each for KURINTO TEXT, KURINTO BOOK, KURINTO SANS, and KURINTO MONO.
- **Sampler_Typeface.pdf**: A one-sheet PDF for each typeface that shows examples of use in various writing systems and styles.
- **OFL_LicenseCompatibility.pdf**: A “white paper” that looks at the legal issues of using source fonts under various licenses in a composite font that is issued under the OFL.

- **UnicodeCheatSheet.pdf**: A one-page quick reference to the most commonly used Unicode characters.

/Docx

The Microsoft Word documents that are the source documents for the PDF documentation in this release – this User’s Guide, the Quick Start Guide, the Specimen Books documents, etc. – are provided in this directory.

This directory also has a set of Word documents and PDF files for papers I have published. The original version as well as versions updated to use Kurinto fonts are included.

These documents can be examined or even used as the starting point for your own publications.

These .docx documents depend on Kurinto fonts, which should be installed on your system before opening these documents. In addition, Kurinto_UserGuide.docx uses the **AMIRI QURAN COLORED** font, which can be installed from /Misc/AmiriQuranColored.ttf.

Other than these font dependencies, these documents are self-contained. There are no external references – all images and content are included in the document files.

/Fonts

The primary directory of TrueType fonts.

/Fonts_Aux

Fonts that have historic writing systems (i.e. those not in active use) and writing systems that are not currently in the Unicode standard (e.g. Klingon).

/Fonts_Core

Font files for the [Core variant](#) of fonts. These fonts are appropriate for use on Web pages because they are substantially smaller than other fonts. See [Appendix 1](#) for the characters contained in these fonts.

/Fonts_Lang

Specialized fonts for Tibetan, Khmer, Mongolian, and Asian writing systems (Chinese, Japanese, and Korean).

/Images

Various bitmap images in PNG format related to Kurinto that might be useful. In particular, I have included high-resolution versions of the design metrics images.

These images are release under the OFL.

/Licenses

The set of licenses under which the source fonts have been distributed. These are .txt files with various encodings.

/Maps

Character Map documents are PDF files that show every character in each typeface.

Kurinto Text

Version 2.139 -- 1-Mar-2020 @ 4:20 PM

ז	ט	י	ך	כ	ל	מ	נ	ס	ף
FB36	FB38	FB39	FB3A	FB3B	FB3C	FB3E	FB40	FB41	FB43
פ	צ	ק	ר	ש	ת	ו	ב	כ	פ
FB44	FB46	FB47	FB48	FB49	FB4A	FB4B	FB4C	FB4D	FB4E

A sample of Kurinto's Character Maps

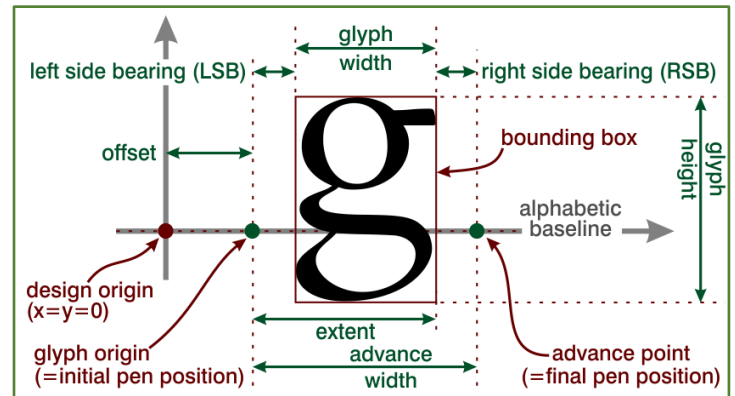
There is one very large *Character Map* PDF document for each regular style font (i.e. not **Bold**, *Italic*, or **Bold Italic**).

The block for each character shows the Unicode code point in the lower right. You can use this hexadecimal code point to enter that character into your document. If you want to enter the character 永, you can look it up (see the sample above). The corresponding code point is U+6C38.

You can also copy (**Ctrl**-**C** on *Windows*) the characters themselves from the PDF document and paste (**Ctrl**-**V**) them into your document.



Note that, for glyphs which are not mapped to any code point, the hexadecimal code point number is replaced by the decimal glyph index in (parentheses). Also note that glyphs are only shown once – even if they are mapped to multiple code points. They are located in the sorted position based on their lowest numbered mapped code point, but the code point that is listed for the glyph is the highest numbered mapped code point.



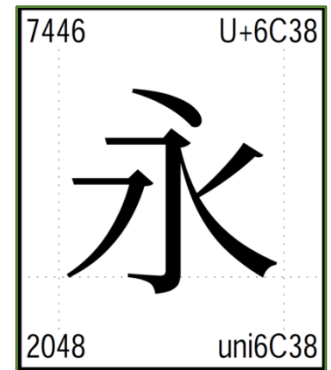
The *Character Map* documents are really big! The files are named: Map_FONTNAME_vv.VVV_Let.pdf. To keep the file names short, FONTNAME is compressed and the “Kurinto” part is removed. These documents are formatted for U.S. letter-sized paper (8½” × 11”, 216 × 279mm).

In addition to the character map PDF files formatted for letter-sized paper, there is an additional set of character map PDF files included in the Developer release package. These files contain additional information about each glyph. They are also formatted for a **much** bigger page size: 30” wide × 18” tall (762 × 457.2mm) – a size I call “display sized”. This size fits nicely onto many contemporary computer displays and lets you view a much larger set of glyphs on each page.

This second set of character map files have names such as: Map_FONTNAME_vv.VVV_Dis.pdf. The **_Dis** indicates that these are Display formatted rather than Letter formatted. They show information in all four corners of each glyph:

- The code point in the upper right (as opposed to the lower right in the Letter-sized character maps),
- The glyph index in the upper left
- The [advance width](#) in the lower left, and
- The glyph name in the lower right.

These augmented glyphs also have dotted lines showing the baseline, left side bearing, and right side bearing.



/Misc

Auxiliary files such as:

- the zero-byte Null.txt file used to calibrate MD5 utilities,
- auxiliary fonts used in some of the sample documents,
- the log file of the pipeline use to construct the fonts, and
- the log file of the release process used to compose the files for the release.

/Unibook

Only in the Developer’s version of the release package. Contain configuration files for the Unibook application used to generation the Code Chart documents in /Doc. Might be useful for anyone attempting to get Unibook working ...

/Validation

Only in the Developer’s version of the release package.

Log files from the various validation tools used to test the generated fonts. Note that these files do contain a number of auto-generated warning and errors that I have tracked down as being spurious or irrelevant.

Installing Fonts

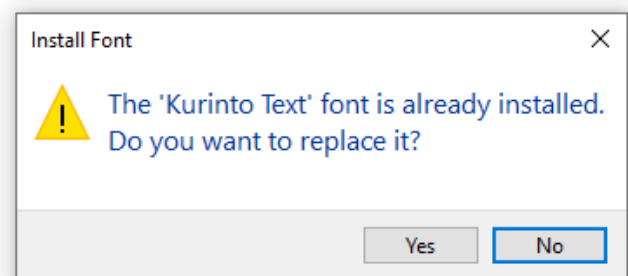
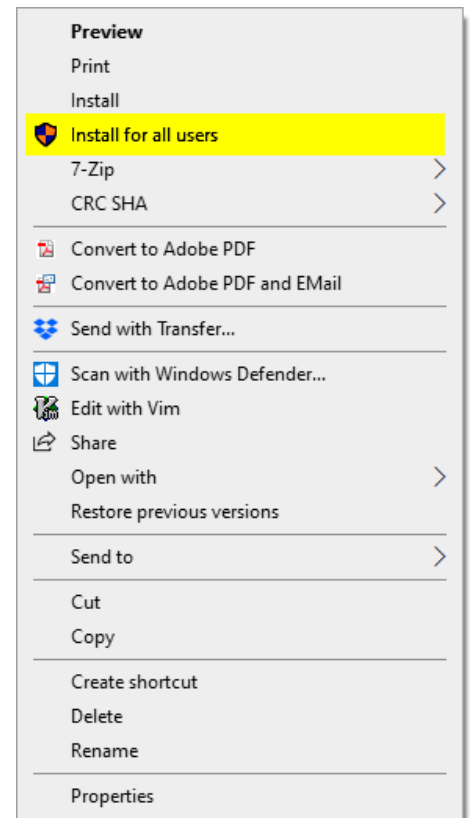
There are many Web-based resources and streaming videos on how to install TrueType fonts on various systems. This section provides one way to install fonts on two of the most common systems.

Windows 10

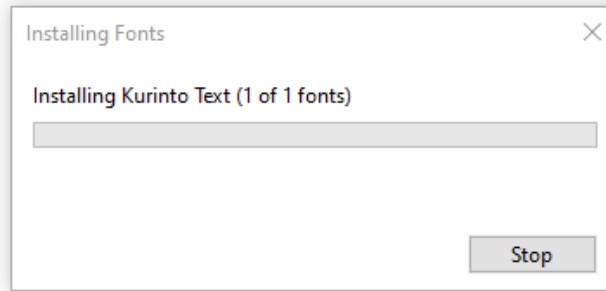
My preference is to install fonts on Windows systems using the **Install for all users** command. This avoids some rather daunting complexities associated with installing fonts “for the Current User”.

However, you will need to be logged in to an account with Administrator privileges:

- Download the release package from www.Kurinto.com.
- Unpack the release package from the .zip file. This can be done on most Windows systems by right-clicking on the .zip file you downloaded and selecting **Extract All ...**. If this does not work, there are many Web-based resources on how to accomplish this step. If you need an unzip utility, 7-Zip is a good option.
- Open Windows Explorer (the Windows “File Manager”) and navigate to the directory (folder) where you unpacked the .zip file. Then move down to the /Fonts sub-directory.
- Right click (or you may be able to press and hold) on the XXX.ttf font files you want to install.
- Click/tap on **Install for all users** menu item. (The **Install** menu item would install for the current user only). See the context menu at the right.
- If this font is already installed, you may this dialog box.¹⁸ Click/tap on **Yes** to replace it:
- The font should now install:



¹⁸ The exact behavior depends on the version numbers of the installed and new fonts. If you are installing a later version, the install may proceed without a warning dialog box. If you are installing the same version, you will be prompted with the warning dialog. If you are overwriting a **later** version of the font with an earlier (older) version, Windows may refuse to install the older font and you will need to explicitly uninstall the font to complete the installation. See [Windows Font Installation](#) for more details.



10. If you intend to use these fonts on Web pages, the fonts in the /Fonts_Core directory should also be installed.
11. If you use Tibetan, Khmer, Myanmar (Burmese), or an Asian language (Chinese, Japanese, Korean), the fonts in the /Fonts_Lang directory should also be installed (not available in the Lite release package).
12. If you use a historic writing system (i.e. one not currently spoken) or a writing system not in Unicode (e.g. Klingon), the fonts in the /Fonts_Aux directory should also be installed (not available in the Lite release package).

If you encounter issues installing on Windows or have a non-standard Windows setup, the suggestions in the [Windows Font Installation](#) section might be helpful.

iOS / Mac Systems

Since I do not work on Apple systems, I am including install instructions from David J. Perry:

Mac OS X: The easiest way is to right-click (or Control-click) on the font file(s) and choose to open with Font Book, which will typically be the default. You can also drag the font(s) from your folder into Font Book. The file will be validated and then appear in the list of fonts.

There are other ways to install fonts in OS X by copying the font file to various locations. You can use this method if you know what you're doing.



Alternately, here is a page on the Apple web site on installing fonts:

<https://support.apple.com/en-us/HT201749>

On-Screen Display

Kurinto fonts are optimized for print rendering.

For best on-screen rendering, it helps to have ClearType set up properly on Windows. When you bring up the ClearType interface, you can immediately see the effect of the changes you make.

To get started, click on the Windows start menu  or  in the lower left and type “ClearText in the search box. You are looking for “Adjust ClearType Text / Control Panel”.

Here are some web references on setting up ClearType:

- <https://www.boxaid.com/blog/how-to-fix-jagged-poor-quality-fonts-or-text-on-windows/>
- <https://www.winhelp.us/change-font-smoothing-in-windows.html>
- <https://superuser.com/questions/807951/fonts-smoothing-on-windows-how-to-disable-cleartype-but-still-get-a-smoothing>
- <https://superuser.com/questions/803637/how-to-disable-directwrite-in-google-chrome-37>
- <https://support.google.com/chrome/answer/95290?hl=en>

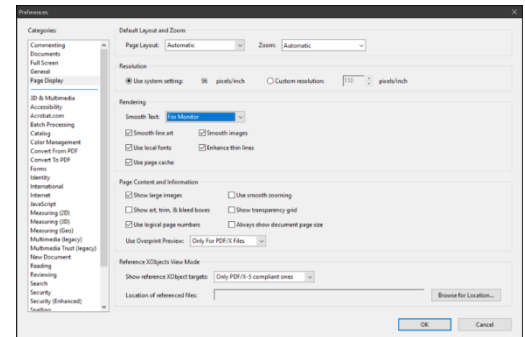
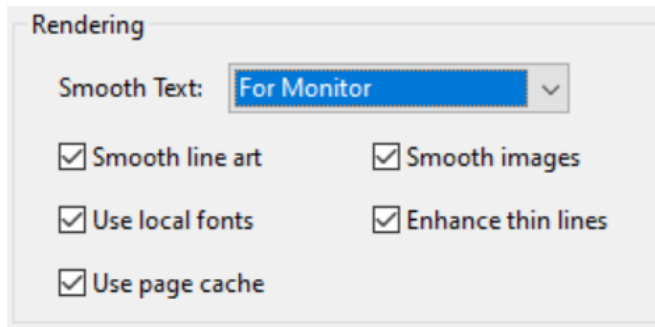
Viewing PDF Files On-Screen

I have been able to dramatically improve the look of PDFs on computer monitors by setting a few configuration parameters. These parameters are in the application you use to view the PDF files (the “PDF viewer”).

One drawback of this situation is that the viewing experience of users of your published PDF files is dependent on **them** to set the parameters on their PDF viewer. You might consider providing them with a version of these instructions, but that is likely to help only a small fraction of your users.

It is beyond the scope of this document to provide specific advice across the huge variety of PDF viewers. However, the general approach is to find the configuration parameters in your PDF viewer, experiment with the settings, and re-open the document to see the effect. Here is one example that has worked for me:

For Adobe Acrobat Pro v9.0, open a PDF file and select Edit ►►► Preferences ►► Page Display. The Preferences window (shown at the right) has a section for “Rendering”:



It is not always obvious what settings will improve your on-screen renderings ... you may need to experiment and re-load your PDF reader to see the results.

Using Unicode

An important part of good typesetting is using the full range of Unicode characters that are available. Getting set up with an easy way to (A) locate and (B) enter Unicode characters into your documents will go a **long** way to achieving that goal.

Locating Unicode Characters

To locate Unicode characters, I use these resources:

- the *Unicode Cheat Sheet*, included as a PDF with each release package,
- the *Unicode Code Charts* documents or on-line resources,
- the **Kurinto_CodeChart_XXX.pdf** documents,
- the *BabelMap* application, or
- one of the Kurinto map files for the specific font I am using (included in the release packages).

The Unicode Cheat-Sheet

This single-page document is designed to help authors quickly look up code points for many common characters. It is organized by character groups, with:

- quotes and brackets on the upper left,
- the sets of ligatures, currency symbols, math symbols, fractions, and arrows, organized in their own blocks,
- the common superscripts and subscripts in their own block with the fraction slash character (which can be used to compose arbitrary fractions),
- space characters of various widths shown with a white background ordered by increasing width, and
- the extensive set of diacritics for the Latin letters at the bottom.

The color coding is used to distinguish between characters in the [Core Characters set](#) from those in the Main fonts.

This sheet is available as a PDF file in the Kurinto distribution package. Print one (or several) out and keep them handy.

Unicode Code Charts and Indexes

The *Unicode Code Charts* are available as a set of PDF files that can be downloaded from

<http://www.unicode.org/charts/>. Each of the PDF files provides detailed information on the characters in a specific block of code points. A sample of the main character display is shown at the right, with the code point and a sample character for that code point.

If you have a system with sufficient memory and speed to deal with a very large PDF (2,621 pages and over 108MB), then you may prefer the monolithic code chart PDF at

<http://www.unicode.org/Public/12.1.0/charts/> (for version 12.1 of Unicode).

There are also numerous Web-based resources for locating Unicode characters. Here are several:

- Unicode Character Index – <https://Unicode.org/charts/charindex.html>
- Unicode Script Charts – <https://Unicode.org/charts/script/index.html>
- Unicode Lookup – <https://UnicodeLookup.com/>
- Compart AG – <https://Compart.com/en/unicode/>
- Scarfboy – <http://Unicode.Scarfboy.com/>

Locating Non-Unicode Characters

Characters in Kurinto that are outside of the Unicode Standard are often called “Private Use Area” characters (PUA).

Code Charts

You can locate specific PUA characters in the various **Kurinto_CodeChart_XXX.pdf** documents. These documents are the most detailed regarding annotations about individual characters.

IPA Extensions						
	025	026	027	028	029	02A
0	ɐ 0250	ɡ 0260	ɥ 0270	ʀ 0280	ʐ 0290	ɕ 02A0
1	ɑ 0251	ɡ 0261	ɱ 0271	ʁ 0281	ʑ 0291	ʑ 02A1

*A sample from the
Unicode Code Charts*

Maps Documents

You can locate specific PUA characters using the set of PDF documents in the /Maps subdirectory of the distribution package. These documents show every character in each typeface:

Kurinto Text Aux									
Version 2.182 -- 10-Jun-2020 @ 6:31 PM									
𐌖	𐌗	𐌘	𐌙	𐌚	𐌛	𐌜	𐌝	𐌞	𐌟
A872	A873	A874	A875	A876	A877	E000	E001	E002	E003
𐌠	𐌡	𐌢	𐌣	𐌤	𐌥	𐌦	𐌧	𐌨	𐌩
E004	E005	E006	E007	E008	E009	E00A	E00B	E00C	E00D
𐌪	𐌫	𐌬	𐌭	𐌮	𐌯	𐌰	𐌱	𐌲	𐌳
E00E	E00F	E010	E011	E012	E013	E014	E015	E016	E017

A sample of Kurinto's Character Maps

You can browse the character map documents and locate a character by shape. The hexadecimal number in the lower right of each block give the code point. If you want to enter the Tengwar character “𐌣”, you can look it up (see the sample above). The corresponding code point is U+E007.

Text Input, or “How to Type Unicode Characters”

To get enter Unicode text, you will want to set up your keyboard for your primary writing system. If you use multiple writing systems, it helps to have a smooth way to switch your keyboard between the writing systems that you use most. In addition, you will need a tool to assist in entering Unicode characters into your document.

Microsoft Word

To enter a code point such as U+1234, type 1 2 3 and 4. This will enter “1234” into your document. Then hold down the Alt key and type X (this is often written as Alt + X). *Word* will convert the “1234” into “𐌖” – the Unicode ETHIOPIC SYLLABLE SEE.

There are some understandable “gotcha’s” with this system. When you type Alt + X, *Word* will interpret whatever digits precede the cursor as a Unicode code point. For example, if you are trying to enter “1𐌖” and have the text “11234”, the Alt + X approach will get you the code point for U+11234, which will be “𐌖” – a combining character in the Khojki writing system. In this situation, I will typically add a space character before your code point to get the character you want and then remove the extra space after the character is converted.

Web Pages

If you are editing HTML documents for web pages, you can enter any Unicode code point using the syntax, for example, `Ӓ`. However, the number in this syntax is a decimal (base 10) version of the code point number.

So, if you wish to enter 永 – the Han Chinese kanji character for “eternal”¹⁹ at code point U+6C38 – you can translate the hexadecimal 6C38 into decimal 27,704 and enter `永` into your HTML source code. You can do this conversion using an on-line “hex-to-decimal” conversion utility, or look up the corresponding character in [KurintoCharacterList.pdf](#).

As of this writing, some (but not all) browsers will interpret the syntax `永` as a hexadecimal number.

Adobe and Open Office

I do not currently use these applications, but I have been told that *Adobe InDesign* and *OpenOffice* (www.OpenOffice.org) have custom displays of the palette of glyphs in a given font. You can select the characters and enter them by clicking on the code point you want.

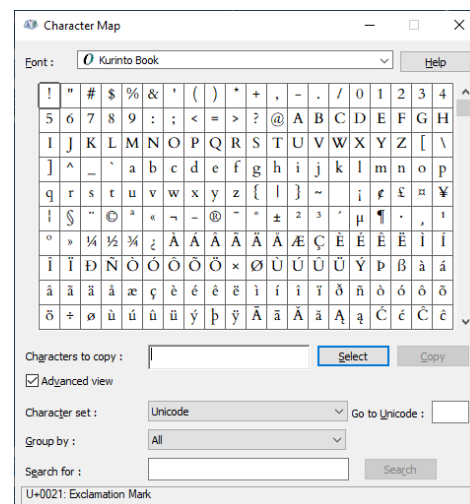
Windows

For other applications on *Windows*, you may need to use a separate application to produce the character so that it can be entered into your text.

Character Map is a standard Windows application to display the characters of any font. If you check the **Advanced View** option, you can select **Unicode** for the Character Set. Once you locate a character, you can copy it to the clipboard and paste it into your document.

However, as of January 2020 in Windows 10, *Character Map* has numerous flaws that limit its usefulness:

- Characters are displayed in a single, tiny, fixed size.
- The application does not show any characters above U+FFFF.
- For reasons unexplained, it does not display certain characters in that range below U+FFFF.

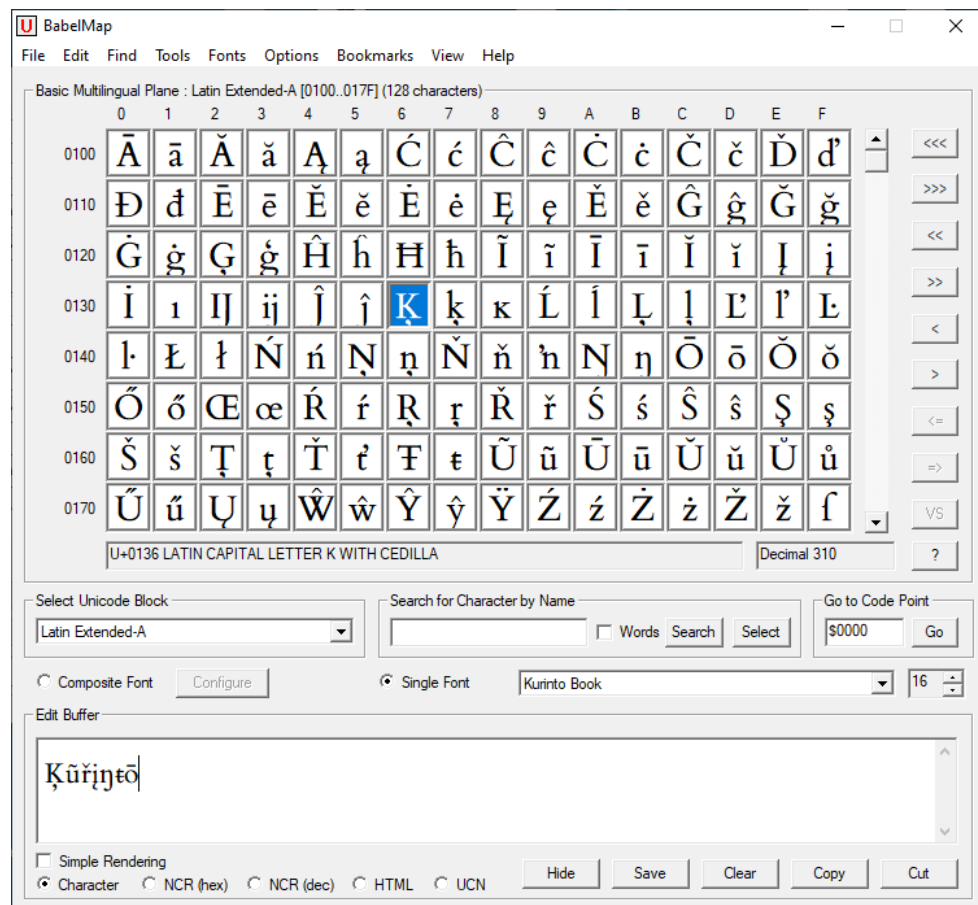


Windows Character Map

These limitations severely limit the usefulness of *Character Map* for practical typesetting.

¹⁹ See <https://en.wiktionary.org/wiki/%E6%B0%B8>.

An excellent alternative is Andrew West's *BabelMap*, freely available at



Andrew West's *BabelMap*

<http://www.Babelstone.co.uk/>. It is similar in intent to Windows *Character Map*, but has vastly more functionality, features, and support.

The *Keyman* program (<https://Keyman.com/> from SIL International) is another Windows application that I have not explored, but appears to be useful. It supports over 1,000 keyboard layouts.

Linux and MacOS

I do not currently used Linux systems for font development, so I cannot offer advice. However I have heard good reports on these options:

- On Linux systems such as Ubuntu, *KMFL* (<http://kmfl.sourceforge.net/>) is available.
- *Ukelele* (<http://scripts.sil.org/ukelele>) is available for Mac OS X versions 10.2 and later.
- *XKB* (<http://www.x.org/wiki/XKB>) may be useful.
- The utilities *gucharmap* and *kchaselect* let you access the full Unicode range on Ubuntu systems.



Using the Fonts

*“With twenty-six soldiers of lead
I have conquered the world”
– James Mosley²⁰*

This section describes how the fonts are designed and how best to use them. It also explains some of the design choices that I have made.

Segmentation and Font Variants

Kurinto fonts handle all the characters and writing systems defined by Unicode – 137,994 characters in 150 writing systems as of Unicode version 12.1. Kurinto fonts also include an additional 30,097 characters in 77 “auxiliary” writing systems and character collection. This brings the total to 168,097 characters in 227 writing systems.

Current font technology does not handle more than 65,535 glyphs (character shapes) in a single font file.

To handle this restriction, Kurinto fonts are segmented into font

“variants”. The variants are designed to minimize the problems created by limitations on character counts. The variants are segmented by writing systems (rather than by plane or block) so that you are less likely to need to switch font variants.

The remainder of this section outlines which writing systems are contained in each of the font variants. For a complete list of precisely which writing systems and the code points contained in each font variant, see the **Scripts.txt** file that is included in each release package.



Contemporary Myanmar and Historic Pyu writing systems at the Myanmar National Museum, Yangon, December 2018. Photo: Vera Shanov

²⁰ James Mosley. *The Caslon foundry in 1902: selections from an album*, Matrix 13 (1993), pp. 34–42.

Type face	Width (blank = Normal)	Core	(Main)	Aux	Music 2,791 addl. music chars + Main	UFI Medieval Unicode Font Initiative + Cyrillic + Runic
Text		R B / <i>BI</i>	R B / <i>BI</i>	R B / <i>BI</i>	R B / <i>BI</i>	
Book		Lt R B Ltl / <i>BI</i>	Lt R B Ltl / <i>BI</i>	Lt R B Ltl / <i>BI</i>		R B / <i>BI</i>
Sans		R SB B / SBI <i>BI</i>	R SB B / SBI <i>BI</i>	R B / <i>BI</i>	R B / <i>BI</i>	
Seri		R B / <i>BI</i>	R B / <i>BI</i>	R B / <i>BI</i>		
Mono	Narrow		R B / <i>BI</i>			
		R B / <i>BI</i>	R B / <i>BI</i>			
	Semi-Wide		R B / <i>BI</i>			

R=Regular, **B**=Bold, *I*=Italic, ***BI***=Bold Italic,
Lt=Light, *Ltl*=Light Italic, SB=Semi-Bold, *SBI*=Semi-Bold Italic

Green = Kurinto Lite & Full Distrib.	Blue = Kurinto Full Distrib. Only	White = Possible in future releases	Grey = No plans for implementation
---	--------------------------------------	--	--

Core Variant

The Core fonts contain the most frequently used characters. This includes the set of 657 characters from the Pan-European character set called Windows Glyph List 4 (“WGL4”).²¹ These fonts also contains additional characters recommended by SIL / NRSI and Google as well as a handful of characters that I consider important.

The set of characters in the Core fonts are listed in [Appendix 1, The Core Character Set](#).

The Core font variants can be used on Web pages, since they are smaller than the other font variants.

Core fonts have names such as **KURINTO BOOK CORE** and **KURINTO TEXT CORE BOLD ITALIC**. The corresponding font files would have names such as KurintoBook**Core**-Rg.ttf and KurintoText**Core**-BdIt.ttf. These font files are located in the /Fonts_Core subdirectory of release packages.

Commonality

All the font variants for a given typeface have the identical character shapes for the Core character set. This means that these core characters will appear identical regardless of which font variant you happen to be using.

²¹ See https://en.wikipedia.org/wiki/Windows_Glyph_List_4.

For (an extreme) example, if you are writing a Khmer-language document on Klingon mating rituals using the **KURINTO BOOK** typeface, you will be using the fonts **KURINTO BOOK KM** for text in Khmer and **KURINTO BOOK AUX** for text in Klingon. However, as with many non-European language documents, there are some sections of text that use European characters.²²

ពិធីរៀបអាពាហ៍ពិពាហ៍វីលីនតុន 《Klingon Mating Rituals》

១១៤២៤- ៦៩៩៩៩៩៩៩- ៩៩៩៩៩ ៤៨៩៩៩

...

Since both **KURINTO BOOK KM** and **KURINTO BOOK AUX** share the same Core character set, the sections of European text can be used from either of the fonts with the same results.

Main Variant

The primary fonts for authoring multi-lingual documents.

The Main variant fonts include all 90 writing systems that are not marked as “Ancient” or “Historic” (e.g. Cyrillic, Cherokee, Georgian, Thai, etc.). They also include the [Core characters](#) (the WGL4 subset) as well as the most common 21,792 characters of the Han writing system.

The term “Main” is not explicitly included in the name of the font or the file name of the .ttf file. Main variant fonts have names such as **KURINTO BOOK** and **KURINTO TEXT BOLD ITALIC**. The corresponding font files would have names such as `KurintoBook-Rg.ttf` and `KurintoText-BdIt.ttf`. These font files are located in the /Fonts subdirectory of release packages.

Aux Variant

Fonts designed for working with ancient and historic languages that are part of the Unicode Standard, as well as auxiliary writing systems that are not currently part of Unicode.

The Hist variant fonts include all 60 writing systems marked as “Ancient” or “Historic” (e.g. Cuneiform, Nabataean, Runic, etc.). They also include the [Core characters](#) (the WGL4 subset), 54 additional writing system that are not currently part of Unicode (e.g. Visible Speech, Kinya, Klingon, etc.), and 22 additional blocks of characters (e.g. Icons, Extended Hieroglyphs, and the characters of the Medieval Unicode Font Initiative).

Aux fonts have names such as **KURINTO BOOK AUX** and **KURINTO TEXT AUX BOLD ITALIC**. The corresponding font files would have names such as `KurintoBookAux-Rg.ttf` and `KurintoTextAux-BdIt.ttf`. These font files are located in the /Fonts_Aux subdirectory of release packages.

²² This sample of a combined Khmer-Klingon-English document was constructed with the assistance of Google Translate, The Klingon Translator at <https://www.kli.org/>, and the pIqaD encoder at <https://dadap.github.io/pIqaD-tools/universal-transliterator/>.

Music Variant

The **MUSIC** variant of Kurinto fonts contains an extensive character set defined by the Standard Music Font Layout (SMuFL) Version 1.3 specification (see <https://www.w3.org/2019/03/smufl13/>).

Music font variants have font names and file names such as **KURINTO BOOK MUSIC** and KurintoBook**MUSIC**-Rg.ttf. These font files are located in the /Fonts_Aux subdirectory of release packages.

For a complete roster of characters, see the Kurinto_CodeChart_Music.pdf document in the /Doc subdirectory of the distribution package.

UFI Variant

A font variant that provide the characters of several Unicode Font Initiatives, including MUFI (the Medieval Unicode Font Initiative, CYFI (the Cyrillic Font Initiative), and RUFI (the Runic Font Initiative).

UFI font variants have font names and file names such as **KURINTO BOOK UFI** and KurintoBook**UFI**-Rg.ttf.

These font files are located in the /Fonts_Aux subdirectory of release packages.

Asian Language Variants

Type face	Width <i>(blank = Normal)</i>	HK	JP	KM	KR	SC	TB	TC	CJK
		Hong Kong + Core	Japanese + Core	Khmer & Myanmar + Core	Korean + Core	Simp. Chinese + Core	Tibetan + Core	Trad. Chinese + Core	Rare Kanji + Core
Text		R B B	R B B	R B B	R B B	R B B	R B B	R B B	R B B
Book		R B B	R B B	R B B	R B B	R B B	R B B	R B B	R B B
Sans		R B B	R B B	R B B	R B B	R B B	R B B	R B B	R B B
Seri		R B B	R B B	R B B	R B B	R B B	R B B	R B B	R B B
Mono	Narrow								
		R B B	R B B		R B B	R B B		R B B	R B B
	Semi-Wide								

Fonts on this chart are only in the Full Distribution.

Color indicates the glyph style:

Gothic style CJK glyphs	Mincho style CJK glyphs	Hanazono style CJK glyphs
Sans serif style glyphs		Serif style glyphs

The Asian-language variants include:

- fonts for the logographic “CJK” (Chinese, Japanese, and Korean) languages that require over 65 thousand characters, and
- the languages Khmer, Myanmar, and Tibetan that have extensive character-shaping directives, requiring them to be packaged in a different OpenType font file.

All Asian Language font file variants are located in the /Fonts_Lang subdirectory of release packages.

CJK Glyph Flavors

Kurinto uses three flavors of CJK characters: Gothic, Mincho, and Hanazono:

	Regular	Italic	Bold	Bold-Italic
Gothic	返	返	返	返
Mincho	返	返	返	返
Hanazono	返	返	返	返

- **Gothic:** the Asian equivalent of “sans serif”. This style is used for **KURINTO SANS**, **SERI**, **PLOT**, **TYPE**, **ARIA**, and **CALI**. The glyphs are based on **NOTO SANS CJK**.

KURINTO MONO also uses a Gothic style, but the glyphs are derived from **SOURCE HAN MONO**.

However, both **NOTO SANS CJK** and **SOURCE HAN MONO** share a common lineage, so the results are very similar.

- **Mincho:** the Asian equivalent of “serif”. This style is used for Kurinto serif typefaces: **TEXT**, **ARTE**, and **TMOD**. The glyphs are based on the open-source **NOTO SERIF CJK**.
- **Hanazono:** a variation of the Mincho style. These glyphs are derived from the open-source **HANAZONO** fonts, which provides a more complete coverage of the Unicode code points for the Han writing system used by the CJK languages. They are used in the Asian-language variants of **KURINTO BOOK**. **HANAZONO** includes the rare and historic logographs that are not available in the **NOTO CJK** fonts.

Most CJK fonts do not provide italics, since they are rare in Asian languages. However, I have included a modified version of italics. The slant angle – often set in the range of 10°– 14° – is set to 6° for the five CJK variants of fonts (described in the next section). Rather than using italics for emphasis, the italic styles could be used in situation such as headings or promotional material, where you might want a more “active” look.

Note that the Hanazono flavor of characters do not have Bold and Bold-Italic styles. Hanazono also has another limitation, that will be described in the next section.

The CJK Variants

As a group, the CJK Asian-language variants share the common Han writing system. However, there are significant regional variations to the shape of many of the Han characters, so they are provided in five separate language variants.²³

- **HK:** Traditional Chinese – Hong Kong (繁體中文–香港), the regional flavor of Traditional Chinese as used in Hong Kong. Supports Big Five and HKSCS-2016, with the glyphs for both mostly adhering to Hong Kong conventions.²⁴
- **JP:** Japanese (日本語), which also uses the Katakana and Hiragana writing systems as well as a regional flavor of Han. Supports all the kanji in JIS X 0208, JIS X 0213, and JIS X 0212 to include all kanji in Adobe-Japan1-6.²⁵
- **KR:** Korean (한국어 / 韓國語), which also uses the Hangul and Hanja writing systems as well as a regional flavor of the Han writing system. Supports over 1.5 million archaic Hangul syllables and 11,172 modern syllables as well as all CJK ideographs in KS X 1001 and KS X 1002.
- **SC:** Simplified Chinese (简体中文), the regional flavor of the Han writing system in wide use since the late 1950s in mainland China. This flavor is also in current use in Singapore. Supports the GB 18030 Standard and China’s *Table of General Chinese Characters* (通用规范汉字表 – [PRC-Edu 2013]).
- **TC:** Traditional Chinese – Taiwan (正體中文– 臺灣), used primarily in Taiwan (the Republic of China). Supports BIG5. Traditional Chinese glyphs comply with the glyph standard of the Taiwan Ministry of Education (教育部國字標準字體).

While the Han writing system includes over 89,000 logographic characters, contemporary communication is typically limited to a subset of $\approx 40,000$ logographs. Each of the five CJK variants listed above contains the Core character set, plus the regional variations of the set of $\approx 40,000$ logographs needed for virtually all contemporary communication.

Font names and file names are, using examples of the HK variant, **KURINTO BOOK HK** and **KURINTO TEXT HK**. The font files can be found in the /Fonts_Lang subdirectory and have names like KurintoBook**HK**-Rg.ttf. and KurintoText**HK**-Rg.ttf.

This table shows the flavors of U+8FD4 in each of the five variants. Look carefully at the character shapes in the Gothic row to identify the differences ...

²³ See [Variant Identifiers](#) for how these variant identifiers were chosen.

²⁴ The “supports” statement for HK is from the ReadMe.pdf file of Source Han Mono Version 1.002, released June 3, 2019.

²⁵ The “supports” statements for JP and the remaining variants are based on the Help text for Noto CJK, retrieved from <https://www.google.com/get/noto/help/cjk/> on March 31, 2020.

	HK Trad. Chinese – Hong Kong 繁體中文-香港	JP Japanese 日本語	KR Korean 한국어 韓國語	SC Simplified Chinese 简体中文	TC Trad. Chinese – Taiwan 正體中文-臺灣
Gothic	返	返	返	返	返
Mincho	返	返	返	返	返
Hanazono	返	返	返	返	返

In some cases, the differences are subtle. Looking at the Gothic row, HK and TC differ only the curve of the top stroke. JP and SC differ in the close of the central “x” stroke. However, these differences do call for different font variants.

Looking at the Mincho row, you might notice that the HK and TC variants are the same. For this code point, the top stroke of the Mincho HK glyph does not curve upward. This is a short Noto Serif CJK, the source font for Mincho, does not provide an HK variant of the font.

The HK variant fonts that use the Mincho flavor of glyphs is identical to the TC font variant.

This bottom row of the table also highlights another shortcoming of the Hanazono flavor – a font that was created from the Japan-based GlyphWiki project:²⁶

The Hanazono flavor of characters are only available in the JP style. Therefore, all CJK font variants of Kurinto Book – HK, JP, KM, SC, and TC – use the Japanese style of characters.

²⁶ See <http://glyphwiki.org/> or <http://en.glyphwiki.org/>, which has over 580,000 hanzi registered as of March 31, 2020.

The example character above highlights a single character that is different in each of the five flavors. However, in practice, such variations are sparse. Here are translations of Article 1 of the Universal Declaration of Human Rights in each of the five flavors, with the differing glyphs highlighted in red:

Kurinto Text

HK 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

JP 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

KR 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

SC 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

TC 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

Kurinto Sans

HK 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

JP 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

KR 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

SC 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

TC 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。

The CJK Variant

In addition to **JP**, **KR**, **HK**, **SC**, and **TC**, there is an additional variant called **CJK**. In addition to the [Core Character Set](#), the **CJK** variant contains all logographs that are missing from the subset of all logographs that are not in the $\approx 40,000$ character subset.

This variant has font names such as **KURINTO BOOK CJK** with corresponding font file names such as KurintoBook**CJK**-Rg.ttf.

The KM and TB Variants

Certain writing systems have complex conjunct formations, positioning rules, or ligatures. The fonts that represent these writing systems require extensive OpenType coding to correctly represent them. In the same way that OpenType files have a limit on the number of glyphs, they also have limits on the amount of OpenType coding in a single file. While any of these writing systems on their own would not exceed that limitation, combining them with other fonts in a single composite font cannot be done.

Three writing systems – Khmer, Myanmar (Burmese), and Tibetan – fall into this category. These writing systems are available in two font variants: **KM** (Khmer and Myanmar) and **TB** (Tibetan).

Each of these variants includes the [Core Character Set](#). These fonts have names such as **KURINTO BOOK KM** with corresponding file names such as KurintoBook**KM**-Rg.ttf. All these fonts are located in the /Fonts_Lang subdirectory of release packages.

Styles

Each typeface has fonts with different weights and slants. At a minimum, fonts are provided in the **RBIBI** styles:

- **Regular.** The name refers to the weight of these fonts, and is also called in various contexts: “Normal”, “Plain”, “Roman”, and “Standard”. The style “Regular” is not part of the font name, so fonts have names such as **KURINTO BOOK**. The file names of these fonts end in “-Rg.ttf”.
- **Bold.** Fonts have names such as **KURINTO BOOK BOLD** and the file names of these fonts end in “-Bd.ttf”.
- **Italic.** Regular-weight fonts that have a cursive or oblique style (see the next section). The slant of italicized fonts varies, but $+11^\circ$ is common. Fonts have names such as **KURINTO BOOK ITALIC** and the file names of these fonts end in “-It.ttf”.
- **Bold Italic.** Fonts have names such as **KURINTO BOOK BOLD ITALIC** and the file names of these fonts end in “-BdIt.ttf”.

Italics: Oblique versus Cursive

Kurinto uses two distinct styles for italic characters: *Oblique* and *Cursive*.

The cursive italic style originated with Venetian typographers in 1500.²⁷ Their original goals included saving space on the printed page, attaining a more calligraphic look, and improving the aesthetic look of the typeset text. This was done by emulating the look of handwritten text.

Emphasis (as far as I have seen) was never a goal in the original development of italics. However, mixing *cursive italics* with upright text causes it to stand out dramatically.

The *oblique italic style* simply slants the upright text. Oblique uses a “slope” or “skew” and avoids the calligraphic effect of traditional italics. Hoefer and Frere-Jones described obliques as more “keen and insistent”²⁸, which is another approach to achieving emphasis.

Here is a comparison (using 14pt **KURINTO TROM** and **KURINTO TMod**):

Regular: Now the world has one language and a common speech. (TRom)

Cursive: Now the world has *one language* and a *common speech*. (TRom)

Oblique: Now the world has *one language* and a *common speech*. (TMod)

These samples highlight the degree of emphasis and differentiation in these two approaches to italicization. Personally, I like to use oblique italics to denote *Publication Titles* and other text that

²⁷ Hendrik D. L. Vervliet (2008). *The Palaeotypography of the French Renaissance: Selected Papers on Sixteenth-century Typefaces*. BRILL. pp. 287–319. See https://en.wikipedia.org/wiki/Italic_type for a general overview.

²⁸ Tobias Frere-Jones and Jonathan Hoefler. *Whitney*. Retrieved 16 December 2016 from <http://www.typography.com/fonts/whitney/features/whitney-stylistic-sets>.

needs to be distinguished for semantic reasons, and reserve cursive italics for more dramatic contextual emphasis.

Note that the typefaces that use oblique italics are done using an automated process that does have some subtle shortcomings. One example is a careful examination of “O” in its cursive “*O*” and oblique “*O*” forms (these samples use 18pt **KURINTO TROM** and **KURINTO TMod**). The axis of the capital O is nicely rotated in the cursive form. However, the axis of the inner and outer contours produced by the automated oblique process do not result in the same rotation angles.

In general, the primary and secondary Kurinto typefaces use the oblique italics style and the metric compatible typefaces use the cursive italics style.

Cyrillic characters in the typefaces that use the cursive italics style deserve special note: the glyphs for the lower-case characters have a significantly different shape than their upper-case counterparts. For example:

In **KURINTO TEXT**, which uses the oblique italics style, U+0438 CYRILLIC SMALL LETTER I in Bold is **и** and in BoldItalic is ***и***.

In **KURINTO TROM**, which uses the true italic slanting style, U+0438 CYRILLIC SMALL LETTER I in Bold is **и**, but in Bold Italic it has a completely different shape: ***и***.

See https://en.wikipedia.org/wiki/Russian_cursive for the history of this difference.

Underlining and Strikethrough

OpenType fonts have specific settings for the vertical position and thickness of underlining and ~~strikethrough~~ adornments. ~~Double strikethrough~~, double underline, and many other adornments are created in some applications, typically based on the OpenType settings for strikethrough and underline.

Kurinto fonts have a consistent set of rules across all typefaces and fonts:

1. The **vertical position** of the center of the underline and the center of the strikethrough line are positioned a fixed distance above the baseline.
2. The **thickness** of the underline and strikethrough vary with each font. They are set based on the visual weight of the typeface and style (bold, italic, etc).

I chose this scheme (out of many that I tested) so that line of text with mixed typefaces and styles will not have a strikethrough or underline “jump around” vertically. The thickness of the line(s) will change, but all strikethrough and underlines will be centered vertically.

Some examples:

- Underlines set in Kurinto Curv Bold should work with text set in Kurinto Plot²⁹
- Double underlines set in Kurinto Curv Bold should work with text set in Kurinto Plot
- Strikethrough ~~set in Kurinto Curv Bold should work with~~ text set in Kurinto Plot.
- Double strikethrough ~~~~set in Kurinto Curv Bold should work with~~~~ text set in Kurinto Plot.

These examples generally work well, except for the ~~double strikethrough~~, which does jump vertically between different typefaces. I suggest avoiding double strikethrough if you are changing typefaces in a single line of body text.

See [Underlining and Strikethrough Revisited](#) in the Technical Information section for more details.

²⁹ You might also notice that the • on each of the bullet points above has the strikethrough adornment. I do not know how to avoid this behavior in *Word*.

Typefaces

*“Calligraphy is jewelry fashioned by the hand
from the pure gold of the intellect”*

— Abū Hayyān al-Tawhīdī (10th c.)

Typefaces (or “font families”) are groups of fonts that share a similar design.

Kurinto provides 21 font families:

- Five Primary Typefaces that are part of the [Lite distribution package](#) and all other distribution packages,
- Seven additional Secondary Typefaces that are only in the [Full](#) and [Developer’s](#) distribution packages, and
- Nine additional Metric Compatible Typefaces that are in the [Full](#) and [Developer’s](#) distribution packages.

The very brief samples in the following paragraph provide a few key characters – distinctive characters in European writing systems and a sprinkling of example characters from other scripts. See the [Representative Characters](#) section for more info on why these were chosen.

For samples of these fonts that are far more complete, see the *Specimen Book* document in the Kurinto_SpecimenBook.pdf file that is part of each distribution package.

And, yes, each of the typeface names is a four-letter word.



Primary Typefaces

The five primary typefaces provide serve most of the needs of body text in a document. These fonts have the most complete coverage of the Unicode characters across the font variants.

Type face	Width (blank = Normal)	Core	(Main)	Aux	Music 2,791 addl. music chars + Main	UFI Medieval Unicode Font Initiative + Cyrillic + Runic
Text		R B BI	R B BI	R B BI	R B BI	
Book		Lt R B BI	Lt R B BI	Lt R B BI		R B BI
Sans		R SB B SBI BI	R SB B SBI BI	R B BI	R B BI	
Seri		R B BI	R B BI	R B BI		
Mono	Narrow		R B BI			
		R B BI	R B BI	R B BI		
	Semi-Wide		R B BI			

R=Regular, B=Bold, I=Italic, BI=Bold Italic,
Lt=Light, LtI=Light Italic, SB=Semi-Bold, SBI=Semi-Bold Italic

Green = Kurinto Lite & Full Distrib.	Blue = Kurinto Full Distrib. Only	White = Possible in future releases	Grey = No plans for implementation
---	--------------------------------------	--	---------------------------------------

Type face	Width (blank = Normal)	HK Hong Kong + Core	JP Japanese + Core	KM Khmer & Myanmar + Core	KR Korean + Core	SC Simp. Chinese + Core	TB Tibetan + Core	TC Trad. Chinese + Core	CJK Rare Kanji + Core
Text		R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI
Book		R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI
Sans		R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI
Seri		R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI
Mono	Narrow								
		R B BI	R B BI		R B BI	R B BI		R B BI	R B BI
	Semi-Wide								

Fonts on this chart are only in the Full Distribution.
Color indicates the glyph style:

Gothic style CJK glyphs	Mincho style CJK glyphs	Hanazono style CJK glyphs
Sans serif style glyphs	Serif style glyphs	

x1ftRégò'Ê1&
永ओशঐЖボ

A modern typeface designed to make body text pleasant and easy to read. I consider this my primary go-to font for typesetting body text.

This section and most of the body text for this *User's Guide* are typeset in KURINTO TEXT.

Composition

The core character set is based on Charis SIL, developed by SIL International, which was “*specifically designed to make long texts pleasant and easy to read, even in less than ideal reproduction and display environments.*”

The lineage of this font includes Bitstream Charter, one of the first fonts designed for laser printers. The basic character set is similar (but not identical) to Bitstream Charter, designed by Matthew Carter. The following notice accompanies the Bitstream Charter fonts:

(c) Copyright 1989-1992, Bitstream Inc., Cambridge, MA.

You are hereby granted permission under all Bitstream propriety rights to use, copy, modify, sublicense, sell, and redistribute the 4 Bitstream Charter (r) Type 1 outline fonts and the 4 Courier Type 1 outline fonts for any purpose and without restriction; provided, that this notice is left intact on all copies of such fonts and that Bitstream's trademark is acknowledged as shown below on all unmodified copies of the 4 Charter Type 1 fonts.

BITSTREAM CHARTER is a registered trademark of Bitstream Inc.

The fonts in the Kurinto Text typeface are composed of 144 source fonts, including Amiri, Hana-Min, Khmer Busra, many fonts contributed by George Douros, and numerous fonts of the Noto series. See the Map.txt file in the release package for the specific source fonts for this typeface and font.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height and are square, which results in a size for each character of ($\text{Em} \times \frac{11}{8}$).

OpenType Features

Kurinto Text implements an extensive set of OpenType features spanning the many writing systems supported by the Main and Aux font variants. These features are largely inherited from the source fonts that make up Kurinto Text.

Most of the OpenType Features described in [Alternate Characters and Layout Features](#) are implemented for the Latin writing system.

There are extensive OpenType Features for many writing systems, including Greek and Cyrillic. However, the specific additions of Kurinto for Microsoft Word described in [Alternate Characters and Layout Features](#) are not implemented for Greek and Cyrillic.

Likewise, the [ACF block](#) is implemented only for the Latin characters.

xlftRégòÊ1&
永ओشჲ𐎡𐎠

A classic “old-style” serif typeface based on a design from the Italian Renaissance.

It is suitable for academic works and any publication where a classical or historical look is suitable. The diacritics are sized and positioned for legibility.

This section is typeset in **KURINTO BOOK**.

The pan-European characters are derived from the Cardo font developed by David J. Perry. Cardo is based on a metal typeface design by Francesco Griffo for Venetian printer Aldus Manutius, cut around 1495.

Book has three weights: Light, Regular, and Bold. Note that this difference between Light and Regular is quite subtle. This paragraph is set in Kurinto Book Light ... can you detect the difference from the paragraphs above?

There is a one-page sampler – [Sampler_Book.pdf](#) – available in the release package. There are also specimens of this typeface across a wide variety of languages in the file [LanguageBook_Book.pdf](#).

Composition

The fonts in the Kurinto typeface are composed of 75 source fonts, including **AMIRI**, **HANA-MIN**, **GENTIUM PLUS**, **KHMER BUSRA**, many fonts contributed by George Douros, and numerous fonts of the **NOTO** series. See the [Map.txt](#) file in the release package for the specific source fonts for this typeface and font.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height and are half as wide. This results in a height of $(Em \times \frac{1}{8})$ and a width of $(Em \times \frac{1}{16})$.

OpenType Features

Kurinto Book implements an extensive set of OpenType features spanning the many writing systems supported by the Main and Aux font variants. These features are largely inherited from the source fonts that make up Kurinto Book.

All of the OpenType Features described in [Alternate Characters and Layout Features](#) are implemented, and these features are available from the standard OpenType layout features (e.g. smcp), through the alternate OpenType layout features for accessibility from Microsoft Word (e.g. stylistics sets), and directly by code points.

Likewise, the [ACF block](#) is fully implemented.

xlfRregòÊ1&

永 ओ ش ۛ 𐤎 𐤊 𐤍

A clean, Gothic, sans-serif design focused on readability.

This section is set in **KURINTO SANS**.

Composition

The core characters are derived from Noto Sans.

The bold version of Kurinto Sans is distinctive in its compact look.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height and are half as wide.

This results in a height of (Em × $\frac{1}{8}$) and a width of (Em × $\frac{1}{16}$).

xlftRegòÊ1&
永ओشवेЖボ

A Humanist, “semi-serif” design with tapered strokes and very modest serifs.

The name “**SERI**” is a contraction of “semi-serif” (and also a play on truncating the word “serif”). This typeface was originally called “**KURINTO SEMI**”, but that wreaked havoc with some applications which interpreted the name “**KURINTO SEMI BOLD**” as a font named “**KURINTO**” with the style “**SEMI BOLD**”.

This section is typeset in **KURINTO SERI**.

Composition

The core characters are based on Libertinus Sans, which was derived from Biolinum created by Philipp H. Poll.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

x1ftRégòÊ1&
永ओعشЖボ

A design that accommodates diverse writing systems in a “faux-monospaced” typeface. Each character is 1×, 2×, 3×, ... the width of the Latin letters. This approach is sometimes called “duospaced” or “multi-spaced”.

The body text in this section is typeset in Kurinto Mono.

Widths

This typeface includes three widths: Kurinto Mono Narrow, Kurinto Mono (regular width), and Kurinto Mono SemiWide.

Composition

The core characters are derived from on Noto Sans Mono, designed by The Monotype Corporation for Google. Many of the writing systems in the Aux font variant are derived from FairfaxHD by Rebecca G. Bettencourt. For a complete roster of the source fonts and contributors, see the Map.txt, Scripts.txt, and Contributors.txt files in the release package.

Characteristics

The default figures - 0123456789 - are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height ($\text{Em} \times 1\frac{1}{8}$) and have a width of 60% of an Em.

OpenType Features

Kurinto Mono implements an extensive set of OpenType features spanning the many writing systems supported by the Core, Main, and Aux font variants. These features are largely inherited from the source fonts that make up Kurinto Mono.

The OpenType Features described in [Alternate Characters and Layout Features](#) as well as the [ACF block](#) are implemented, with these exceptions:

- The 'smcp' and 'c2sc' language features use the original small-cap characters of the Noto Sans Mono source font. If you want the Small-Cap characters from the ACF block, use 'ss11' (Stylistic Set 11) and 'ss12' for 'smcp' and 'c2sc' respectively. These Stylistic Set language features are available from Microsoft Word.
- The proportional figures in the ACF block (available via code points and through OpenType language features) are implemented. The proportional "1" uses $\frac{2}{3}$ the width of normal "1".

Secondary Typefaces

The secondary typefaces support and complement the primary typefaces. They are more specialized in purpose and provide more limited coverage of Unicode characters than the primary typefaces.

Type face	Width (blank = Normal)	Core	(Main)	Aux	Music 2,791 addl. music chars + Main	UFI Medieval Unicode Font Initiative + Cyrillic + Runic
Arte		R B / BI	R B / BI	R B / BI		
Curv		Lt R SB B Ltl / SBI BI				
News		R B / BI				
Olde	Narrow	R B / BI				
		R B / BI				
	Wide	R B / BI				
Plot		R B / BI	R B / BI			
Roma		R B / BI				R B / BI
Type	Narrow		R B / BI			
		R B / BI	R B / BI			
	Semi-Wide		R B / BI			

R=Regular, **B=Bold**, *I=Italic*, **BI=Bold Italic**,
Lt=Light, *Ltl=Light Italic*, SB=Semi-Bold, *SBI=Semi-Bold Italic*

Blue = Kurinto Full Distrib. Only	White = Possible in future releases	Grey = No plans for implementation
--------------------------------------	--	--

Type face	Width (blank = Normal)	HK	JP	KM	KR	SC	TB	TC	CJK
		Hong Kong + Core	Japanese + Core	Khmer & Myanmar + Core	Korean + Core	Simp. Chinese + Core	Tibetan + Core	Trad. Chinese + Core	Rare Kanji + Core
Arte		R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI	R B BI
Curv									
News									
Olde	Narrow								
	Wide								
Plot			R B BI						
Roma									
Type	Narrow								
			R B BI						
	Semi- Wide								
		Blue = Kurinto Full Distrib. Only		R=Regular, B=Bold, I=Italic, BI=Bold Italic					
		Gothic style CJK glyphs		Mincho style CJK glyphs		Hanazono style CJK glyphs			
				Serif style glyphs					

xlftR egòÊ 1 &
永 ओ ش Ɔ Ж 𐐃

An alternative book typeface with a particular flair, based on a Modified Venetian design.

This section is typeset in **KURINTO ARTE**.

Composition

This typeface is based on Kelvinch, developed by Paul James Miller as an offshoot to Victor Gauntly's Gentium Book effort.

Characteristics

The default figures – 0123456789 – are Tabular / Hybrid.

Some of the distinctive elements are the sloped bar of the 'e' and the diamond dots for the 'i', 'j', and period. The “*” has a counter-rotation reminiscent of a fan.

xlftRegoE1&φЖ

A relaxed Humanist design suitable for informal body text or headings.

This design is based on a typeface that I have used extensively for titles and heading in more casual documents. This is a sans-serif typeface with gentle hourglass curves on many of the strokes, giving the font a lively and slightly relaxed feel. **KURINTO CURV** is implemented using my own re-drawing of the letterforms.

This section is set in Kurinto Curv Core. Also, headings throughout this document are set in **KURINTO CURV CORE BOLDITALIC**.

Composition

One distinctive aspect of this typeface is that the Bold-Italic style uses an 8° slant angle – less than the typical 10°–12° slant angle of most typefaces. This make Kurinto Curv BoldItalic useful for continuous text (especially in headings) rather than just as an emphasized, stand-out phrase. This paragraph is an example of the BoldItalic style.

This font is also available in a Light style (*including italics*) and also as a ...

SemiBold style (*including italics*). However, caution is needed in Microsoft Word: If you click the [B] button while using the Light or SemiBold styles, Word will perform its dreaded an auto-bold feature. Better to intentionally choose the style you want from the font pulldown list.

The lowercase “x” is distinctive for its curve on the backstroke.

This class of fonts are sometimes described as “hybrid sans-serif”, “humanist stressed sans-serif”, “flare serif”, or “glyphic serif”. Other typefaces in this class include Albertus, Carter Sans (https://en.wikipedia.org/wiki/Carter_Sans).

Note that only the **CORE** variant of **KURINTO CURV** is currently implemented. This typeface does not have the full Unicode support that some of the other Kurinto fonts provide.

Characteristics

The default figures – 0123456789 – are Proportional / Lining.

OpenType Features

All OpenType Features described in [Alternate Characters and Layout Features](#) are implemented. In particular:

- Stylistic Set 11 converts LOWER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 12 converts UPPER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 13 converts LOWER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 14 converts UPPER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 15 renders UPPER-CASE LETTERS IN TITLING CAPS.
- Stylistic Set 16 renders UPPER-CASE LETTERS WITH EXTRA SPACING.
- The style of figures can be selected using the Number Spacing, Number Forms, and Stylistic Sets menus, as shown in the Specimen Book.

The [ACF block](#) is fully implemented for this font, with all code points directly accessible.

x1ftRegoE1&φЖ

A contemporary design based on the newsprint tradition and, in particular, **TIMES NEW ROMAN**.

About Times New Roman

This section provides background on Times New Roman, which relates to **KURINTO NEWS**, **KURINTO TMod**, and **KURINTO TRom**.

This text is from the **TIMES NEW ROMAN** v 3.00 TrueType font file (last modified February 6, 2009):

This remarkable typeface first appeared in 1932 in The Times of London newspaper, for which it was designed. It has subsequently become one of the world's most successful type creations. The original drawings were made under Stanley Morison's direction by Victor Lardent at The Times. It then went through an extensive iterative process involving further work in Monotype's Type Drawing Office. Based on experiments Morison had conducted using Perpetua and Plantin, it has many old-style characteristics but was adapted to give excellent legibility coupled with good economy. Widely used in books and magazines, for reports, office documents and also for display and advertising.

Additional information gleaned from the [Wikipedia page for Times New Roman](#):

Stanley Morison had proposed an older Monotype typeface, **PLANTIN**, as a basis for the design, and **TIMES NEW ROMAN** mostly matches the dimensions of **PLANTIN**. The main change was that the contrast between strokes was enhanced to give a crisper image. As a typeface designed for newspaper printing, **TIMES NEW ROMAN** has a high x-height, short descenders to allow tight linespacing and a relatively condensed appearance.

The term “Roman” in the name of the font is a reference to the regular or roman style (sometimes also called Antiqua) for the Latin letters. Roman type has roots in Italian printing of the late 15th and early 16th centuries, but the **TIMES NEW ROMAN** design has no connection to Rome or to the Romans.

Composition

The core character set is based on **STIX2**, a set of fonts released by Ross Mills and John Hudson of Tiro Typeworks for the Scientific and Technical Information Exchange (STIX) consortium to serve the scientific and engineering community. See <https://www.stixfonts.org/> for more information. See also the **KURINTO TRom** typeface below, which contains the history of the **STIX1** fonts.

Characteristics

The default figures – 0¹2¹3¹4¹5¹6¹7¹8¹9¹ – are Tabular / Lining.

Note that, while **KURINTO NEWS** is designed in the style of **TIMES NEW ROMAN**, the letters are generally wider than the original font. A document set in **TIMES NEW ROMAN** will expand if you switch to **KURINTO NEWS**. For fonts that avoid this issue, see **KURINTO TROM** and **KURINTO TMod** below.

The Box Drawing and Block Elements characters take the full line height and are half as wide as the line height (or $\frac{1}{16}$ of an Em).

Styles

The Bold version of this font uses the bold version of Stix2.

*The Italics and **Bold Italics** versions of this font use the oblique italics style.*

OpenType Features

This font does not implement the [ACF block](#), in favor of the extensive OpenType implementation inherited from Stix2.

The OpenType features include layout directives for Cyrillic, Greek, and Latin (general and localized for Romanian and Turkish).

x1ftKego € 1 & q x

A “blackletter” design that emulates the masthead of The New York Times.

Composition

It is based on Chomsky, a font developed by Fredrick Brennan. For background on blackletter fonts in general, see <https://en.wikipedia.org/wiki/Blackletter>.

Because this is a special-purpose font, the character set is limited to the Core character set: KURINTO OLDE CORE, KURINTO OLDE CORE BOLD, KURINTO OLDE CORE ITALIC, and KURINTO OLD CORE BOLD ITALIC.

Some may ask why typeface design that is at the fringes of contemporary usage is in Kurinto. Well, I became fascinated with the transition from Fraktur to Antiqua style. It’s an Orwellian saga played out on a world stage during the Second World War with Hitler as the driving force. The Wikipedia pages at <https://en.wikipedia.org/wiki/Fraktur> and https://en.wikipedia.org/wiki/Antiqua-Fraktur_dispute are good starting points.

The figures (digits) are Proportional Lining, with no alternative forms currently available.

Characteristics

The default figures – 0123456789 – are Proportional / Lining.

OpenType Features

All OpenType Features described in [Alternate Characters and Layout Features](#) are implemented. In particular:

- Stylistic Set 11 converts LOWER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 12 converts UPPER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 13 converts LOWER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 14 converts UPPER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 15 renders UPPER-CASE LETTERS IN TITLING CAPS.
- Stylistic Set 16 renders UPPER-CASE LETTERS WITH EXTRA SPACING.
- The style of figures can be selected using the Number Spacing, Number Forms, and Stylistic Sets menus, as shown in the Specimen Book.

The [ACF block](#) is fully implemented for this font, with all code points directly accessible.

xlftRegoE1&φЖ

An ultra-thin design that emulates the trace of a plotter pen.

This section is set in KURINTO PLOT BOLD, except for the next paragraph ...

This paragraph is typeset in KURINTO PLOT. I intentionally set the text above in the Bold version of this font, in case it was just too thin to see.

Composition

This typeface is based on the “Hair” and “UltraNarrow” styles of FIRA SANS, developed by Erik Spiekermann, Ralph du Carrois, Anja Meiners, and Botio Nikoltchev of Carrois Type Design.

Characteristics

The default figures – 0123456789 – are Proportional / Lining.

xlftRegoE1&φЖ

An old-style Garalde typeface based on the Clarendon designs of the 17th century.

The style of Roma is based on the Clarendon, designed in the 17th century and used at the Oxford University Press (aka Clarendon Press). It represents a transition from the earlier 16th century typefaces, represented by **KURINTO BOOK**, and later designs, such as the transitional serif designs of the early 18th century represented by **KURINTO GRGA**.

This section is set in **KURINTO ROMA**.

Composition

The core characters are based on JUNICODE, by Peter S. Baker. See <https://en.wikipedia.org/wiki/Junicode> for a description.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

xlfRegE1&φЖ

A “faux-monospace” typeface design that evokes a typewritten feel while avoiding the aesthetic and readability issues inherent in true monospaced fonts.

Many graphic designers work with the aesthetic of monospaced typefaces (such as this paragraph and the next) to create a nostalgic association with typewriters, computer terminals, and programming. Monospaced (fixed-width) typefaces appear in many settings, even though the technical restrictions that caused the use of a fixed-width font no longer apply.

True monospaced fonts, such as Kurinto CMod and Kurinto CNew, have some percentage of their characters “squashed” or “stretched” horizontally to meet the fixed-width requirement. The “m” and “i” are typical examples, but “&” and “.” are even more severe. The most frequent offender is the space character itself. The resulting optical imbalances make the text significantly more tiresome to read after the first few sentences. This paragraph and the one above are examples of the “squashed and stretched” syndrome, since they are set in Kurinto CNew. This document now switches typefaces back to ...

Kurinto Type is a version of Kurinto CNew that uses proportionally spaced characters but retains the look of typewritten text. Graphic designers get the look that they want, and readers are spared the pain of reading a true monospaced font.

I have heard this class of fonts called “faux-monospace”, “mono-faked”, and “fauxno-spaced”.

This section (except for the second and third paragraphs) is set in **KURINTO TYPE**.

Composition

The glyphs for the **CORE** variant are based on **KURINTO CNEW**. The (greatly) expanded character set provided in the **MAIN** variant shares the glyphs from Kurinto Sans. These characters are provided for completeness, even though the glyphs do not match the style of the core characters set.

Also note that the line height of this font matches the other primary and secondary fonts of the Kurinto font folio. This is significantly larger than Kurinto CNew, which is metrically compatible with Courier New and matches the smaller line height of that font.

Characteristics

The default figures – 0123456789 – are Proportional / Lining. The spacing of these digits is significantly tighter than in **KURINTO CNEW**. However, the variation in width between the digits is relatively small since these characters are inherited from fixed-width glyphs.

OpenType Features

All OpenType Features described in [Alternate Characters and Layout Features](#) are implemented. In particular:

- Stylistic Set 11 converts LOWER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 12 converts UPPER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 13 converts LOWER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 14 converts UPPER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 15 renders UPPER-CASE LETTERS IN TITLING CAPS.
- Stylistic Set 16 renders UPPER-CASE LETTERS WITH EXTRA SPACING.
- The style of figures can be selected using the Number Spacing, Number Forms, and Stylistic Sets menus, as shown in the Specimen Book.

The [ACF block](#) is fully implemented for this font, including the Proportional Figures in the range U+10FE00..U+10FE39.

Metric Compatible Typefaces

Type face	Width (blank = Normal)	Core	(Main)	Aux	Music 2,791 addl. music chars + Main	UFI Medieval Unicode Font Initiative + Cyrillic + Runic
Aria		R B / BI	R B / BI	R B / BI		
Bria		R B / BI				
Cali		R B / BI	R B / BI	R B / BI		
CMod		R B / BI				
CNew		R B / BI				
Gara		R B / BI				
Grga		R B / BI				
TMod		R B / BI	R B / BI	R B / BI		
TRom		R B / BI				

R=Regular, B=Bold, I=Italic, BI=Bold Italic

Blue = Kurinto Full Distrib. Only	White = Possible in future releases	Grey = No plans for implementation
--------------------------------------	--	--

Type face	Width (blank = Normal)	HK	JP	KM	KR	SC	TB	TC	CJK
		Hong Kong + Core	Japanese + Core	Khmer & Myanmar + Core	Korean + Core	Simp. Chinese + Core	Tibetan + Core	Trad. Chinese + Core	Rare Kanji + Core
Aria		R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>
Bria									
Cali		R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>
CMod									
CNew									
Gara									
Grga									
TMod		R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>	R B <i>I BI</i>
TRom									
Blue = Kurinto Full Distrib. Only		R=Regular, B=Bold , <i>I=Italic</i> , BI=Bold Italic							
Gothic style CJK glyphs		Mincho style CJK glyphs		Hanazono style CJK glyphs					
		Sans serif style glyphs		Serif style glyphs					

xlftRegoE1&φЖ

A metric-compatible replacement for **ARIAL**, the default sans-serif font for contemporary Windows operating systems.

This section is set in **KURINTO ARIA**.

Composition

The core characters are primarily sourced from **ARIMO**, originally developed by Steve Matteson at Ascender and licensed by Google for the Chrome operating system as part of the Croscore set of fonts.

A full set of font variants is implemented for **KURINTO ARIA** by augmenting the Core Character Set with the same glyphs as **KURINTO SANS**. However, in order to maintain metric compatibility with **ARIAL**, the line height of **KURINTO ARIA** is substantially smaller than **KURINTO SANS**. This mean that characters from some of the writing systems will exceed the vertical bounds of this font, and may be clipped or cause collisions.

Metrics Model

The metrics model for **KURINTO ARIA** is **ARIAL** v7.00 by The Monotype Corporation (based on the original design by Robin Nicholas and Patricia Saunders in 1982), last modified January 17, 2018 and broadly distributed with Microsoft operating systems.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

x1ftRegoE1&φЖ

A metric-compatible replacement for Cambria.

This section is typeset in **KURINTO BRIA**.

Composition

The core character set is based on **CALADEA**, designed by Carolina Giovagnoli and Andres Torresi and released in 2013 under the Apache 2.0 open-source license. It is available under the Croscore fonts distributed by Google.

Metrics Model

The metrics model for **KURINTO BRIA** is **CAMBRIA** v6.99 by Monotype Imaging and Tiro Typeworks, last modified November 14, 2017 and broadly distributed with Microsoft operating systems.

Microsoft's description of the font is:

Cambria has been designed for on-screen reading and to look good when printed at small sizes. It has very even spacing and proportions. Diagonal and vertical hairlines and serifs are relatively strong, while horizontal serifs are small and intend to emphasize stroke endings rather than stand out themselves. This principle is most noticeable in the italics where the lowercase characters are subdued in style to be at their best as elements of word-images. When Cambria is used for captions at sizes over 20 point, the inter-character spacing should be slightly reduced for best results. The design isn't just intended for business documents: The regular weight has been extended with a large set of math and science symbols. The Greek and Cyrillic has been designed under close supervision of an international team of experts, who aimed to set a historical new standard in multi-script type design.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

Note that the Box Drawing and Block Elements characters of this font do not have a consistent width. They vary and cannot be reliably combined to make multi-character images.

The line height is based on the metrics model font and is noticeably compact.

Styles

The Bold version of this font is based on the Caladea source fonts.

*The Italics and **Bold Italics** versions of this font use the cursive style.*

xlftRegoE1&φЖ

A metric-compatible replacement for Calibri, a sans-serif font that replaced Arial as the default font in many Microsoft Office application in 2007.

This section is typeset in **KURINTO CALI**.

Composition

The core character set is based on **CARLITO**, designed by designed by Lukasz Dzedzic of the tyPoland foundry and adopted by Google for ChromeOS as a font-metric compatible replacement for **CALIBRI**.

Metrics Model

The metrics model for **KURINTO CALI** is **CALIBRI** v6.23 by Microsoft Corporation (designer Luc(as) de Groot), last modified June 15, 2018 and broadly distributed with Microsoft operating systems.

Microsoft's description of the font is:

Calibri is a modern sans serif family with subtle roundings on stems and corners. It features real italics, small caps, and multiple numeral sets. Its proportions allow high impact in tightly set lines of big and small text alike. Calibri's many curves and the new rasterizer team up in bigger sizes to reveal a warm and soft character.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height and are ½ of an Em (i.e. 1 En) wide.

The line height is based on the metrics model font and is noticeably compact.

Styles

The Regular and Bold versions of this font have almost no stroke variations, resulting in a rather “plain” look that probably adds to its readability.

*The Italics and **Bold Italics** versions of this font use a stylized cursive style.*

OpenType Features

This font does not implement the [ACF block](#), in favor of the extensive OpenType implementation inherited from **CARLITO**.

The OpenType features for the Core variant include layout directives for Cyrillic (general and localized for Serbian), Greek, and Latin (general and localized for Romanian and Phonetic transcription IPA).

x1fRgE1&φЖ

A metric-compatible replacement for **COURIER NEW** with a modern design.

This section is typeset in **KURINTO CMod**.

Composition

The core characters are inherited from **COUSINE** developed by Steve Matteson.

Metrics Model

The metrics model for **KURINTO CMod** is **COURIER NEW** v6.92 by The Monotype Corporation (designer Howard Kettler), last modified November 13, 2017 and broadly distributed with Microsoft operating systems.



A metric-compatible replacement for Courier New. This typeface preserves the typographic style of the original - with its typewritten, slab-serif design.

This section is set in **KURINTO CNEW**.

Composition

The glyphs are based on (and substantially expanded from) Courier Prime designed by Alan Dague-Greene.

Note that only the **CORE** variant of **KURINTO CNEW** is currently implemented. This typeface does not have the full Unicode support that some of the other Kurinto fonts provide.

Metrics Model

The metrics model for **KURINTO CNEW** is **COURIER NEW** v6.92 by The Monotype Corporation (designer Howard Kettler), last modified November 13, 2017 and broadly distributed with Microsoft operating systems.

Kurinto CNew is notably different from Courier New:

- The Regular style of Kurinto CNew uses a substantially heavier weight than Courier New. This is a style choice in the interest of readability and also to balance the substantial whitespace that often accompanies how Courier is used (screenplays, for example).
- For comparison, this paragraph is set in Courier New, with significantly lighter weight to the strokes.
- The italics of Kurinto CNew are modeled on the “casual” script of vintage typewriters.
- OpenType features including Small Caps, Petite Caps, and Titling Caps, as well as all eight styles of figures have been implemented. See the OpenType Features section below for details.

Characteristics

Since this font is a true monotype font, every character in the font has the same width. This includes:

- characters that would normally render as “zero-width” characters, such as U+2060 Word Joiner, which renders in this font as a space,

- very narrow characters, such as “!”,
- very wide characters, such as the U+2015 Horizontal Bar: “—”, which renders in this font in the same width as U+002D Hyphen-Minus: “-”.

The default figures - 0123456789 - are Tabular / Lining.

OpenType Features

All OpenType Features described in [Alternate Characters and Layout Features](#) are implemented, including the features dealing with proportional figures. Even though this is a fixed-width font, proportional glyphs can be used as part of OpenType features.

In particular:

- Stylistic Set 11 converts LOWER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 12 converts UPPER-CASE LETTERS TO SMALL CAPS.
- Stylistic Set 13 converts LOWER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 14 converts UPPER-CASE LETTERS TO PETITE CAPS.
- Stylistic Set 15 renders UPPER-CASE LETTERS IN TITLING CAPS.
- Stylistic Set 16 renders UPPER-CASE LETTERS WITH EXTRA SPACING.
- The style of figures can be selected using the Number Spacing, Number Forms, and Stylistic Sets menus, as shown in the Specimen Book.

The [ACF block](#) is fully implemented for this font, including the Proportional Figures in the range U+10FE00..U+10FE39.

xlftR ego E 1 & φ Ж

A metric-compatible replacement for Monotype Garamond™, the predominant implementation among the many commercial fonts based on Claude Garamont's Antiqua typeface Garamond.

This section is set in **KURINTO GARA**.

History

From the [Microsoft Typograph web site](#):

Monotype Drawing Office 1922. This typeface is based on roman types cut by Jean Jannon in 1615. Jannon followed the designs of Claude Garamond which had been cut in the previous century. Garamond's types were, in turn, based on those used by Aldus Manutius in 1495 and cut by Francesco Griffo. The italic is based on types cut in France circa 1557 by Robert Granjon. Garamond is a beautiful typeface with an air of informality which looks good in a wide range of applications. It works particularly well in books and lengthy text settings.

Composition

The core character set is based on **EB GARAMOND**³⁰ (»Egenolff-Berner-Garamond«), released by Georg Duffner in 2011 under the Open Font License. Duffner based the design on a specimen printed by Egenolff-Berner in 1592, with italic and Greek characters based on Robert Granjon's work, as well as the addition of Cyrillic characters and OpenType features such as swash italic capitals and schoolbook alternates. Its name is an acronym for **E**genolff-**B**erner-**G**aramond which refers to the fact that the letter forms are taken from the Egenolff-Berner specimen. As Georg Duffner could not complete the bold weights for personal reasons, the project was continued by Octavio Pardo.

It has been claimed that **GARAMOND** uses much less ink than **TIMES NEW ROMAN** at a similar point size, so changing to **GARAMOND** could be a cost-saver for large organizations that print large numbers of documents, especially if using inkjet printers. **GARAMOND**, along with **TIMES NEW ROMAN** and **CENTURY GOTHIC**, has been identified by the GSA as a “toner-efficient” font.

³⁰ The text of this section is largely sourced from <https://en.wikipedia.org/wiki/Garamond> and https://en.wikipedia.org/wiki/EB_Garamond.

Metrics Model

The metrics model for **KURINTO GARA** is **GARAMOND** v2.40 by The Monotype Corporation (originally designed in 1922), last modified April 2, 2004. The description of the font by Monotype is:

This typeface is based on roman types cut by Jean Jannon in 1615. Jannon followed the designs of Claude Garamond which had been cut in the previous century. Garamond's types were, in turn, based on those used by Aldus Manutius in 1495 and cut by Francesco Griffo. The italic is based on types cut in France circa 1557 by Robert Granjon. Garamond is a beautiful typeface with an air of informality which looks good in a wide range of applications. It works particularly well in books and lengthy text settings.

Note that the angle of the Italic styles of this font is quite pronounced – you might even say extreme. The source font design uses an angle of 17.2°.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height and share the width of the capital “U” (about 71% of an Em).

Styles

The Bold version of this font uses the EB Garamond extensions by Octavio Pardo.

*The Italics and **Bold Italics** versions of this font use the cursive style. However ... note that there are, as of this version of Kurinto, significant issues with the letter-spacing of these styles!*

OpenType Features

This font does not implement the [ACF block](#), in favor of the extensive OpenType implementation inherited from **EB GARAMOND**.

The OpenType features include layout directives for Cyrillic (general and localized for Macedonian and Serbian) and Latin (general and localized for Azerbaijani, Crimean Tatar, German, Latin, and Turkish).

xlftRegoE1&φЖ

A metric-compatible replacement for Georgia, a “transitional serif” typeface designed in 1993 by Matthew Carter for Microsoft Corporation. Carter based his design on Scotch Roman, an early 19th century typeface that Carter felt would appear elegant but legible when printed in small sizes or rendered on low-resolution screens.

This section is typeset in KURINTO GRGA.

Composition

The core character set is based on GELASIO, designed by Eben Sorkin and Viviana Monsalve and released under the OFL. The current development of that font is accessible in an [open-source repository on GitHub](#).

Metrics Model

The metrics model for KURINTO GRGA is GEORGIA v5.592 by Carter & Cone (designer Matthew Carter), last modified October 17, 2017 and broadly distributed with Microsoft operating systems.

Characteristics

The default figures – 0123456789 – are Proportional / Hybrid – the only typeface in the Kurinto Folio with Hybrid / Proportional default figures.

The Box Drawing and Block Elements characters take the full line height and are a shade narrower than the width of the capital “X” (about 71% of an Em).

The line height is based on the metrics model font and is noticeably compact.

Styles

The Bold version of this font uses fairly dramatic thick/thin stroke variations to get a very “punchy” boldness.

The Italics and Bold Italics versions of this font use the cursive style.

OpenType Features

This font does not implement the [ACF block](#), in favor of the extensive OpenType implementation inherited from GELASIO.

The OpenType features include layout directives for Cyrillic, Greek, and Latin (general and localized for Azerbaijani, Catalan, Crimean Tatar, Kazakh, Moldavian, Dutch, Romanian, Tatar, and Turkish).



A modern, metric-compatible replacement for Times New Roman. See [About Times New Roman](#) for the background on this design.

This section is typeset in **KURINTO TMOD**.

Composition

The core character set is based on **TINOS**, designed by Steve Matteson and released under the Apache 2.0 open-source license in 2013 as part of Google’s Croscore fonts package. **TINOS** is a derivative of the widely used **LIBERATION SERIF** font design.

Aside from the metric compatibility, **TINOS** does not particularly resemble **TIMES NEW ROMAN**, being much squarer in shape with less fine detail and blunt ends rather than ball terminals.

Metrics Model

The metrics model for **KURINTO TMOD** (and **KURINTO TROM**) is **TIMES NEW ROMAN** v7.00 by The Monotype Corporation (based on the original design by Stanley Morison and Victor Lardent in 1932), last modified January 17, 2018 and broadly distributed with Microsoft operating systems.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height and are half as wide as the line height (or $\frac{1}{16}$ of an Em).

The line height is based on the metrics model font and is noticeably compact.

Styles

The Bold version of this font uses fairly dramatic thick/thin stroke variations to get a very “punchy” boldness.

*The Italics and **Bold Italics** versions of this font use an intermediate style between oblique and cursive. Kurinto’s italics versions are derived from the Tinos fonts (i.e. not automatically generated).*

OpenType Features

This font does not currently implement the [ACF block](#), or its related OpenType layout features.

x1ftRegoE1&φЖ

A traditional, metric-compatible replacement for Times New Roman. See [About Times New Roman](#) for the background on this design.

This section is set in **KURINTO TROM**.

Composition

The core character set is based on the first version of the STIX fonts, last updated in 2013. The design was developed by Coen Hoffman of Elsevier. The history of the font is described in a paragraph contained within the font file itself:

Arie de Ruiter, who in 1995 was Head of Information Technology Development at Elsevier Science, made a proposal to the STI Pub group, an informal group of publishers consisting of representatives from the American Chemical Society (ACS), American Institute of Physics (AIP), American Mathematical Society (AMS), American Physical Society (APS), Elsevier, and Institute of Electrical and Electronics Engineers (IEEE). De Ruiter encouraged the members to consider development of a series of Web fonts, which he proposed should be called the Scientific and Technical Information eXchange, or STIX, Fonts. All STI Pub member organizations enthusiastically endorsed this proposal, and the STI Pub group agreed to embark on what has become a twelve-year project. The goal of the project was to identify all alphabetic, symbolic, and other special characters used in any facet of scientific publishing and to create a set of Unicode-based fonts that would be distributed free to every scientist, student, and other interested party worldwide. The fonts would be consistent with the emerging Unicode standard, and would permit universal representation of every character. With the release of the STIX fonts, de Ruiter's vision has been realized.

Metrics Model

The metrics model for **KURINTO TROM** (and **KURINTO TMOD**) is **TIMES NEW ROMAN** v7.00 by The Monotype Corporation, last modified January 17, 2018 and broadly distributed with Microsoft operating systems.

Characteristics

The default figures – 0123456789 – are Tabular / Lining.

The Box Drawing and Block Elements characters take the full line height and are slightly narrower than the capital “A” (about 71% of an Em).

The line height is based on the metrics model font and is noticeably compact.

Styles

The Bold version of this font is derived from the Stix1 source.

*The Italics and **Bold Italics** versions of this font use an cursive italics style.*

OpenType Features


This font does not currently implement the [ACF block](#), or its related OpenType layout features.

Making Use of All the Characters

This section offers some advice on how to use all the characters at your disposal to raise type typographic level of your documents.

The suggestions in this section use only characters from the Core variant of fonts, which means that every recommended characters is available in all Kurinto fonts.

Dashes

Typing  on your keyboard will often get you U+002D - HYPHEN-MINUS. Sadly, this turns out to be one of the least useful versions of the many dashes available in Unicode. Because it was part of the Basic Latin subset, it got used for every imaginable purpose that a horizontal, straight-line character could be used. This multi-purpose nature meant that it really isn't useful for much, typographically speaking.

Here are some alternates, and examples of situations where they can improve the look and readability of your document:

- U+2011 NON-BREAKING HYPHEN - (Core) A hyphen which matches the HYPHEN-MINUS in design, but which cannot be used to break a word across lines. Use this, for example, in “E-mail”, since it is a word unit that should not be broken.
- U+2012 FIGURE DASH – (Core) Designed to be used with numerical digits. The figure dash has the width of a “0” (zero character) and is vertically centered on the zero character: from **KURINTO BOOK**: “0–0”. This tends to be higher than the EN DASH and EM DASH, which appear in **KURINTO BOOK** as “0—0” and “0—0”, respectively.
- U+2013 EN DASH – (Core) Designed as a replacement for the word “through”. For example: “Jan–Jun”. *Microsoft Word* is often configured to convert a typed HYPHEN-MINUS to an EN DASH after you complete the following word. This form of dash centers on the lower-case “x” and is often at the same vertical position as a ~~–strikethrough–~~, so it is not so useful when working with numbers. The width is exactly one En and ½ of an Em.
- U+2014 EM DASH — (Core) Used in pairs—if you so choose—to offset parenthetical text. You can optionally use spaces — especially in the U.S. — around the EM DASH. The width is exactly one Em and is vertically centered on the lower-case “x”.
- U+2015 HORIZONTAL BAR — (Core) Sometimes called a “quotation dash”, it is used to introduce quoted text in some typographic styles. In many fonts, the length is ¾ of an Em. I often use it as the leader in a byline or signature line, such as at the bottom of page 2 of this document:
— Clint Goss [clint@goss.com], as of July 25, 2020
- U+2027 HYPHENATION POINT · (Core) A visible symbol used to indicate correct positions for word breaking, as in dic·tion·ar·ies.

- U+2212 MINUS SIGN – (Main) A dedicated mathematical operator for numeric expressions: 12 – 34. Compare against a keyboard dash: 12 - 34.
- U+23AF HORIZONTAL LINE EXTENSION — (Main) Designed to be used in sequences to generate long connected horizontal lines such as these three: ———. The Unicode spec says that it can be “used for extension of Arrows”, but the vertical alignment and thickness is not consistent with the arrow characters in many fonts: —→ —→.
- U+23E4 STRAIGHTNESS LINE EXTENSION — (Main) A character in the miscellaneous technical block that “represents line straightness in a technical context”.
- U+2796 HEAVY MINUS SIGN — (Main) A dingbat that is a heavy variant of the arithmetic symbol.
- U+2E3A TWO-EM DASH —— (Main) Used in a text setting to indicate missing letters or words. The width is exactly 2 Em.
- U+2E3B THREE-EM DASH ——— (Main) Used to indicate an entire missing word. The width is exactly 3 Em. It is often used in a sequence of bibliographic entries to stand in for a repeated author:

Roger Wilco, *Songs of Warm Women, Cold Beer, and Rusty Trucks*, 1919.
 ———, *Songs of Cold Women, Warm Beer, and Dusty Ducks*, 1920.

You might also want to consider these other forms of dashes:

- U+00AD – SOFT HYPHEN - (Core)
- U+02D7 – MOD LET MINUS SIGN - (Main)
- U+0320 – COMB MINUS SIGN BELOW _ (Main)
- U+2010 – HYPHEN - (Core)
- U+2043 – HYPHEN BULLET ▪ (Main)
- U+2053 – SWING DASH ~ (Main)
- U+207B – SUPERSCRIPT MINUS ⁻ (Main)
- U+208B – SUBSCRIPT MINUS ₋ (Main)
- U+FE58 – SMALL EM DASH - (Main)
- U+FE63 – SMALL HYPHEN-MINUS - (Main)
- U+FF0D – FULLWIDTH HYPHEN-MINUS - (Main)

For more suggestions, see the *Hyphens and Dashes* page in *Butterick’s Practical Typography* at <https://practicaltypography.com/hyphens-and-dashes.html>. See also [GPO 2016] pages 205–208.

Quotation Marks

⟨⟨⟨⟨ <« ‹„ “ ‘xx’ ”, „, › » > ⟩⟩⟩⟩

The Wikipedia page on Quotation Marks at https://en.wikipedia.org/wiki/Quotation_mark has a complete roster of the choices available in Unicode and how they are used in various languages and writing systems. There is also additional information on pages such as <https://en.wikipedia.org/wiki/Guillemet>.

Here is a roster of the choices, typeset in Kurinto Text:

- "xx" – U + 0022, quotation mark, Typewriter (“programmer’s”) quote, ambidextrous. Also known as “double quote”.
- 'xx' – U + 0027, apostrophe, Typewriter (“programmer’s”) straight single quote, ambidextrous.
- «xx» – U + 00AB and U + 00BB, left/right-pointing double angle quotation mark, Double angle quote (chevron, guillemet, duck-foot quote).
- ‘xx’ – U + 2018–19, left/right single quotation mark, Single curved quote, left. Also known as inverted comma or turned comma.
- ,xx' – U + 201A–1B, single low-9/high-reversed-9 quotation mark, Low single curved quote.
- “xx” – U + 201C–1D, left/right double quotation mark, Double curved quote.
- „xx“ – U + 201E–1F, double low-9/high-reversed-9 quotation mark, Low double curved quote.
- <xx> – U + 2039–3A, single left/right-pointing angle quotation mark, Single angle quote.
- „ – U + 2E42, double low-reversed-9 quotation mark, also called double low reversed comma, quotation mark.

Quotation marks in dingbats:

- ‘xx’ – U + 275B–5C, heavy single turned comma/comma quotation mark ornament.
- “xx” – U + 275D–5E, heavy double turned comma /comma quotation mark ornament.
- “xx ” – U + 1F676–77, sans-serif heavy double turned comma/comma quotation mark ornament.
- „ – U + 1F678, sans-serif heavy low double comma quotation mark ornament.

Quotation marks in Braille Patterns:

- .: xx:. – U + 2826–34, braille pattern dots-236/356, Braille double opening/closing quotation mark.

Quotation marks in Chinese, Japanese, and Korean (CJK):

- [xx] – U + 300C–0D, left/right corner bracket.
- [xx] – U + 300E–0F, left white corner bracket.
- ``xx`` – U + 301D–1E, (reversed) double prime quotation mark.

- „ – U + 301F, low double prime quotation mark.

Alternate encodings:

- ┌ – U + FE41, presentation form for vertical left corner bracket.
xx
- ┐ – U + FE42, presentation form for vertical right corner bracket.
- ┌ – U + FE43, presentation form for vertical left white corner bracket.
xx
- ┐ – U + FE44, presentation form for vertical right white corner bracket
- " xx " – U + FF02, fullwidth quotation mark.
- ' xx ' – U + FF07, fullwidth apostrophe
- [xx] – U + FF62–63, halfwidth left/right corner bracket.

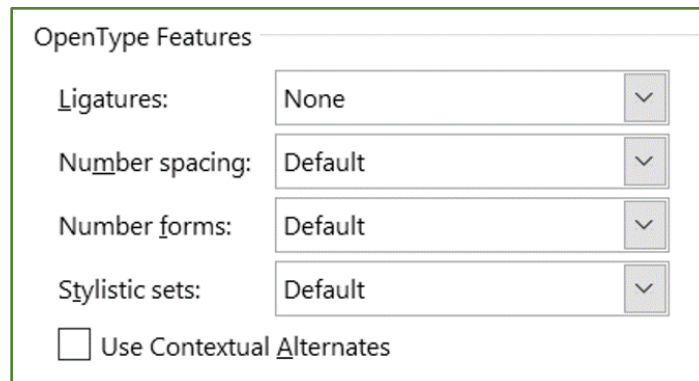
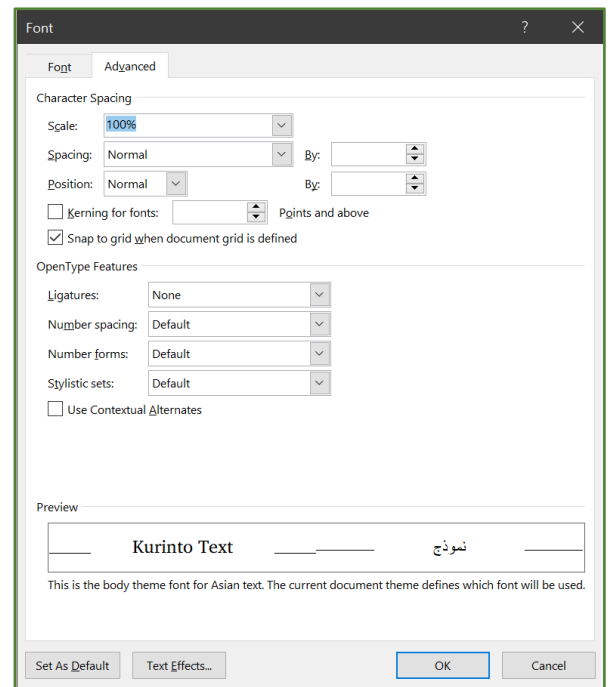
Alternate Characters and Layout Features

Kurinto fonts provide a wide array of features that let you control the look and layout of characters in your document. Some of those features are directly accessible through *Microsoft Word*, as described in this section.³¹ See [OpenType Layout Features](#) if you are looking for more technical information.

The primary way to access alternate characters and layout features in *Microsoft Word* is through the Font dialog box accessible from: Home ►► Font ►► Advanced. This technique works on selected text, insertion points, and document styles. The Font / Advanced dialog box is shown on the right.

The Character Spacing section affects the size and spacing of characters.

The OpenType Features section of the Font / Advanced dialog box gives you access to alternate characters and layout features:



Ligatures

A typographic ligature is a single character than replaces two or more characters, typically by joining them together. The “æ” character is a joining of “a” and “e”. You can obtain ligatures in two ways:

- **Unicode ligatures** are accessed using Unicode code points for specific ligature characters. For example, the “æ” is U+00E6. The *Specimen Book* document (Kurinto_SpecimenBook.pdf) shows each ligature character in Unicode and displays how they look in each of the Kurinto typefaces. However, the set of Phonetic ligatures are intended for use with the International Phonetic

³¹ The OpenType layout features described in this section are not currently available in other *Microsoft Office* applications such as *Excel* and *PowerPoint*.

Alphabet (IPA). While they are technically available for any use, the Phonetic ligatures and should **not** be used for general text.³² For more information on Unicode ligatures, see https://en.wikipedia.org/wiki/Orthographic_ligature.

- **OpenType ligatures** are accessed by selecting one of the choices in the Ligatures option box.

The choices for the Ligatures option list in Microsoft Word are:

- **None.** Setting Ligatures to **None** turns off the Standard, Contextual, Historical, and Discretionary ligatures. However, many other ligatures remain active – including required ligatures that are basic to many writing systems (Arabic, for example).
- **Standard Only** turns on core ligatures in the OpenType [liga] set.
- **Standard and Contextual** turns on the OpenType [liga] and [clig] sets.
- **Historical and Discretionary** turns on the OpenType [hlig] and [dlig] sets.
- **All** turns on all the available OpenType ligatures above.

Here are the OpenType ligatures available in Kurinto Book for the Latin writing system:

Standard [liga]

ffi ⇒ ffi ffl ⇒ ffl ff ⇒ ff fi ⇒ fi fj ⇒ fj fl ⇒ fl

Historical [hlig]

ffi ⇒ ffi ffl ⇒ ffl fh ⇒ fh fi ⇒ fi fl ⇒ fl ft ⇒ ft
NT ⇒ NT VI ⇒ V

Discretionary [dlig]

Th ⇒ Th ct ⇒ ct st ⇒ st

Other ligatures are available for different writing systems. For example, enabling Discretionary Ligatures in Hebrew [hebr] will enable the transformation לx ⇒ ל in Kurinto Book.

The Specimen Book document (Kurinto_SpecimenBook.pdf) displays all the ligatures that can be controlled by the OpenType Features dialog box of Microsoft Word for each Kurinto typeface.

Alt Caps and Figures

The remainder of this section describes Alt Caps and Figures, which let you change the look of capital letters in the Latin, Greek, and Cyrillic writing systems. They also let you select the style for figures (digits, numerals).

³² See Steven Moran and Michael Cysouw, *The Unicode Cookbook for Linguists: Managing Writing Systems Using Orthography Profiles*, Language Science Press, 2018, page 61.

This section describes specifically how to access those alternate character forms in *Microsoft Word*. However, you can also access them using:

- The OpenType layout format tag – such as [smcp] for Small Caps. Access to OpenType layout format tags is provided by some recent authoring tools. See [OpenType Layout Features](#) for more information.
- The Unicode code points for the characters you want. You can directly access any of the Alt Caps and Figures characters using the code points mapped out in [Appendix 2](#).

Number Spacing and Number Forms

These second and third options in the OpenType Features section control how numbers (“figures”, or “digits”) are displayed.

Before looking at the options, we will look at the types of figures supported by Kurinto fonts:

Proportional figures vary in width. They are useful in text to get a balanced look for each digit. Proportional figures are also the preferred style for all-cap settings, such as headlines and titles. They are also effective anywhere that additional emphasis is desired for the figures, even in running text.

Tabular figures all share a common width. They are the preferred style for columns of numbers, such as tables, price lists, financial data, and listings.

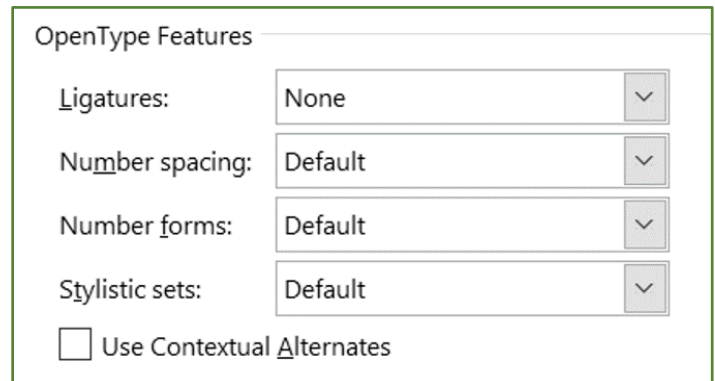
Here are examples, set in 18pt KURINTO CURV:³³

Proportional: 0[|]1[|]2[|]3[|]4[|]5[|]6[|]7[|]8[|]9[|]
Tabular: 0[|]1[|]2[|]3[|]4[|]5[|]6[|]7[|]8[|]9[|]

For example, if you are typesetting these columns of numbers, tabular is clearly the right choice:

<u>Proportional</u>	<u>Tabular</u>
1,111,111	1,111,111
9,800,000	9,800,000
7,999,111	7,999,111

³³ The markers between the digits are U+10FF23 characters that provide a reference to horizontal spacing without taking any horizontal space. These characters are available in every Kurinto font. See [Visual Font Metrics](#) for a description of these characters.



Lining figures³⁴ are approximately uniform in height to match the capital letters in the font. For example:

0123456789

Old-style figures³⁵ are used in the body text of fonts that call for a more traditional look. They vary in height and alignment, more in keeping with the look of lower-case letters. See https://en.wikipedia.org/wiki/Text_figures. For example:

0123456789

Hybrid figures are a compromise between old-style and lining figures. They retain some of the feel of the old-style figures without the (sometimes extreme) vertical alignment variations. Georgia, Merriweather, and Donegal are contemporary fonts that use hybrid figures.³⁶ For example:

0123456789

Overscore figures are provided in Kurinto to solve a problem that I encountered while trying to typeset digits with a line over them. For example: 16.6̄6̄. The overscore effect is not available in *Microsoft Word* (as far as I can tell). For example:

0123456789

Eight possibilities of figures are available in most Kurinto fonts: [**PROPORTIONAL** or **TABULAR**] × [**LINING**, **OLDSTYLE**, **HYBRID**, or **OVERSCORE**]. Here is how to get each of these combinations:

0123456789 **PROPORTIONAL–LINING**. If this is not the default format for a font, you can select Number spacing: Proportional and Number forms: Lining.

0123456789 **PROPORTIONAL–OLDSTYLE**. If this is not the default format for a font, you can select Number spacing: Proportional and Number forms: Old-style.

0123456789 **TABULAR–LINING**. Select Number spacing: Tabular and then Number forms: Lining.

0123456789 **TABULAR–OLDSTYLE**. Select Number spacing: Tabular and Number forms: Old-style.

The Hybrid and Overscore layout features are not directly available in Microsoft Word. Kurinto fonts use the Stylistic Sets feature of OpenType to access these alternate characters:

³⁴ Also known as aligning, modern, or cap figures.

³⁵ Also known as non-lining, text, ranging, hanging, billing, antique, or medieval figures.

³⁶ See [FF 2017], pages 127–128.

0123456789 PROPORTIONAL-HYBRID. Select Stylistic Sets: SS17

0123456789 PROPORTIONAL-OVERSCORE. Select Stylistic Sets: SS18

0123456789 TABULAR-HYBRID. Select Stylistic Sets: SS19

0123456789 TABULAR-OVERSCORE. Select Stylistic Sets: SS20

Each of the Kurinto typefaces has a default style of figures:

	Proportional	Tabular
Lining	Curv Olde Plot Type	Text Book Sans Seri Mono News Roma Aria Bria Cali CMod CNew Gara TMod TRom
OldStyle		
Hybrid	Grga	Arte

Stylistic Sets

In addition to the use of the four Stylistic Sets SS18 – SS20 described above, there are 6 stylistic sets that implement Alternate Capitals (“Alt Caps”). These examples are set in 16.5pt KURINTO CURV. The first line of text in each example contains mixed-case letters and the second line contains all upper-case letters:

Default: The Quick Brown Fox Jumped Over the Lazy Dog.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

SMALL CAPS (SS11):THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

PETITE CAPS (SS13): THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

C2SC (SS12): the quick brown fox jumped over the lazy dog.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

C2PC (SS14): the quick brown fox jumped over the lazy dog.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

TITLING CAPS (SS15) THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

Cap Spacing (SS16) The Quick Brown Fox Jumped Over the Lazy Dog.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

Small Caps

SMALL CAPS use regular capital letters but replace the lower-case letters with small versions of the capital letters. SMALL CAPS can be used to draw attention to the opening phrase or line of a new section of text, or to provide an additional style in a dictionary entry where many parts must be typographically differentiated.

Small Caps are accessed in Microsoft Word by selecting the Stylistic Set SS11 and in other OpenType-aware application with the layout tag [smcp].

Typesetting SMALL CAPS can be challenging. If the font does not provide alternate glyphs, *Word* will synthesize the effect by shrinking the capital letters. This approach renders poorly, since the shrunken letters appear too thin in relation to the capital letters and the rest of the text. Well-designed SMALL CAPS are not simply scaled-down versions of normal capitals; they retain the same stroke weight as other letters and have a wider aspect ratio for readability. Here is a comparison:

Default: Real Small Caps are Better

SYNTHESIZED: REAL SMALL CAPS ARE BETTER [X] SMALL-CAPS

TYPOGRAPHIC: REAL SMALL CAPS ARE BETTER SS11

The first (Default) line is set in 20pt Kurinto Curv.

The second (Synthesized) line shows the synthesized SMALL CAPS that Word produces if you check off the “Small caps” checkbox on the Font dialog box. Notice that the weight of the text is significantly lighter than the Default. Also notice that the small capitals are fairly large – this minimize the “thinning” effect of optically shrinking the characters. The result is that the lower case letters appear thin and discordant next to the capital letters.

The third (Typographic) line shows the typographic SMALL CAPS produced by selecting Stylistic Set SS11 or [smcp]. The weight of the Default text is preserved, and the height of the small capitals can be reduced more than the Synthesized text without the small capitals becoming overly thin. The result is a consistent stroke weights between upper- and lower-case letters.

Petite Caps

Petite capitals are similar to SMALL CAPS, but are smaller. For more information on both SMALL CAPS and Petite Caps, see https://en.wikipedia.org/wiki/Small_caps.

Petite Caps are accessed by selecting the Stylistic Set SS13.

Titling Caps

Titling capitals are designed for use with very large point sizes (e.g. Newspaper headlines), where the regular capitals would be too heavy. The letter shapes retain the overall dimensions of the regular capital letters, but the overall weight of the font – the thickness of the stems – is reduced. See https://en.wikipedia.org/wiki/Titling_capitals.

Titling Caps are accessed by selecting the Stylistic Set SS15.

Kerning

Because of the [Kerning pitfall](#), using kerning in your document is discouraged if you are publishing to PDF. Kurinto fonts do contain kerning directives, but kerning for body text in *Microsoft Word* is turned off by default.

If you wish to turn on kerning in *Microsoft Word*, set the kerning controls in the Font ➞ Advanced ➞ Character Spacing section.

Metric Compatibility

Metric compatible fonts match the metrics (i.e. glyph dimensions) of another font. If you replace a font with a metric-compatible alternative, the formatting of the document or web page should not change. Line and page breaks should stay in the same locations, and text in frames or boxes will not overflow their boundaries.

To address the [Reflow Pitfall](#), Kurinto contains a set of typefaces that are “metric compatible” (also called “metrically equivalent”) with commonly-used fonts.

For example, the line height and character widths of **KURINTO TROM** match **TIMES NEW ROMAN**.

See https://wiki.archlinux.org/index.php/Metric-compatible_fonts

Metric compatible fonts match the metrics (i.e. glyph dimensions) of another font (often generics such as Helvetica, Times or Courier). Due to their matching metrics, replacing a font with a metric-compatible alternative does not change the formatting of the document or a web page. Such fonts are often developed for FOSS systems to display pages correctly.

You replace your font, and everything moves. You need a metric compatible font!

Pitfalls

Here is a roster of the pitfalls I have encountered while authoring documents for publication. The Kurinto fonts are designed to address all these issues.

Font Selection

The selection of which font to use for a given character varies (widely) from application to application. The rules that a particular application uses when deciding which font to display can be complex and unpredictable. I have even had situations where a character **that IS in the specified font** is replaced by a character from a completely different font.

Vacillating Fonts in a Browser

Here is a rather drastic example of a browser repeatedly switching fonts:³⁷

A screenshot of a browser window displaying Japanese text. The text is "あなたは以下の条件に従う限り、自由に：" (You are free to follow the following conditions, within the limits of). The text is rendered in a bold sans-serif font, but the characters "に" and "に" are rendered in a different, normal-weight serif font, illustrating font switching.

The fonts selection vacillates between a bold sans-serif font and a normal-weight serif fonts.


Here is another example:³⁸

A screenshot of a browser window displaying the English text "The generated font can be used". The text is rendered in a serif font, but the characters "e" and "e" are rendered in a different, normal-weight sans-serif font, illustrating font switching.

... it almost looks like a ransom note.

Mis-matched Fonts in Microsoft Word

Here is an example of a moderately inappropriate font selection by *Microsoft Word*:

A screenshot of a Microsoft Word document showing a title in two lines. The first line is "Risto Pekka Pennanen, 'Balkan Folk Music Research and the Ottoman Legacy'" and the second line is "«Проучавање Фолклора На Балкану и Отоманско Наслеђе», Muzikologija – Musicology, Volume 8, English and Serbian, 2008, pages 127–147." The text is set in a modern sans-serif font, but the Cyrillic characters in the second line are rendered in a different font, illustrating a font mismatch.

The text is set in Humanist521, a modern, sans-serif font with clean lines. This font does not have glyphs for the Cyrillic characters of the Serbian-language version of the title. Unfortunately,

³⁷ This is a screen capture from the display of the [Creative Commons CC-BY-2.1-JP License](https://creativecommons.org/licenses/by/2.1/jp/) at <https://creativecommons.org/licenses/by/2.1/jp/> by Google Chrome Version 71.0.3578.98 x64 on February 8, 2018. The Japanese-language text “あなたは以下の条件に従う限り、自由に：” roughly translates to English as: “As long as you comply with the following terms, you are free to:”.

³⁸ The second screen capture is from the Google Translate version of the Japanese-language page at <http://mplus.font-face.jp/#/> on April 21, 2020.

Microsoft Word has chosen to substitute Book Antiqua from my collection of 2000+ fonts as the closest match. This is an old-style font with serifs and thin stroke widths, and the results are visually jarring.

In the version below, I have selected the Univers font by hand for the Serbian-language text. This font that blends better (but not ideally) with Humanist521:

Risto Pekka Pennanen, “Balkan Folk Music Research and the Ottoman Legacy”
«Проучавање Фолклора На Балкану и Отоманско Наслеђе»,
Muzikologija – Musicology, Volume 8, English and Serbian, 2008, pages 127–147.

Kurinto's Approach

The Kurinto fonts minimize issues of font selection by using a strategy that involves the Panose settings in each font. This is a technical solution that appears to work well in practice. See [Panose Settings](#) for details.

On-the-Fly Font Substitution

When you are typing text into *Microsoft Word* and enter a character that does not exist in the currently selected font, Word selects an alternative font that has the character. However, as you keep typing, ***Microsoft Word* does not automatically return to the original font.**

Here is an example, with the initial font set to **TIMES NEW ROMAN**:

The new span measures ø1.87 meters
and provides a stronger brace factor.

When I entered the ø character³⁹ on my system, Word switched fonts to **CAMBRIA MATH**, but did not switch back to **TIMES NEW ROMAN** when I carried on typing “1.87 ...”. The remainder of the sentence is set in **CAMBRIA MATH**, which is a significant change in typography.

In the version below, I have changed the rest of the sentence back to **TIMES NEW ROMAN** by hand:

The new span measures ø1.87 meters
and provides a stronger brace factor.

Kurinto's Approach

This situation should not happen with Kurinto fonts, since they provide a complete coverage of Unicode characters.

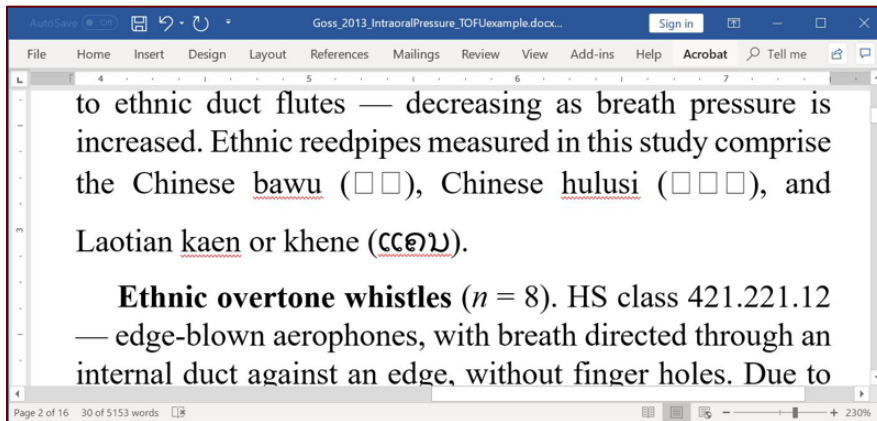
³⁹ The diameter sign “ø” can be entered in *Microsoft Word* by typing “2300” and then **Alt** + **X**.

Tofu

Tofu is the colloquial name use for the “blank” characters that appear when no appropriate glyph can be found for that character position. An example is shown on the right, photographed from an on-screen program display for an episode of Nova on PBS.



The spelling of “Hawaiʻi” uses the Unicode character U+02BB ‘ MODIFIED LETTER TURNED COMMA, which appears as the classic Tofu box because the glyph for that code point was not available.



The image to the left shows an example of a *Microsoft Word* window with a section of a research study I published in 2013.

Tofu shows up for the Chinese characters because *no* available font on my system (at the time I was writing this paper) had the required characters.

Tofu (for the Chinese characters)

Dealing with Tofu can be a big challenge:

- What language is causing the problem?
- Which writing system does that language use?
- What is the Unicode range corresponding to that writing system?
- What are my choices in fonts for that Unicode range, and what are their attributes (cost, embedding permissions, subsetting permissions, style, glyph coverage, and all the other pitfalls outlined in this section)?



There is also a more pernicious side-effect of Tofu: Every author who cares about the typographic look of their document must venture beyond the characters printed on their keycaps. Rather than “April-May”, authors learn about U+2013 – EN DASH and are rewarded with “April–May”. They might progress to using U+2212 – MINUS for “ $\beta = 17 - 42$ ” and even — if they persist — real quotation dashes.

Tofu is a huge roadblock to that learning process. Getting “April□May” can halt the exploration of typography, sending a potentially typographically-aware author scurrying back to “keycaps only”.

So — buried down in the User Guide at the tail end of one of the many pitfalls — here is another rationale for why the world needs full-featured fonts:

To support authors in their development of typesetting skills
by providing a rich, consistent, and reliable set of fonts.

Discordant Character Groups

Unicode evolved over time, adding additional characters with each version. Groups of similar characters – for example, fractions – were often allocated to different blocks in the Unicode standard. The result is that some fonts implement only a subset of a group of characters. If you use a selection of characters from a group, they may be pulled from different fonts, and the result can be visually jarring.

Here is how *Microsoft Word* displays some fractions that were set in Calibri. Three of the fractions are not present in Calibri, and *Microsoft Word* uses Tahoma for those characters. This gives us a change in style, size, and disrupts the line spacing and character spacing:

1/4 1/5 1/6 1/7 1/8 1/9 1/10

Fractions in a mix of Calibri and Tahoma.

Standby Characters

A surprising number of fonts have standby, or “placeholder”, glyphs for certain character positions. These situations can be more vexing than Tofu, since the system cannot tell that the character is effectively missing from the font.

Here is some Runic text retrieved from the web⁴⁰ that displays correctly in my browser:

Runes (ᚷᚢᚦᚱᚿ) on the Web

I cut this text from my browser and pasted into *Microsoft Word*. I used a font designed for Biblical scholars, medievalists, and linguists. It implements characters for Runic, but many of the Runic characters are actually placeholders – shown in boxes with rounded corners:

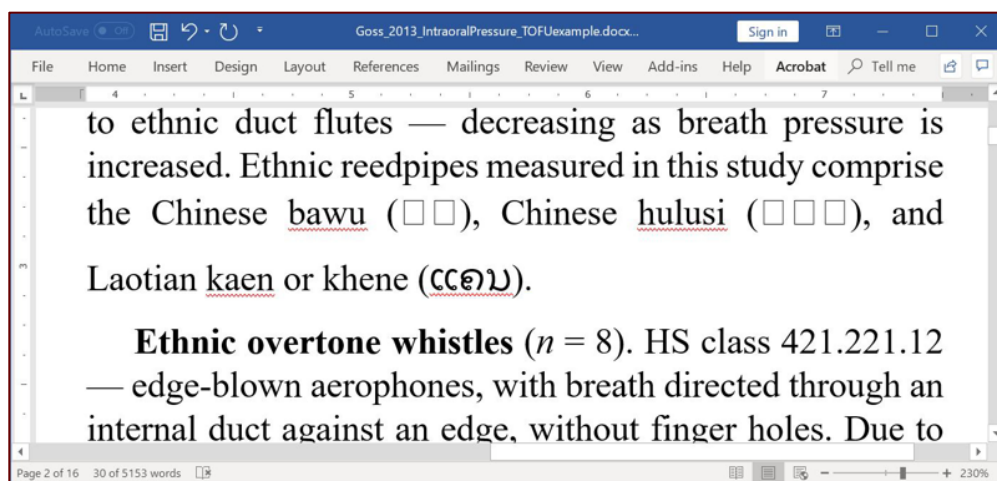
⁴⁰ Retrieved from <http://www.personal.psu.edu/ejp10/blogs/gotunicode/charts/runes.html> on August 7, 2019.

Runes (𐀀𐀁𐀂𐀃𐀄) on the Web

Surprise Layout Changes

Changing fonts (specifically or by an application's font substitution algorithm) can cause vexing changes in the layout of the text. This relates (in part) to the specified line height metrics in different fonts.

The screenshot below shows a spurious change in the line spacing caused by the Lao characters “ແຄນ”. The line spacing for the font Leelawadee UI (the font chosen by *Microsoft Word*) has a substantially higher inter-line spacing, which causes Word to insert whitespace above the line that



Spurious changes in line spacing (for the Lao characters).

contains those characters.

These issues are caused by fonts that set their vertical (line height) metrics so that each character in the font is contained within the span of the Upper and Lower Boundary Lines. This avoids issues of clipping and collisions, but can wreak havoc with line spacing in your document. For example, this paragraph has the word “Unicode” translated into Standard Arabic: **يونيكود** ... and I have typeset that word in the open-source font **AMIRI QURAN COLORED** by Kaled Hosney.⁴¹ Because some of the characters in the Quranic style of Arabic have particularly tall ascenders and descenders, Amiri has larger vertical metrics than the other fonts in this paragraph, which causes the line above to use more vertical space – which can be visually jarring.

While these types of layout changes typically occur between lines of text, they can also cause problems within a single line. Here is an example of a line of text with in-line format shifting due to changes in fonts for the extended Latin characters “ū”, “ā”, and “ī”:

⁴¹ If you are viewing the .docx version of this document, you will need to have the **AMIRI QURAN COLORED** font installed on your system. This font is included in Kurinto release packages in /Misc/AmiriQuranColored.ttf.

Abū Hayyān al-Tawhīdī

Here is the correctly formatted text, from a font that has all the characters:

Abū Hayyān al-Tawhīdī

Kurinto's Approach

Kurinto addresses these issues by using consistent line-height metrics across all fonts in the folio. The fonts also have consistent metrics for underlining, subscripts, superscripts, and other layout metrics.

Style Coverage

Most word processors expect a “core” set of the four **RBIBI** styles: Regular, **Bold**, *Italic*, and ***Bold-Italic***. If one of these style variants is not available in the font, the authoring system may try to approximate the effect, sometimes with bizarre results:

1. 1939. Marian Anderson – *Coolness America* at the Lincoln Memorial.
2. 1962. Joan Baez – *House Carpenter*.
3. 1976. Djivan Gasparian – *I Will Not Be Sold in this World* (And also a great demonstration of playing solo over drone).

Examples of Microsoft Word's attempt to create a Bold-Italic text

Kurinto's Approach

All Kurinto fonts have the four core **RBIBI** styles: Regular, **Bold**, *Italic*, and ***Bold-Italic***.

Font File Format

Word will not embed OTF (a format by Adobe) format fonts. An article by Microsoft confirms this issue. It was last updated November 4, 2019 and is located at <https://docs.microsoft.com/en-us/office/troubleshoot/office/fails-embedding-adobe-opentype-font>. Here is an excerpt (retrieved January 22, 2020):

You cannot embed an Adobe OpenType font in a document in an Office program

Symptoms

When you try to embed fonts in a Microsoft Office document, Adobe OpenType fonts that have the .otf extension are not embedded.

...

Workaround

To work around this issue, use only fonts that have the .ttf extension in documents in which you intend to embed the fonts.

Kurinto's Approach

All Kurinto files have a .ttf extension.

Embedding

When a digital document (e.g. a PDF file) is created on system **A** and viewed or printed on system **B**, the rendering of the document on those two systems should be as close as possible. One aspect of this issue relates to fonts: the same character shapes need to be available on both systems.

To accomplish this, digital documents typically embed the characters in the fonts used to create the document inside the document itself. To view or print the document on system **B**, which may not have the same fonts installed, the embedded character shapes are used to render the document as it appeared on system **A**. Documents without embedded fonts are typically rendered using the closest available font on the system where it is being viewed — often with poor results.

However, embedding font characters may violate the Copyright restriction of a font.⁴² To address this issue, TrueType files have permission flags that indicate whether embedding of the font in documents is allowed. Many fonts have these flags turned off, preventing the author from embedding the font in the PDF file. This practice promotes the commercial interests of the font vendor, but can severely hamper users.

In cases where a font is marked “Not Embeddable”, some authoring tools such as *Microsoft Word* convert text in that font to bitmapped images. This can dramatically balloon the size of the file and render poorly when printing or viewing the document in high resolution (e.g. when “zooming in”).

In my tests on October 22, 2017, I converted the primary body text in a 383-page document from a font that does allow embedding to a font that does not. The size of the resulting PDF file increased from 9 MB to 52 MB.

Kurinto's Approach

All Kurinto fonts allow embedding in digital documents.

⁴² It is even possible, in some cases, to reverse-engineer a digital document and extract the shapes of the font's characters.

Subsetting

Another permission flag in TrueType files controls whether the entire font needs to be embedded or a subset of characters can be embedded. The subset that we want is made up of the characters that are actually used in the document.

A font whose permission flags do not allow subsetting force the entire font to be included in the output PDF file. This causes the PDF file size to balloon dramatically.

Kurinto's Approach

All Kurinto fonts allow subsetting when embedding.

Kerning

Kerning is the process of fine-tuning the space between characters to improve readability and visual appeal. Here is an example of an extreme case where kerning is called for:

BRAVATA **BRAVATA**

The horizontal spacing between “AV” and all subsequent pairs of letters are tightened up to the point where the bounding box of each of those letters partially overlaps the following letter. So, the lower-right corner of the “A” is partially underneath the upper-left corner of the “V” that follows.

While kerning produces a moderate improvement in the look of the document, it can severely degrade the usability of the document. Depending on the particular application you use to view the PDF (e.g. Adobe Acrobat Reader), you may encounter these issues:

- Paragraphs with kerned text cannot be searched. This likely results from an incompatibility between the search algorithm and the directives that adjust the position of each character.
- Kerning significantly expands the size of the resulting digital document.
- Kerned text in a PDF file cannot be selected with Adobe’s PDF viewing tools.

In my tests on October 29, 2017, turning on kerning on a 400-page document and exporting it to PDF in *Microsoft Word* increased the size the PDF file from 9,674KB to 11,505KB. It also disabled the search capability of Adobe Acrobat Pro and interfered with the selection of text in the PDF file.

In some PDF viewers, bizarre issues can arise with kerned text. Here is a sample of on-screen text posted in 2017 by user mfdeakin on the Gentoo forum:⁴³

⁴³ See <https://forums.gentoo.org/viewtopic-t-1060342-start-0.html>.

About the National Science and Technology Council

The National Science and Technology Council (NSTC) is the principal means by which the Executive Branch coordinates science and technology policy across the diverse entities that make up the Federal research and development (R&D) enterprise. One of the NSTC's primary objectives is establishing clear national goals for Federal science and technology investments. The NSTC prepares R&D packages aimed at accomplishing multiple national goals. The NSTC's work is organized under five committees: Environment, Natural Resources, and Sustainability; Homeland and National Security; Science, Technology, Engineering, and Mathematics (STEM) Education; Science; and Technology. Each of these committees oversees subcommittees and working groups that are focused on different aspects of science and technology. More information is available at www.whitehouse.gov/ostp/nstc.

Kurinto's Approach

Kurinto fonts contains kerning directives from the original source fonts, but kerning for body text in *Microsoft Word* is turned off by default. If you wish to turn on kerning in *Microsoft Word* (not recommended when publishing to PDF), set the kerning controls in the Font ➞ Advanced ➞ Character Spacing section.

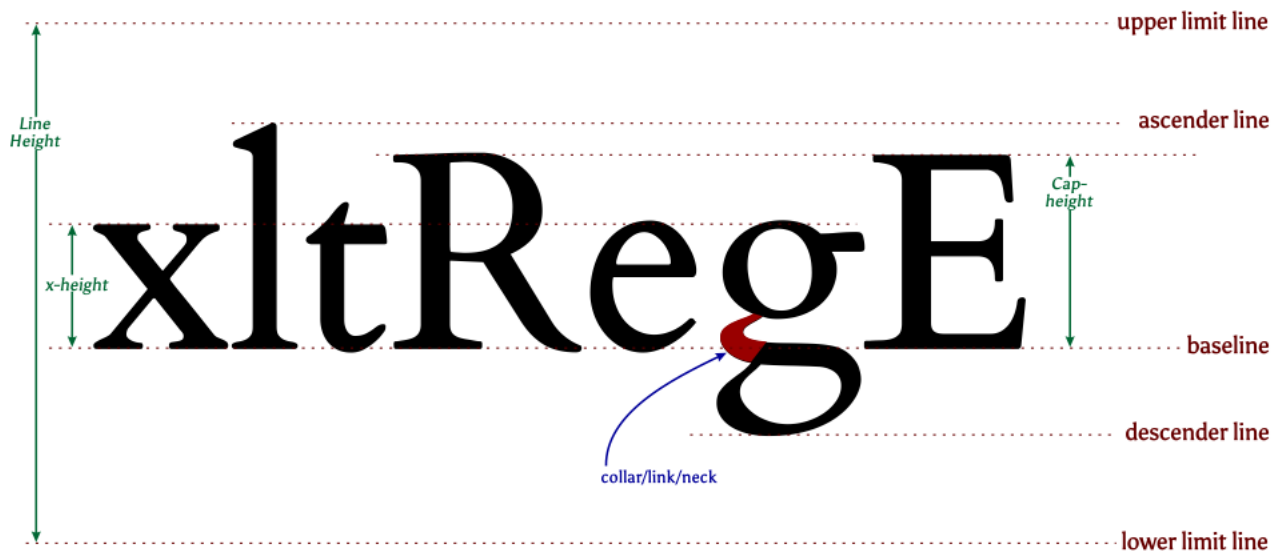
Reflow

If you change a font in your document, the dimensions of the characters (line height and individual character widths) will likely change. This causes the entire document to “reflow” ... a rippling effect where the locations of line breaks and page breaks in your document change.

Problems occur when images, frames, tables, and diagrams move from one page to another. The reflow situation can also cause orphans and widows – paragraphs that show a single dangling line, or even pages with a fragment of text at the top.

Kurinto's Approach

Kurinto provides several [metrically compatible](#) fonts for common fonts such as **TIMES NEW ROMAN**. These Kurinto fonts preserve the same line height and character widths of the common font, so the document should not reflow.



Typesface Anatomy

Font File Format

Kurinto fonts are .ttf files with TrueType outlines (quadratic Bézier curves) and OpenType tables.

Vertical Metrics

Kurinto fonts share consistent vertical (line height) metrics. This is a key principle that avoids [surprise layout changes](#) when switching between typesfaces or variants.

While Kurinto's approach to vertical metrics keeps line heights consistent and avoids surprise layout changes, it does have several issues: clipping and collisions.

Clipping

Some of the characters in Kurinto fonts extend above and/or below the vertical boundaries of the line. There are several possibilities for the "beyond bounds" portions of these characters:

- they could be "clipped" – cut off at the upper and/or lower boundary lines and incompletely rendered, or
- they could be "fully rendered" on the page, and possibly collide with another character (discussed in the next section).

In my experience⁴⁴, Microsoft Word will clip on-screen displays during document authoring and print preview displays, but that PDF rendering, hardcopy printouts, and browser rendering will display characters fully rendered, as long as they do not extend into margins or are covered by other object (such as shapes, graphics, text boxes, or frames).

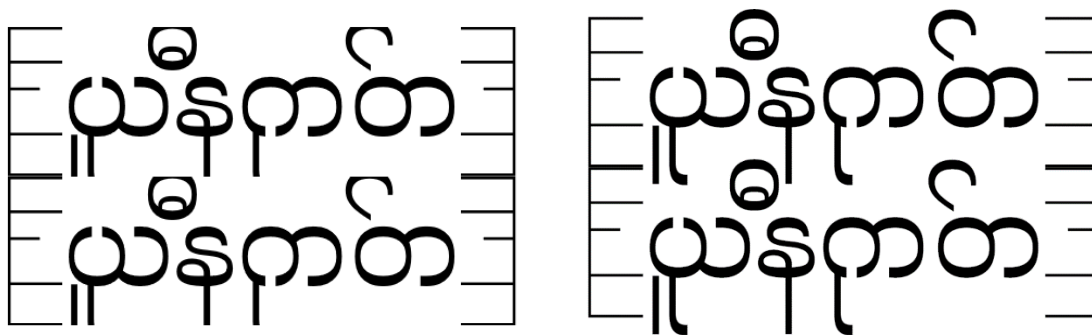
This means that, while viewing clipped characters when you are editing a document may be annoying, the final rendering will not be clipped.

Here are some examples of clipped characters, fully rendered characters, and character collisions. For these examples, I am using a translation of “Unicode” by Google Translate on March 20, 2020 into Myanmar (Burmese) as “ယူနီကုတ်” (“yuunekote”). The remainder of this section is typeset in **KURINTO BOOK KM**, to access the Myanmar writing system.

One helpful tool is to surround the Burmese text with the two vertical metrics characters U+10FF10 and U+10FF10: [ယူနီကုတ်]. This helps visualize the upper and lower bounds. Another technique is to intersperse several U+10FF14 vertical metrics overlay characters: [ယူနီကုတ်] to clearly see the portions of the characters that exceed the vertical bounds. It also helps to place two copies of the text on neighboring lines:

[ယူနီကုတ်] “Unicode” in Myanmar (Burmese) typeset in **KURINTO BOOK KM**
[ယူနီကုတ်] with a paragraph line height of “Single”.

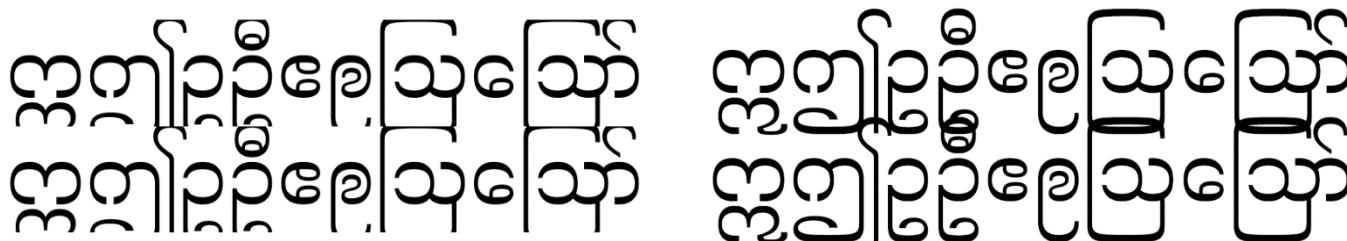
This text will display differently (clipped or fully rendered) depending on whether you are viewing the .docx or the .pdf version of this document, so I am including screenshot images of how they render in these two settings:



The left image is a screenshot from Microsoft Word, and the right is from Adobe Acrobat viewing the exported PDF file. The text is clipped in *Word*, but renders correctly in PDF. The characters overlap vertically, but do not actually intersect (collide). This might be acceptable if it occurs once or twice in your text.

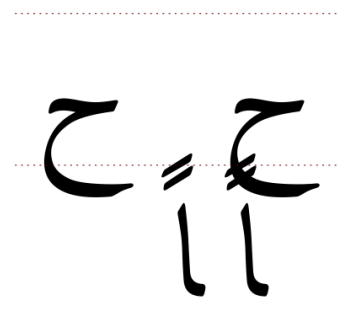
⁴⁴ Based specifically on Microsoft Word for Office 365 MSO, Adobe Acrobat 9.0.0 Pro, Hewlett Packard LaserJet Pro 400 color MFP m475dn, and Microsoft Windows 10x64. However, I have generally seen these same attributes over the years on various versions of Microsoft Windows and Office, PDF viewers, and an array of printers.

In the next pair of images, the text is far more problematic. I am using random Burmese characters that do collide. Again, the left image is a screenshot from Word and the right is from the rendered PDF file:



Collisions

Keeping a consistent line height does open the possibility of collisions – situations where characters on neighboring lines intersect. This issue crops up in writing systems that have very tall characters and character that extend much further from the baseline (up or down) than pan-European characters.



An extreme example of Arabic script is shown at the right. The upper line shows two characters U+062D ح ARABIC LETTER HAH, which extend substantially below the Lower Boundary Line. The second line shows two characters U+FD3C ا ARABIC LIGATURE ALEF WITH FATHATAN FINAL FORM, which extend slightly above the Upper Boundary Line. If the ALEF happens to appear above the HAH, they will collide:



There are two approaches to addressing this situation:

- If you encounter an occasional collision in your document, adjust the text so that the colliding characters do not intersect.
- If you use a writing system that has very tall characters that create lots of collisions, you can increase the line spacing so that no characters intersect. This can be done in *Word* using the “Multiple” choice for paragraph line spacing.

Calculating Vertical Distance

If your text is single-spaced (as is this document), you can determine the exact line height and number of lines per vertical distance. The table on the next page gives you some examples. The boldface rows with green background show the most common point sizes for text.

Font point size	Line Height			Count of Lines			
	points	inches	cm	per inch	per cm	per 11" page	per A4 page
6	8 1/4	0.115	0.291	8.727	3.436	96	102.047
7	9 5/8	0.134	0.340	7.481	2.945	82.286	87.469
8	11	0.153	0.388	6.545	2.577	72	76.535
8 1/2	11 11/16	0.162	0.412	6.160	2.425	67.765	72.033
9	12 3/8	0.172	0.437	5.818	2.291	64	68.031
9 1/2	13 1/16	0.181	0.461	5.512	2.170	60.632	64.451
10	13 3/4	0.191	0.485	5.236	2.062	57.600	61.228
10 1/2	14 7/16	0.201	0.509	4.987	1.963	54.857	58.313
11	15 1/8	0.210	0.534	4.760	1.874	52.364	55.662
11 1/2	15 13/16	0.220	0.558	4.553	1.793	50.087	53.242
12	16 1/2	0.229	0.582	4.364	1.718	48	51.024
13	17 7/8	0.248	0.631	4.028	1.586	44.308	47.099
14	19 1/4	0.267	0.679	3.740	1.473	41.143	43.735
15	20 5/8	0.286	0.728	3.491	1.374	38.400	40.819
16	22	0.306	0.776	3.273	1.288	36	38.268
18	24 3/4	0.344	0.873	2.909	1.145	32	34.016
20	27 1/2	0.382	0.970	2.618	1.031	28.800	30.614
22	30 1/4	0.420	1.067	2.380	0.937	26.182	27.831
24	33	0.458	1.164	2.182	0.859	24	25.512
26	35 3/4	0.497	1.261	2.014	0.793	22.154	23.549
30	41 1/4	0.573	1.455	1.745	0.687	19.200	20.409
36	49 1/2	0.688	1.746	1.455	0.573	16	17.008

font point size × 11/8 72 points per inch

For example, if you are using 12pt text in any Kurinto font (any typeface or variant), lines will be 16½ points high and an 8½ × 11" page will fit exactly 48 lines (with no margins).

If you are double-spacing your text, you can double the line height and halve the line count numbers. For other line spacing multipliers (which can be set in *Word* using the “Multiple” choice for line spacing), the numbers in the table can be adjusted accordingly.

Automatic Line-spacing Adjustments

Microsoft Word, in an effort to be “helpful”, automatically adjusts line spacing under certain conditions. As far as I know, these automatic adjustments cannot be switched off. One case where they occur is any line that contains an ideographic character, such as 永. It also happens with any font, such as **KURINTO TEXT JP**, that contains writing systems with logographic characters. The single character above is in **KURINTO TEXT JP**, while the rest of this paragraph is in **KURINTO TEXT**. *Notice that the third line of this paragraph has been given extra vertical space.*

An easy way to handle this situation it is to use Word’s “exact” feature. If you have a paragraph containing any text that causes a spurious line spacing adjustment, select the paragraph and bring up the Paragraph Settings dialog box. (here is the 永 character for demonstration ...) On the Indents and Spacing tab, in the Spacing section, for the Line Spacing choice, select “Exact” and enter the precise value that you want for line spacing. The table above in the Calculating Vertical Distance section provides line spacing settings for many common font sizes. This paragraph, set in 12pt **KURINTO TEXT**, has an Exact Line Spacing of 16.5pt. *Notice that the line height of the third line now matches the other lines in this paragraph.*

Font Identification

Every Kurinto font has these standard characters:



The character U+10FF00 Kurinto Font Folio v2.195 2020-07-24 FOLIO IDENTIFICATION is an English-language identification of the font folio.

The character to the left is U+10FF00 in 42pt **KURINTO TEXT**.

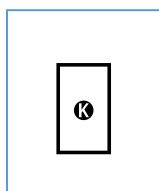
The character U+10FF01 Kurinto Text v2.195 2020-07-24 FONT IDENTIFICATION identifies the specific font in English-language text, including the full font name and the version number.

Kurinto Text
v2.195 2020-07-24

An example is shown to the right in 36pt **KURINTO TEXT**, showing the version of Kurinto that was used to render this *User’s Guide*.

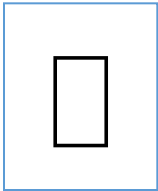
The character U+10FF02 **2.195** FONT VERSION shows the version number of the font with no other adornments. The example shown to the right is set in 36pt **KURINTO TEXT**, showing the version of Kurinto that was used to render this *User’s Guide*.

2.195



The character U+10FF08 □ NOT DEFINED CHARACTER shows the “.notdef” glyph – the glyph shown by the rendering engine if no glyph is available in the given (or substitute) font(s).

This character shows the .notdef glyph used by Kurinto, while the next character ...

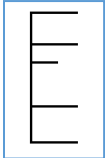



U+10FF09 □ NOT DEFINED CHARACTER PLAIN shows a generic “.notdef” glyph.

These NOT DEFINED code points give access to a glyph which is typically not accessible in a font, and allow documentation of the specifics of a font, such as is done throughout this *User’s Guide*.

Visual Font Metrics

A set of characters in the range U+10FF10–U+10FF1D let authors visualize the vertical and horizontal spacing of the font.




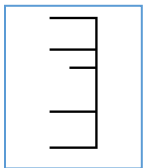
U+10FF10  VERTICAL METRICS GUIDE LEFT is a character that visually demonstrates the key vertical metrics of the font. The character is closed on the left side and open on the right side.







The horizontal lines of the character show, from top to bottom:


- Upper Boundary Line ([UBL](#)) – a horizontal line that represents the intended vertical boundary between a line of text and the end of the previous line of text. Assuming that lines are single-spaced, the UBL of one line coincides with the LBL of the previous line of text. See [Boundary Lines](#). The top of this line matches the UBL of the font.
- Cap height – the height of capital letters in the current font. The top of this line matches the Cap Height of the font.
- X height – the height of a lower-case “x” in this font. The top of this line matches the X Height, or “midline”, of the font.
- Baseline – the horizontal baseline of all characters. The *bottom* of this line matches the baseline of the font.
- Lower Boundary Line ([LBL](#)) – the intended vertical boundary between this line of text and the subsequent line of text. See [Boundary Lines](#). The bottom of this line matches the LBL of the font.

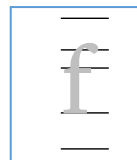
See [Vertical Metrics](#) for details. See [Vertical Metrics – Technical Details](#) for more detailed information on these metrics.

U+10FF11  VERTICAL METRICS GUIDE RIGHT is the pair of the character above. It is open on the left side and closed on the right side. These two characters can be useful in showing the vertical alignment of characters:



 Now is the time 
 for all good men 
 to come to the aid 

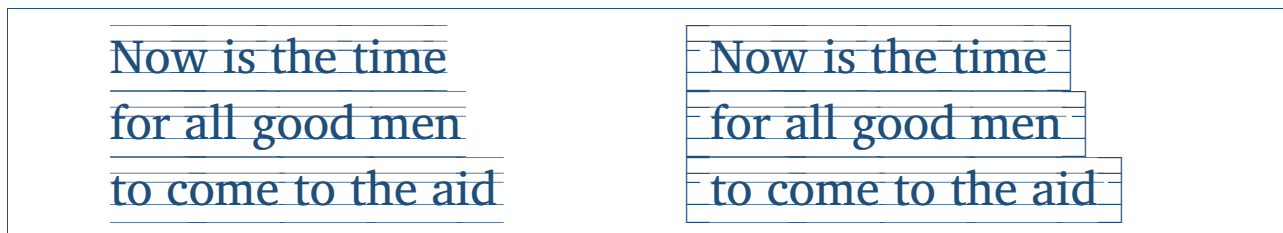
U+10FF12  VERTICAL METRICS OVERLAY EN serves a similar function to the two characters above, but places the guide-lines over the character(s) that follow (i.e. it has zero [advance width](#)). It has the width of an En space. In practical situations, it is helpful to have wider versions of this character, to cover more horizontal space:








U+10FF13  VERTICAL METRICS OVERLAY EM is twice as wide (an Em space).

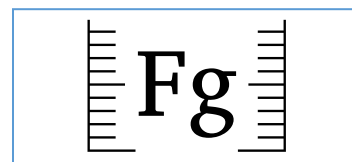
U+10FF14  VERTICAL METRICS OVERLAY 2EM is again twice as wide (two Em spaces).

Here are some examples. The version on the right is using all the characters in the range U+10FF10–U+10FF14:




The five VERTICAL DECIMAL characters:

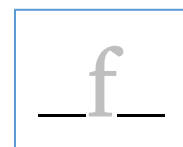
- U+10FF15  VERTICAL DECIMAL GUIDE LEFT
- U+10FF16  VERTICAL DECIMAL GUIDE RIGHT
- U+10FF17  VERTICAL DECIMAL OVERLAY EN
- U+10FF18  VERTICAL DECIMAL OVERLAY EM
- U+10FF19  VERTICAL DECIMAL OVERLAY 2EM



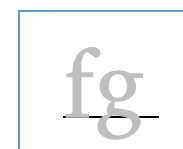
... serve the same function as the three VERTICAL METRICS characters, except they divide the vertical distance within a line into 10 equally spaced vertical divisions. Note that the top line is absent – similar VERTICAL OVERLAY characters on the line above would serve this purpose:





U+10FF1A  BASELINE GUIDE is a character that indicates the baseline. The bottom of this character coincides with the baseline. It is one en wide. For example:



The next three characters are versions of the BASELINE GUIDE that place the guideline over the character(s) that follow (i.e. they have zero [advance width](#)):




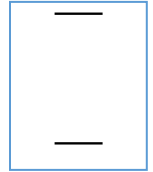
- U+10FF1B  BASELINE OVERLAY EN
- U+10FF1C  BASELINE OVERLAY EM

- U+10FF1D  BASELINE OVERLAY 2EM

For example:

Now is the time




U+10FF1E  VERTICAL LIMITS GUIDE is a character that indicates the Upper and Lower Boundary Lines ([UBL](#) and [LBL](#)). The top of the upper line coincides with the UBL and the bottom of the lower line coincides with the LBL. For example:



Now is the time

The next three characters are versions of the VERTICAL LIMITS GUIDE that place the guideline over the character(s) that follow (i.e. they have zero [advance width](#)):







- U+10FF1F  VERTICAL LIMITS OVERLAY EN
- U+10FF20  VERTICAL LIMITS OVERLAY EM
- U+10FF21  VERTICAL LIMITS OVERLAY 2EM

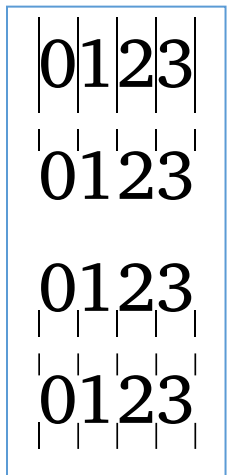
For example:

Now is the time

The four LSB OVERLAY characters:

- U+10FF22  LSB MARKER OVERLAY
- U+10FF23  LSB TOP OVERLAY
- U+10FF24  LSB BOTTOM OVERLAY
- U+10FF25  LSB TOP BOTTOM OVERLAY

... provide a handy way to visualize horizontal character spacing. Each of them overlays the character that follows (i.e. they have zero advance width) and mark the LSB point of that following character in a different style.



You can insert a marker between each character, above the characters, below the characters, or both above and below the characters.

This section shows paragraphs set in various point sizes, using two of the Visual Font Metrics characters described above: U+10FF10 and U+10FF1D. The line height and lines per inch are given in the table above.

11. Разъяренный чтец эгоистично бьёт пятью жердями шустрого фехтовальщика.
12. いろはにほへとちりぬるをわかよたれそつねならむうゑのおくやまけふこえてあさき

1. Kurinto Seri 16 Point = 3.273 lines per inch – Kurinto Seri
v2.195 2020-07-24 **v2.195.**
2. The quick brown fox jumps over the lazy dog.
3. Victor jagt zwölf Boxkämpfer quer über den großen Sylter Deich.
4. Þú dazt á hnéð í vök og yfir blóm sexý pæju.
5. Høj bly gom vandt fræk sexquiz på wc.
6. Do bạch kim rất quý, sẽ để lấp vô xương.
7. 天地玄黄，宇宙洪荒。日月盈昃，辰宿列张。寒来暑往，秋收冬藏 (HK)
8. ऋषियों को सताने वाले दुष्ट राक्षसों के राजा रावण का सर्वनाश करने वाले विष्णुवतार
9. نَصُّ حَكِيمٍ لَهُ سِرٌّ قَاطِعٌ وَذُو شَأْنٍ عَظِيمٍ مَكْتُوبٌ عَلَى ثَوْبٍ أَخْضَرَ وَمُغْلَفٌ بِجِلْدٍ أَزْرَقٍ
10. অদ্য আষাঢ়ে উষায় ঐশাণ কোণে মেঘের ফাঁকে বিদ্যুৎছটার বলক ঠাণ্ডে ঋষভ সিংহ

1. Kurinto Sans 24pt=2.182 lines/in Kurinto Sans
v2.195 2020-07-24 **v2.195**
2. The quick brown fox jumps over the lazy
3. Victor jagt zwölf Boxkämpfer quer über den
4. Þú dazt á hnéð í vök og yfir blóm sexý pæju.
5. Høj bly gom vandt fræk sexquiz på wc.
6. Do bạch kim rất quý, sẽ để lấp vô xương.
7. 天地玄黄，宇宙洪荒。日月盈昃， (HK)
8. ऋषियों को सताने वाले दुष्ट राक्षसों के राजा रावण का
9. نَصُّ حَكِيمٍ لَهُ سِرٌّ قَاطِعٌ وَذُو شَأْنٍ عَظِيمٍ مَكْتُوبٌ عَلَى

10. অদ্য আষাড়ে উষায় ঈশাণ কোণে মেঘের ফাঁকে

Width of Digits and Spacing Characters

All the digit characters in a font with [tabular figures](#) have the same width. The character U+2007 FIGURE SPACE also shares the same width as the digits.

Character Alignment in Asian Writing Systems

Asian languages contain thousands of “logographic” characters – complex characters in the “kanji” character set that have many strokes. Kanji characters were initially derived from calligraphic pictures of what they represent.⁴⁵

In addition, Japanese uses three additional character sets: katakana, hiragana, and often mixes in Western characters as well. These four writing systems can be used in combination within the same sentence, each with its own purpose and rules, as well as its own visual weight and texture.

Visually complex kanji combine with each other to represent most objects, ideas and actions. Flowing hiragana connects and conjugates kanji among other uses, while angular katakana represents non-Japanese proper nouns and concepts.

In the image below, characters are underlined in purple. There are many thousands of kanji characters with about 2,000 used in everyday Japanese.

母の誕生日にヒヤシンスの
鉢植えを贈った。

“I bought a potted hyacinth for my mother’s birthday.” – Image by Eiko Nagase

Hiragana characters are underlined in blue and Katakana characters are show in green. There are about 48 of each of these classes of characters.

This mix of writing systems creates substantial typographic challenges. Here are a few basic principles used in Kurinto that reduce the complexity of typesetting Asian languages:

- Kurinto pairs “Gothic” style characters with sans serif fonts (**KURINTO SANS**, **SERI**, and **MONO**) and “Mincho” style characters with serif fonts (**KURINTO BOOK** and **TYPE**).
- The full-cap height and square profile of Asian characters make them appear larger than Western characters. Kurinto compensates by reducing the effective point size for Asian characters by 10–15%.
- Kurinto moves the baseline of Asian characters down to a separate “Ideographic Baseline”.

On the assumption that Kurinto will most often be used in documents that mix Western and Asian writing systems, the goal is to minimize (to some extent) the “all caps” look that happens when Asian characters are intermixed with upper-case and lower-case Western text.

⁴⁵ This section is based on a blog post by Eiko Nagase, available at <http://aqworks.com/en/blog/2016/09/20/perfect-japanese-typography/>.

Here is a sample interleaving English and Japanese “my mother’s birthday” set in 32 point
KURINTO TEXT JP:

My 母の mother’s 誕生日 birthday

Notice that, while the Hiragana character (の) aligns to the baseline and (with a small overshoot) the cap-height, the Kanji characters (誕生日) have a noticeably lower baseline and upper bound.

Roadmaps

This section provides roadmaps to the blocks of Unicode code points and how they are used in Kurinto fonts. This information is current as of Unicode version 12.1.

Unicode Planes

This first roadmap shows the overall structure of the 17 Unicode “planes” and how they are currently allocated.

Roadmap of All Unicode Planes																
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00__ ... FF__	Plane 0 - Basic Multilingual Plane (BMP)														Private Use Area (PUA)	
100__ ... 1FF__	Plane 1 - Supplementary Multilingual Plane (SMP)															
200__ ... 2FF__	Plane 2 - Supplementary Ideographic Plane (SIP)															
300__ ... 3FF__	Plane 3 - Tertiary Ideographic Plane (TIP)															
400__ ... DFF__	Planes 4-13 - Unassigned / Reserved															
E00__ ... EFF__	Plane 14 - Supplementary Special-Purpose Plane (SSP)															
F00__ ... FFF__	Plane 15 - Supplementary Private Use Area - A (SPUA-A)															
1000__ ... 10FF__	Plane 16 - Supplementary Private Use Area - B (SPUA-B)															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Assigned (partially)				Private Use			(unassigned)								

The green areas contain code points assigned by *The Unicode Standard*. Not all code points in these areas are currently assigned to characters. However, those not currently assigned are reserved for future allocation.

The pink zones are set aside for “private use” by organizations and font designers. Kurinto makes extensive use of these private use areas for writing systems and collections of characters that are not currently part of Unicode.

Core Variant Fonts – Full Roadmap

Subsequent roadmaps show the blocks as they are allocated within a specific plane and font variant. Many of the blocks of code points are allocated to a specific writing system.

This Roadmap shows the assigned code points in each of the **CORE** font variants:

Roadmap of All Core Variant Fonts																		
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
00__	Basic Latin (95 of 95)								Latin-1 Supplement (96 of 96)									
01__	Latin Extended-A (128 of 128)								Latin Extended-B (7 of 208)									
02__												Spacing Modifier Letters (9 of 80)						
03__	Combining Diacritical Marks (1 of 112)							Greek and Coptic (72 of 135 - all Greek)										
04__	Cyrillic (98 of 256)																	
05__																		
1D__																		
1E__	Latin Extended Additional (8 of 256)																	
1F__																		
20__	General Punctuation (50 of 111 of which 23 are WGL4)							Superscripts and Subscripts (1 of 42)			Currency Symbols (4 of 32)							
21__	Letterlike Symbols (6 of 80)					Number Forms (4 of 60)				Arrows (7 of 112)								
22__	Mathematical Symbols (17 of 256)																	
23__	Miscellaneous Technical (4 of 256)																	
24__																		
25__	Box Drawing (40 of 128)								Blocks (8 of 32)		Geometric Shapes (15 of 96)							
26__	Miscellaneous Symbols (11 of 256)																	
27__																		
DF__																		
E0__ F8__	Private Use Area (2 of 6400)																	
F9__ FA__																		
FB__	Alphabetic Presentation Forms (2 of 58)																	
FC__ FD__																		
FE__												Arabic Presentation Forms B (1 of 141)						
FF__															Specials (5 of 5)			
100__ 10FE__																		
10FF__	Font Ident	Visual Font Metrics																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
	All characters are WGL4				Mix of WGL4 and non-WGL4				Non-WGL4			Kurinto			(unused)			

The **CORE** font variants contain all the characters in the WGL4 subset plus some additional characters.

In addition, some of the core fonts include an additional set of blocks for Alternate Capitals and Figures (ACF, or “Alt Caps and Figures”):

Roadmap for the ACF (Alt Caps and Figures) in Core Variant Fonts																
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10FA	Alt Caps and Figures: Small Caps															
10FB	Alt Caps and Figures: Petite Caps															
10FC	Alt Caps and Figures: Titling Caps															
10FD	Cyrillic Small Caps				Cyrillic Petite Caps				Cyrillic Titling Caps							
10FE	Prop Lining	Prop OldStyle	Prop Hybrid	Prop Ovscore	Tabular Lining	Tabular OldStyle	Tabular Hybrid	Tabular Ovscore								
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Kurinto		(unused)													

These characters are accessible directly by code points or (in many cases) automatically using OpenType layout features.

For a complete list of characters that are in the ACF blocks, see [Appendix 2](#).

Plane 0 – Kurinto Basic Multilingual Plane

This two-page chart that follows shows how Kurinto allocates the Basic Multilingual Plane to font variants.

Please note: The roadmap charts that follow are useful as a general guide to how the characters are allocated across font variants. However, since Kurinto divides allocates **writing systems** to variants rather than blocks of code points, these charts are not precise.

Blocks with horizontal lines in them are not currently implemented due to lack of available, usable source fonts.

Roadmap of the Basic Multilingual Plane in Kurinto Font Variants																	
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00	Basic Latin								Latin-1 Supplement								
01	Latin Extended-A								Latin Extended-B								
02	Latin Extended-B				IPA Extensions				Spacing Modifier Letters								
03	Combining Diacritical Marks						Greek and Coptic										
04	Cyrillic																
05	Cyrillic Supplement			Armenian						Hebrew							
06	Arabic																
07	Syriac				Arabic Supplement				Thaana				NKo				
08	Samaritan			Mandaic		Syriac Supp.	(reserved)				Arabic Extended-A						
09	Devanagari								Bengali								
0A	Gurmukhi								Gujarati								
0B	Oriya								Tamil								
0C	Telugu								Kannada								
0D	Malayalam								Sinhala								
0E	Thai								Lao								
0F	Tibetan																
10	Myanmar									Georgian							
11	Hangul Jamo																
12	Ethiopic																
13	Ethiopic								Ethiopic Supp.		Cherokee						
14	Unified Canadian Aboriginal Syllabics																
15																	
16									Ogham		Runic						
17	Tagalog		Hanunóo		Buhid		Tagbanwa		Khmer								
18	Mongolian										Unified Canadian Aboriginal Syllabics Extended						
19	Limbu				Tai Le			New Tai Lue						Khmer Symbols			
1A	Buginese		Tai Tham										Combining Diacritical Marks Extended				
1B	Balinese								Sundanese				Batak				
1C	Lepcha				Ol Chiki			Cyrillic Ext-C	Georgian Extended				Sundanese Supp.	Vedic Extensions			
1D	Phonetic Extensions								Phonetic Extensions Supplement				Combining Diacritical Marks Supplement				
1E	Latin Extended Additional																
1F	Greek Extended																
20	General Punctuation						Superscripts and Subscripts			Currency Symbols			Combining Diacritical Marks for Symbols				
21	Letterlike Symbols				Number Forms				Arrows								
22	Mathematical Symbols																
23	Miscellaneous Technical																
24	Control Pictures			Optical Char. Recognition		Enclosed Alphanumerics											
25	Box Drawing								Blocks		Geometric Shapes						
26	Miscellaneous Symbols																
27	Dingbats												Miscellaneous Mathematical Symbols-A			Supp. Arrows-A	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	Kurinto Core		Kurinto Main		Kurinto Aux		Kurinto HK, JP, SC, TC, and TW					Kurinto KM		Kurinto TB			
	(unused)		Italics = Right-to-Left														

Roadmap of the Basic Multilingual Plane in Kurinto Font Variants - Continued																	
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
28	Braille Patterns																
29	Supplemental Arrows-B								Miscellaneous Mathematical Symbols-B								
2A	Supplemental Mathematical Operators																
2B	Miscellaneous Symbols and Arrows																
2C	Glagolitic					Laitn Extended-C			Coptic								
2D	Georgian Supplement			Tifinagh					Ethiopic Extended					Cyrillic Extended-A			
2E	Supplemental Punctuation								CJK Radicals Supplement								
2F	Kangxi Radicals															Ideogr. Descr. Chars	
30	CJK Symbols and Punctuation				Hiragana						Katakana						
31	Bopomofo			Hangul Compatibility Jamo						Kanbun	Bopomofo Extended	CJK Strokes			Katakana Phonetic Ext.		
32	Enclosed CJK Letters and Months																
33	CJK Compatibility																
34	CJK Unified Ideographs Extension A																
4C																	
4D															Yijing Hexagram Symbols		
4E	CJK Unified Ideographs																
9F																	
A0	Yi Syllables																
A3																	
A4									Yi Radicals				Lisu				
A5	Vai																
A6	Vai				Cyrillic Extended-B						Bamum						
A7	Modifier Tone Letters		Latin Extended-D														
A8	Syloti Nagri			Common Indic Number Forms	Phags-pa				Saurashtra					Devanagari Extended			
A9	Kayah Li			Rejang		Hangul Jamo Extended-A			Javanese					Myanmar Extended-B			
AA	Cham					Myanmar Extended-A			Tai Viet					Meetei Mayek Extensions			
AB	Ethiopic Extended-A			Latin Extended-E			Cherokee Supplement					Meetei Mayek					
AC	Hangul Syllables																
D6																	
D7											Hangul Jamo Extended-B						
D8	Surrogates (reserved for exclusive use as UTF-16 surrogates and are not legal code points for characters)																
DF																	
E0	Private Use Area																
F8																	
F9	CJK Compatibility Ideographs																
FA																	
FB	Alphabetic Presentation Forms																
FC	Arabic Presentation Forms A																
FD																	
FE	Variation Selectors	Vertical Forms	Combining Half Marks	CJK Compatibility Forms		Small Form Variants		Arabic Presentation Forms B									
FF	Halfwidth and Fullwidth Forms															Specials	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	Kurinto Main		Kurinto Aux		Kurinto HK, JP, SC, TC, and TW						Private Use		(unused)		Ital = R-to-L		

Private Use Area Characters

The **CORE** variant of Kurinto fonts has only two characters in the Private Use Area: the two characters in the WGL4 block at U+F000 and U+F001 are the ligatures “fi” and “fl” that are part of the WGL4 subset.

The **MAIN** variant of Kurinto fonts adds other areas that are reserved for compatibility with various corporate standards:

Roadmap for the Private Use Area in the Kurinto Main Variant Fonts																
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0 EF	(unused)															
F0	WGL4	Box-A	(Reserved for Microsoft Symbol Fonts)													
F1	48			SIL (Kurinto Book only)												
F2																
F3																
F4 F7	(unused)															
F8	(Reserved for Apple Compatibility)										Apple					
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Windows Glyph List 4				Box Drawing Ext-A			SIL Corporate			Reserved		(unused)			
The Apple reserved area at shown at F8F0-FF is only for the single character F8FF - the Apple Logo.																

The Box-A block provides a space character that matches the width of characters in the Box Drawing and Block Elements blocks. It also provides other box drawing entries that may be of use.

The SIL block is reserved for characters allocated by SIL International and defined in their *SIL Corporate PUA Documentation*. For a detailed list of characters, see the Kurinto_CodeChart_main.pdf document in the /Doc directory of the distribution package. See also https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=PUA_home.

Aux

The AUX variant of Kurinto fonts has many writing systems that are not part of the Unicode standard:

Roadmap for the Private Use Area in the Kurinto Aux Variant Fonts																	
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
E0	Tengwar								Cirth								
E1	Engsvanyáli					Kinya						Ilianore		Syai			
E2	Verdurian						aUI		Amman-lar			Streich	Xaîni				
E3	Mizarian				Zírínka		Sarkai		Thelwik			Olaetyan					
E4	Nísklôz		Kazat ?Akkorou		Kazvarad		Zarkhánd		Røzhxh			Serivelna		Kelwathi			
E5	Saklor		Rynnan			Alzetjan			Telarasso		Ssûraki		Gargoyle		Ophidian		
E6	Ferengi		Seuss		Sylabica			Ewellic					Amlin		Unifon Extended		
E7	Unifon Extended				Unifon			Solresol	Visible Speech								
E8	Monofon		D'ni						Aurebesh				Tonal				
E9	Glaitha-A								Glaitha-B								
EA	Lhenazi										Wanya						
EB	Orokin (Tennobet)				Std. Galactic		Braille Extended				96						
EC	Cylenian			Syrrin				144									
ED	Deini				Niji				128								
EE	256																
EF	256																
F0	WGL4	(Reserved for Microsoft Symbol Fonts)															
F1	256																
F2	256																
F3	256																
F4													Ath			256	
F5	256																
F6	256																
F7	Creative Commons				208												
F8	(Reserved for Apple Compatibility)										Aiha (Kesh)			Klingon			
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	UCSUR-Extended		WGL4	Creative Commons			Reserved		(unused)								
	UCSUR-Extended (Not implemented - No fonts available)																

Many of these writing systems are part of an extended version of the Under-ConScript Unicode Registry (UCSUR-Ext) maintained by Rebecca Bettencourt. See the Kurinto_CodeChart_Aux.pdf and Kurinto_CodeChart_Aux2.pdf documents in the /Doc directory of the distribution package. See also <https://www.kreativekorp.com/ucsur/>.

Music

The **MUSIC** variant of Kurinto fonts contains a large block of characters in the Private Use Area from the Standard Music Font Layout (SMuFL) Version 1.3 specification (see <https://www.w3.org/2019/03/smufl13/>).

Roadmap for the Private Use Area in the Kurinto Music Variant Fonts																
U+	_0_	_1_	_2_	_3_	_4_	_5_	_6_	_7_	_8_	_9_	_A_	_B_	_C_	_D_	_E_	_F_
E0__	SMuFL 1.3															
...																
ED__																
EE__																
EF__																
F0__	WGL4	(Reserved for Microsoft Symbol Fonts)														
F1__																
F7__																
F8__	(Reserved for Apple Compatibility)															
	0	_1_	_2_	_3_	_4_	_5_	_6_	_7_	_8_	_9_	_A_	_B_	_C_	_D_	_E_	_F_
	SMuFL 1.3	WGL4	Reserved	(unused)												

For a complete roster of characters, see the Kurinto_CodeChart_Music.pdf document in the /Doc subdirectory of the distribution package.

UFI

The **UFI** variant of Kurinto contains a superset of characters from various *Unicode Font Initiative* specifications, including MUFI, CYFI, and RUFI. These encoding standards have been developed by scholars in Medieval studies, linguists, and font designers to support special characters in medieval texts written in the Latin, Cyrillic, and Runic writing systems.

Roadmap for the Private Use Area in the Kurinto UFI Variant Fonts																		
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
E0 E3	Uppercase Latin																	
E4 E7	Lowercase Latin																	
E8											Medieval Latin							
E9																		
EA											MUFI Extension-A							
EB											MUFI Extension-B							
EC																		
ED																		
EE													MUFI Ligatures-A		Enlarged Miniscules			
EF	Small Capitals								Cyrillic-A		MUFI Ext-D	MUFI Lig-B	MUFI Extension-E		Accented Ligatures			
F0	WGL4 & MUFI	Combining Diacritics				Lowercase With Superscripts												
F1	MUFI Extension-C												Combining Marks		General Punctuation			
F2	MUFI Extension-F														Roman Signs		Abbreviations	
F3				Cyrillic-B														
F4																	MUFI Ligatures-C	
F5																		
F6																		
F7	Metrical Characters				Double Diacritics				Number Forms									
F8	Identi fication																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
	MUFI		CYFI		(unused)													

The intent of a “superset” font is to provide a wide range of characters that might exist in documents of various vintages – i.e. that might have been authored at various times during the development of the encoding standards. This "permissive" approach means that, even if a characters at a particular code point has been relocated or decommissioned, these code charts (and the Kurinto MUFI fonts) retain the prior code point mappings. Documents that rely on the old code point mappings should still “work”.

The downside of this “superset” approach is that it supports the use of deprecated and decommissioned characters. Authors are encouraged to check the annotations in these code charts

for such deprecated and decommissioned code points and remap their documents as advised. Also, there are cases where a given code point has been used for more than one character in different standards or versions of a standard. These cases have been resolved by using the most recent encoding standard for that code point.

The general layout of the Private Use Area for UFI fonts is shown on the next page. For more detailed information, including all characters (including those outside the PUA) in the superset of UFI standard as well as extensive annotations on each character, see the *Kurinto Code Charts for the UFI Variant Fonts* provided as a PDF in each distribution package.

Primary Encoding Standards and Sources

MUFI

A recent version of the Medieval Unicode Font Initiative, available at <https://MUFI.info/>. This web site and its underlying database is maintained by Tarrin Wills, Odd Einar Haugen, and other MUFI Board members. This version of Kurinto is based on the MUFI database available as of 2020-07-05. Note that the current version of MUFI is sometimes referred to as "beyond MUFI 4.0" or "MUFI 4 +".

CYFI

The Cyrillic Font Initiative, sourced from:

Victor M. Baranov, David Birnbaum, Ralph Cleminson, Heinz Miklas, and Achim Rabus, *Proposal for a Unified Encoding of Early Cyrillic Glyphs in the Unicode Private Use Area*, Scripta and e-Scripta, Volume 8-9, 2010, pages 9-26, retrieved from <https://www.semanticscholar.org/paper/Proposal-for-a-unified-encoding-of-Early-Cyrillic-A-Baranov-Birnbaum/e20a7450acfd4f62588445d31a163b3b37f22b16> on 2019-08-29.

RUFI

The Runic Font Initiative, based on entries in the MUFI database (cited above), with characters as proposed by Paola Peratello and added by Odd Einar Haugen 2020-02-22. See <https://runic.jimdofree.com/> for background.

Additional Sources

These sources provide information that is often cited in the annotations in these code charts:

MUFI1 through MUFI4

The Medieval Unicode Font Initiative, versions 1.0 through 4.0. These are available in a set of documents (including Errata):

Odd Einar Haugen, MUFI Character Recommendation v. 1.0, 2003-12-08, ISBN 82-90500-58-0, available at <https://hdl.handle.net/1956/2004>, retrieved 2020-07-04. The latest Errata is available at https://folk.uib.no/hnooh/mufi/specs/errata_1-1.html, updated 2007-01-03, retrieved 2020-07-04.

Odd Einar Haugen, MUFI Character Recommendation v. 2.0, Part 2: Code Chart Order, 2006-12-22, ISBN 978-82-8088-529-6, available at <https://hdl.handle.net/1956/2003>, retrieved 2019-08-22.

Odd Einar Haugen (editor), MUFI Character Recommendation v. 3.0, Part 2: Code Chart Order, 2009-07-05 (PDF last modified 2019-08-20), ISBN 978-82-8088-403-9, available at <https://hdl.handle.net/1956/3955>, retrieved 2019-08-22. The Errata is available at <https://folk.uib.no/hnooh/mufi/specs/errata-3-0.html>, dated 2010-07-20, retrieved 2020-07-04.

Odd Einar Haugen, MUFI Character Recommendation v. 4.0, Part 2: Code Chart Order, 2015-12-22, available at <https://hdl.handle.net/1956/10699>, retrieved 2019-05-05.

Wiki

The Medieval Unicode Font Initiative Wikipedia page at, retrieved from https://en.wikipedia.org/wiki/Medieval_Unicode_Font_Initiative on 2020-07-02.

Menota

Another datastore of MUFI information:

Odd Einar Haugen (collector and editor), Menota Entities, 2020-06-11, retrieved from <https://www.menota.org/menota-entities.txt> 2020-07-01.

Bettencourt

Rebecca G. Bettencourt, MUFI UnicodeData Text File, retrieved from <http://www.kreativekorp.com/charset/PUADATA/PUBLIC/MUFI/> on 2020-07-01.

LaTeX

Mikkel Eide Eriksen, Unicode Alphabets for LaTeX, Specimen, 2020-03-11, retrieved from <http://mirror.las.iastate.edu/tex-archive/macros/latex/contrib/unicode-alphabets/docs/specimen.pdf> on 2020-07-04.

Plane 1 – Kurinto Supplementary Multilingual Plane

Roadmap of the Supplementary Multilingual Plane in Kurinto Font Variants																	
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
100	Linear B Syllabary								Linear B Ideograms								
101	Aegean Numbers				Ancient Greek Numbers					Ancient Symbols				Phaistos Disc			
102									Lycian		Carian				Coptic Ep No		
103	Old Italic			Gothic		Old Permic			Ugaritic		Old Persian						
104	Deseret					Shavian			Osmanya			Osage					
105	Elbasan			Caucasian Albanian													
106	Linear A																
107																	
108	Cypriot Syllabary				Imp.Aramaic		Palmyrene		Nabataean				Hatran				
109	Phoenician		Lydian						Meroitic H.		Meroitic Cursive						
10A	Kharoshthi					O.S.Arabian		O.N.Arabian						Manichaean			
10B	Avestan				Parthian		Insc. Pahlavi		Psalt. Pahlavi								
10C	Old Turkic								Old Hungarian								
10D	Hanifi Rohingya																
10E						Rumi Symb.											
10F	Old Sogdian			Sogdian												Elymaic	
110	Brahmi								Kaithi				Sora Sompeng				
111	Chakma				Mahajani				Sharada				Sinh Arch No				
112	Khojki								Multani			Khudawadi					
113	Grantha																
114	Newa								Tirhuta								
115									Siddham								
116	Modi					Mong. Supp.		Takri									
117	Ahom																
118	Dogra													Warang Citi			
119									Nandinagari								
11A	Zanabazar Square				Soyombo								Pau Cin Hau				
11B																	
11C	Bhaiksuki							Marchen									
11D	Masaram Gondi					Gunjala Gondi											
11E															Makasar		
11F													Tamil Supplement				
120	Cuneiform																
123																	
124	Cuneiform Numbers								Early Dynastic Cuneiform								
125	Early Dynastic Cuneiform																
126																	
12F																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	Kurinto Main		Kurinto Aux		(unused)		Italics = Right-to-Left										
	Horizontal lines = not implemented - no source fonts available																

Roadmap of the Supplementary Multilingual Plane in Kurinto Font Variants - Continued																
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
127__ 133__	Egyptian Hieroglyphs															
134__				Egyp. Hieroglyph Fmt. Contr.												
135__ 143__																
144__ 145__	Anatolian Hieroglyphs															
146__																
147__ 167__																
168__ 169__	Bamum Supplement															
16A__					Mro								Bassa Vah			
16B__	Pahawh Hmong															
16C__																
16D__																
16E__					Medefaidrin											
16F__	Miao												Ideo. Sym&Punc.			
170__ 187__	Tangut															
188__ 18A__	Tangut Components															
18B__ 1AF__																
1B0__	Kana Supplement															
1B1__	Kana Ext.-A			Small Kana Ext.				Nushu								
1B2__	Nushu															
1B3__ 1BB__																
1BC__	Duployan									ShFC						
1BD__ 1CF__																
1D0__	Byzantine Musical Symbols															
1D1__	Musical Symbols															
1D2__	Ancient Greek Musical Notation														Mayan Numerals	
1D3__	Tai Xuan Jing Symbols					Counting Rod Numerals										
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Kurinto Main		Kurinto Aux		(unused)		Italics = Right-to-Left									
	Horizontal lines = not implemented - no source fonts available															

Roadmap of the Supplementary Multilingual Plane in Kurinto Font Variants - Continued																	
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1D4__ 1D7__	Mathematical Alphanumeric Symbols																
1D8__ 1D9__	Sutton SignWriting																
1DA__																	
1DB__ 1DF__																	
1E0__	Glagolitic Supp.																
1E1__	Nyiakeng Puachue Hmong																
1E2__													Wancho				
1E3__ 1E7__																	
1E8__	Mende Kikakui																
1E9__	Adlam																
1EA__																	
1EB__																	
1EC__								Indic Siyaq Numbers									
1ED__	Ottoman Siyaq Numbers																
1EE__	Arabic Mathematical Alphabetic Symbols																
1EF__	(reserved)																
1F0__	Mahjong Tiles			Domino Tiles						Playing Cards							
1F1__	Enclosed Alphanumeric Supplement																
1F2__	Enclosed Ideographic Supplement																
1F3__ 1F5__	Miscellaneous Symbols and Pictographs																
1F6__	Emoticons				Ornamental Dingbats				Transport and Map Symbols								
1F7__	Alchemical Symbols								Geometric Shapes Extended								
1F8__	Supplemental Arrows-C																
1F9__	Supplemental Symbols and Pictographs																
1FA__	Chess Symbols						Symbols and Pictographs Extended-A										
1FB__ 1FF__																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	Kurinto Main		Kurinto Aux		Kurinto HK, JP, SC, TC, and TW						(unused)		Italics = Right-to-Left				
	Horizontal lines = not implemented - no source fonts available																

Plane 2 – Kurinto Supplementary Ideographic Plane

Roadmap of the Unicode Supplementary Ideographic Plane (SIP)																
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
200 2A5	CJK Unified Ideographs Extension B															
2A6																
2A7 2B6	CJK Unified Ideographs Extension C															
2B7					CJK Unified Ideographs Extension D											
2B8	CJK Unified Ideographs Extension D															
2B9 2CD	CJK Unified Ideographs Extension E															
2CE																
2CF 2EA	CJK Unified Ideographs Extension F															
2EB																
2EC 2F7																
2F8	CJK Compatibility Ideographs Supplement															
2F9																
2FA																
2FB 2FF																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Kurinto HK, JP, SC, TC, and TW						(unused)									

Plane 3 – Kurinto Tertiary Ideographic Plane

As of Unicode 12.1, the Tertiary Ideographic Plane has no assigned characters. This changed on March 10, 2020 with the release of Unicode v13. It looks something like this, *but these blocks have not yet been implemented in Kurinto fonts*:

Roadmap of the Unicode Tertiary Ideographic Plane (<i>TIP</i>)																
<i>U+</i>	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
300 312	(CJK Unified Ideographs Extension G)															
313						(reserved)										
314 33C	(Small Seal Script)															
33D			(reserved)													
33E 355	(Oracle Bone Script)															
356 2FF	(reserved)															
	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
	Kurinto HK, JP, SC, TC, and TW						Kurinto Aux		(unused)							
	Horizontal lines = not currently implemented															

Plane 14 – Kurinto Supplementary Special-Purpose Plane

The Supplementary Special-Purpose Plane does not contain any graphic characters.

Plane 15 – Kurinto Supplementary Private Use Area - A

This plane is almost entirely allocated to private use. This is used in the Aux fonts for a number of writing systems and useful icons:

Roadmap for Supplementary Private Use Area - A in the Kurinto Aux Variant Fonts																																		
U+	_0_	_1_	_2_	_3_	_4_	_5_	_6_	_7_	_8_	_9_	_A_	_B_	_C_	_D_	_E_	_F_																		
F00__ F0D__	Kinya Syllables																																	
F0E__																																		
F0F__ F15__	Pikto																																	
F16__																																		
F17__ F18__	Semtog																																	
F19__ F4F__																																		
F50__	Regular Icons										Solid Icons																							
F51__ F53__																																		
F54__					Brand Icons																													
F55__																																		
F56__ FF3__																																		
FF4__	Voynich																																	
FF5__																																		
FF6__ FFF__																																		
	0	_1_	_2_	_3_	_4_	_5_	_6_	_7_	_8_	_9_	_A_	_B_	_C_	_D_	_E_	_F_																		
	UCSUR-Extended			Font Awesome			Voynich		(unused)																									
	UCSUR-Extended (Not implemented - No fonts available)																																	

UCSUR-Ext

Additional writing system that are part of an extended version of the Under-ConScript Unicode Registry (UCSUR-Ext) maintained by Rebecca Bettencourt. See the Kurinto_CodeChart_Aux.pdf document in the /Doc directory of the distribution package.

Font Awesome

A large collection of icon glyphs provided by the Font Awesome project.

Voynich

An implementation of the characters of the Voynich manuscript.

Plane 16 – Kurinto Supplementary Private Use Area - B

Core and Main

Roadmap for Supplementary Private Use Area - B in the Kurinto Main Variant Fonts																
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000																
10F9																
10FA	Alt Caps and Figures: Small Caps															
10FB	Alt Caps and Figures: Petite Caps															
10FC	Alt Caps and Figures: Titling Caps															
10FD	Cyrillic Small Caps				Cyrillic Petite Caps				Cyrillic Titling Caps							
10FE	Prop Lining	Prop OldStyle	Prop Hybrid	Prop Ovscore	Tabular Lining	Tabular OldStyle	Tabular Hybrid	Tabular Ovscore								
10FF	Font Ident	Visual Font Metrics														
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Kurinto		(unused)													

Aux

Roadmap for Supplementary Private Use Area - B in the Kurinto Aux Variant Fonts																
U+	__0__	__1__	__2__	__3__	__4__	__5__	__6__	__7__	__8__	__9__	__A__	__B__	__C__	__D__	__E__	__F__
1000__ 1033__																
1034__																
1035__ 1043__	Hieroglyphica - A (Kurinto Sans Aux only)															
1044__																
1045__ 1045__	Hieroglyphica - C (Kurinto Sans Aux only)															
1046__																
1047__ 104F__	Hieroglyphica - B (Kurinto Sans Aux only)															
1050__	Genji-Mon Symbols				Geo-mantic		Visible Formatting Marks			Tag Control Pictures						
1051__	Seven-Segment Figures															
1052__ 10F7__																
10F8__	Control Pictures Extended															
10F9__	Variation Selector Control Pictures															
10FA__	Alt Caps and Figures: Small Caps															
10FB__	Alt Caps and Figures: Petite Caps															
10FC__	Alt Caps and Figures: Titling Caps															
10FD__	Cyrillic Small Caps				Cyrillic Petite Caps				Cyrillic Titling Caps							
10FE__	Prop Lining	Prop OldStyle	Prop Hybrid	Prop Ovscore	Tabular Lining	Tabular OldStyle	Tabular Hybrid	Tabular Ovscore								
10FF__	Font Ident	Visual Font Metrics														
	__0__	__1__	__2__	__3__	__4__	__5__	__6__	__7__	__8__	__9__	__A__	__B__	__C__	__D__	__E__	__F__
	Hieroglyphica		Clint Developed		Kurinto		(unused)									

Hieroglyphica

A set of three adjoining blocks in the highest Unicode plane that provide 6,878 Egyptian Hieroglyphs that augment the official Unicode block of 1,071 characters in the range U+13000–U+1342F. They are derived from fonts authored by mathematician, computer scientist, and classicist George Douros, one of the leading ancient font designers.

The fonts are Abydos Regular and Bold, Version 1.960, last modified December 17, 2017. These fonts were developed to support the proposal for two additional Unicode blocks of extended Hieroglyphs.

Hieroglyphica – A is a block of 4,041 Hieroglyphs located at U+103430–U+1043FF in Kurinto fonts. It is a direct copy of the block from the Abydos region U+13430–U+143FF ... a translation in code points of $+F0000_{10}$.

Hieroglyphica – B is a block of 2,369 Hieroglyphs located at U+104680–U+104FFF in Kurinto fonts. It is a direct copy of the block from the Abydos region U+14680–U+14FFF ... a translation in code points of $+F0000_{10}$.

Hieroglyphica – C is a block of 468 auxiliary Hieroglyphs at U+104460–U+10464F in Kurinto fonts. It is a direct copy of the block from the Abydos region U+F4E60–U+F504C ... a translation in code points of $+F600_{10}$.

Frequently Asked Questions

General

Where does the name ‘Kurinto’ come from?

Kurinto (“koo-IN-toh”) is how I hear my name – “Clint” – pronounced when we visit Japan. See [About the Name](#) for more information.

Licensing

Is it free?

Yes. Kurinto is free like beer and free like liberty. Enjoy!

I want to use Kurinto fonts in my publication – can I?

Yes. Kurinto is released under the *SIL Open Font License Version 1.1* (“the OFL”), which permits use in any publication, whether electronic or printed. For more answers question about using the fonts, see FAQ-OFL.txt or visit https://scripts.sil.org/ofl-faq_web. The text of the OFL is distributed in each release package in the OFL.txt file.

I would like to bundle Kurinto fonts with my application – can I?

Yes. The *SIL Open Font License Version 1.1* allows bundling of the licensed fonts with applications, subject to some restrictions. This permission to bundle applies even for commercial, closed-source applications. See the OFL.txt and the FAQ-OFL.txt files.

Can I use the font on my web site?

Yes. You can create web pages that use Kurinto fonts to display text. This works both if that font is already available on the user's system or if it is delivered via CSS directives such as @font-face.

According to the *SIL Open Font License Version 1.1*, you may also place the Kurinto fonts on your server for people to download. There is further discussion of web-font issues in the FAQ-OFL.txt file.

Is Kurinto going to stay unrestricted and available at no cost?

There is no intention to ever charge users for using Kurinto and its variants. The current version is licensed under a free/open license and the intent is that future versions will be similarly unencumbered.

Can I send you money?

No, but thank you for the thought. Please consider supporting the Unicode Consortium at <http://unicode.org/consortium/donations.html>.

Why does the free, open-source license for Kurinto have a copyright statement?

A central concept of the Open Source movement is how it uses existing copyright law to guarantee the free and open access to and redistribution of software. This may seem counterintuitive, since copyrights have traditionally been used to restrict the rights of access and redistribution.

Kurinto is accessible to anyone, in exchange for abiding by the terms of the *OFL*. The *OFL* uses copyright law to guarantee that no party may restrict or curtail anyone else's right to copy, use, modify, and redistribute the font software. A party that violates the terms of the *OFL* (for example, by restricting the rights of others) loses their rights under the License, placing them in violation of copyright law, and opens the possibility of prosecution.



To further promote open collaboration and development, the guarantees of the *OFL* extend to the source code of the software.

Why isn't this software simply released into the Public Domain?

While it might appear that releasing software into the public domain would be a more noble approach, this path tends not to serve the community as well as the *OFL*.

- The concept of public domain is not universally recognized.
- The mechanism of making a public domain declaration is problematic in many situations, and even illegal in some countries.
- The authors cannot apply a warranty clause and open themselves to litigation by making the code available under a Public Domain declaration.
- Public domain places no restrictions on use. This freedom allows anyone to apply legal restrictions that prevent others (even the original authors) from using the software. Anyone may even claim authorship of the software, making the history of its development unclear. There are many notable cases of these perverse situations involving public domain content.
- Many commercial enterprises avoid using public domain content, because it opens **them** to subsequent litigation by others who have subsequently applied copyright and trademark restrictions.

For these reasons, I am using the well-established tradition of the *OFL* to promote continued, open development of Kurinto.

Modification

I would like to modify Kurinto to add a couple of characters I need. Can I?

Yes. Modifications are allowed, as long as you abide by the conditions of the *SIL Open Font License Version 1.1*.

Will you add glyphs upon request?

If you have a special symbol that you need (say, for a particular transcription system), the best means of doing so will be to ensure that the symbol makes it into the Unicode Standard. It is impossible for us to add every glyph that every person desires, but we do place a high priority on adding pretty much anything that falls in certain Unicode ranges (extended Latin, Greek, Cyrillic). You can send us your requests, but please understand that we are unlikely to add symbols where the user base is very small, unless they have been accepted into Unicode.

Can I send you work I've done to be incorporated into Kurinto Fonts?

Yes. See the Fontlog.txt file for information on becoming a contributor.

Metric-compatible Fonts

Will my page count change if I use one of the metric compatible font?

Possibly. There are so many variables ...

The metrics in Kurinto fonts are based on fonts in use in Windows 10 systems in early 2020 (the specific versions of fonts are given earlier in this User's Guide) and tested in Microsoft Word on Office 365. The closer you are to this setup (application, operating system, vintage), the higher the likelihood of getting exactly the same line breaks and page counts. Some users other word processing application, on other operating systems, or on older versions of Windows have found some page count differences, so test it out.

Technical

Italics and Oblique Styles

See the [Italic vs. Oblique](#) discussion.

Can you help me get Kurinto fonts working on my system?

We cannot afford to offer individual technical support. The best resource is the Kurinto.com website, where we hope to offer some limited help.

However, we do want to hear of any problems you encounter, so that we can add them to the list of bugs to fix in later releases. Our contact address is clint@goss.com. Please understand that we cannot guarantee a personal response.

I can't find all the extended Latin letters in the font. How do I type them?

Kurinto uses Unicode, which means that the computer stores a special, unique code for each letter in your document. Since most keyboards do not have hundreds of keys, special software is needed in order to type the hundreds of special characters supported by the font.

I can't find the 'o with right hook' in the font. Where is it?

Combinations of base letters with diacritics are often called “composite”, “compound”, or “pre-composed” characters. Kurinto fonts have thousands of these characters. See the section on [Locating Unicode Characters](#) to find the code points for these characters.

There are, however, many common combinations that are not represented by a single composite glyph in Unicode. It is possible to enter these into a document, but only as individual components. So, “o with right hook” would be entered as “o”, then “right hook”.

Note that the actual rendition of such a combination of characters might not be perfect. The “right hook” may not be as well placed as if the glyph was pre-composed. However, it is not possible to anticipate every possible combination.

Some diacritics are not aligning well with base glyphs, and if I type more than one diacritic, they run into each other. Why is that?

The smart diacritic positioning in Kurinto relies on OpenType. The application you are using must support this feature in order to see appropriate diacritic positioning.

How do I type the Greek letters?

You need a Unicode-compatible keyboarding system, which is not included in the release.

I'm having problems making PDFs -- why won't my document distill?

Kurinto fonts are large, with lots of glyphs. Some older printer, PDF distillers, and readers can balk at PDFs that have the complete font embedded. The easiest way to avoid this is to have the PDF distiller subset the font. This is generally a good idea anyway (with any font) and can reduce the size of your files.

How do I validate the integrity of a distribution package?

Checksums for each of distribution packages are posted on the Download page at <http://www.Kurinto.com/download.htm>. After you download the .zip file for the release package, calculate the checksums for the file. On Windows 10, you can use: [Right-click] → [CRC SHA] → [*]. Then compare the values to those shown on the Download page.

Support

As this font is distributed at no cost, I am unable to provide a commercial level of personal technical support. The font folio has been through some testing on various platforms, but these fonts do not undergo the quality assurance protocols that one would expect from a commercial digital font foundry. I primarily exercise the fonts on Windows 10 using Microsoft Office. See the section on [Testing](#) for more details.

If you do find a problem, please check the [Caveats](#) section below, and then the [Known Issues](#) section. If your problem appears in either section, know that I am aware of the issue. Otherwise, please report the issue on the GitHub Repository at <https://www.GitHub.com/ClintGoss/Kurinto>. To maximize changes of getting a fix (and minimize my hassles) **please** include as much detail as possible, including specific font names and locations, example documents, and screen captures of the issues. I cannot guarantee any direct response, but my goal is to fix reported bugs in future releases.

Many problems can be solved, or at least explained, through an understanding of the encoding and use of the fonts. Here are some basic hints:

Encoding

The fonts are encoded according to Unicode (see [What is Unicode?](#)), so the application you are using must support Unicode text in order to access letters other than the standard alphabet.

Most Windows applications provide basic Unicode support. You will, however, need some way of entering Unicode text into your document.

Caveats

This section logs general caveats regarding contemporary desktop publishing using OpenType fonts:

1. Displaying outline fonts such as Kurinto on a raster display is done by a rendering engine. There are many in use today: Uniscribe (Windows), Core Type (Apple), FreeType (open source), to name a few. As with all variable technologies, “your mileage may vary” when it comes to the actual, pixel-by-pixel, display of a given character.
2. Kurinto font do not include “hinting”, which is designed to improve the on-screen display of characters.

Known Issues

This section of the User’s Guide has been superseded by a GitHub repository for posting and discussing the development of Kurinto. Please visit:

<https://GitHub.com/ClintGoss/Kurinto>

... for the current list of known issues.

This section logs known issues with the currently release of Kurinto:

Writing Systems

The current version of Kurinto does not support the following writing systems. Unless noted, the reason is that I cannot locate an open-source font (under a license that is glyph-compatible with the OFL) that covers the writing system:

- Nyiakeng Puachue Hmong «Hmnp»: an alphabet script devised for White Hmong and Green Hmong in the 1980s by Reverend Chervang Kong for use within his United Christians Liberty Evangelical Church. Added in version 12.0 of Unicode, released March 5, 2019, in the range U+1E100–U+1E14F.
- Makasar «Maka»: historical letters used to write Makassarese language. Added in version 11.0 of Unicode, released June 5, 2018, in the range U+11EE0–U+11EFF.
- Medefaidrin «Medf»: an artificial language and script created as a Christian sacred language by an Ibibio congregation in 1930s Nigeria. Added in version 11.0 of Unicode, released June 5, 2018, in the range U+16E40–U+16E9F.
- Nandinagari «Nand»: a Brahmic script derived from Nāgarī script which appeared in the 7th century AD. Added in version 12.0 of Unicode, released March 5, 2019, in the range U+119A0–U+119FF.
- Nushu «Nshu»: a syllabic script derived from Chinese characters that was used exclusively among women in Jiangyong County in Hunan province of southern China. Added in version 10.0 of Unicode, released June 20, 2017, in the range U+1B170–U+1B2FF and U+16FE0–U+16FFF.
- SignWriting «Sgnw»: is a system of writing sign languages. Also called Sutton SignWriting. Added in version 8.0 of Unicode, released June 17, 2015, in the range U+1D800–U+1DAAF. See the section below for a discussion of the issues related to this writing system.

SignWriting

SignWriting was developed in 1974 by Valerie Sutton. It is a visually rich writing system that uses abstract pictures of the hands, face, and body. The writing system also specifies the special arrangement of characters on the page – an arrangement that does not follow a sequential order (like the letters that make up written English words). However, the rules of special arrangement are not part of the Unicode specification for this writing system.

Open-source fonts for this writing system do exist. However, they are massive. SignWriting has **very** complex rules of character formation, which are typically implemented in fonts using a system that combines characters using ligatures. Even though there are only 672 assigned code points for this writing system, the ligature systems in these fonts are extensive. One font, **SUTTONSIGNWRITING1D** version 1.1 dated June 29, 2017, by Stephen E Slevinski, contains 38,517 glyphs and 37,811 OpenType ligature specifications.

In addition, there are numerous strategies for one- and two-dimensional font designs and configurations.

Incorporating such a font into Kurinto would effectively mean that it would require its own set of font variants that contain little more than the single writing system. So, if you need to implement SignWriting, I suggest you look at the available open-source resources such as <https://github.com/Slevinski/SuttonSignWriting>.

Bassa Vah

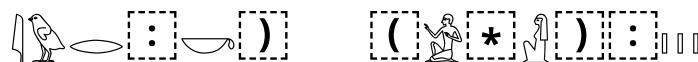
The Bassa Vah script (U+16AD0–U+16AF5) suffers from significant misplacements of diacritics and other issues. In particular:

1. The combining dot U+16AF0 is poorly placed in U+16AEA ☺, giving “☺”. So too with U+16AEA ⊕ U+16AF1: ☺.
2. There is no code point for an inverted “V”, which appears in handwritten text I have transcribed. I have used “^” in its place.
3. There is no < symbol with **two** dots. I have added U+00B7 as a substitute: <̇ .
4. The U+16AF2 combining character behaves poorly with U+16AE6. However, it does combine well with U+16AE8 – ɛ – which is probably the actual intended combination in text I have transcribed.

Egyptian Hieroglyph Format Controls

Microsoft Word does not currently render the Egyptian Hieroglyph Format Controls – U+13430–U+13438: ☐, ☐, ☐, ☐, ☐, ☐, ☐, and ☐ – which control quadrant positioning. Rather than using the format control characters to place the appropriate Hieroglyphs in quadrants, the format control characters are shown in the text.

In the left example, the third hieroglyph should appear above the fourth one, based on the intervening format control. The right example shows a more complex arrangement with three hieroglyphs composed in a two-above-one arrangement:



UCSUR Scripts

These scripts from the UCSUR (Under ConScript Unicode Registry) are not currently implement because I cannot locate suitable fonts. This list shows the name and the Kurinto Script Code:

- U_Engs – Engsvanyáli
- U_Ilia – Ilianore
- U_Syai – Syai
- U_Stre – Streich
- U_Xaîn – Xaîni
- U_Miza – Mizarian
- U_Zírí – Zírí:nka

- U_Sark – Sarkai
- U_Thel – Thelwik
- U_Nísk – Nísklôz
- U_Kaza – Kazat_?Akkorou
- U_Kazv – Kazvarad
- U_Zark – Zarkhánd
- U_Røzh – Røzhxh
- U_Seri – Serivelna
- U_Kelw – Kelwathi
- U_Sakl – Saklor
- U_Rynn – Rynnan
- U_Alze – Alzetjan
- U_Tela – Telarasso
- U_Ssûr – Ssûraki
- U_Ophi – Ophidian
- U_Fere – Ferengi
- U_Mono – Monofon
- U_Lhen – Lhenazi
- U_Niji – Niji
- U_Pikt – Pikto

Note that five more writing systems that were added to the UCSUR after August 20, 2019 were incorporated in Kurinto v2.187. These writing systems have been implemented:

- Orokin (Tennobet)
- Standard Galactic
- Braille Extended
- Cyleneian
- Syrrin

Metric Compatible Typefaces

The source fonts for all the metric compatible fonts were generally chosen from fonts that maintained that they were (or at least “reputed” to be) metrically compatible with their corresponding metrics model font. In practice, creating a fully metric compatible font is quite challenging, and many of the source fonts fell short in some areas.

Some general areas where issues arise:

- Alternate glyphs that are used as character substitutions (e.g. “1/2” to “½”) in some fonts do not have the proper character width for the target glyph.
- Kerning information may not be completely compatible.

For most uses, I expect that you will find the metric compatible fonts useful. However, they may not perfectly emulate the line and page breaks in some cases, and your document may reflow.

Here are some specific known issues with particular Kurinto typefaces:

Aria

The block of Unicode Box Drawing characters typically share the same horizontal metrics, so that they can be used to create continuous-lined geometric shapes. However, U+2502 | BOX DRAWINGS LIGHT VERTICAL in **ARIAL** version 7.00 is significantly narrower than the other characters in that block (1280 FUnits vs. 1451). Since **KURINTO ARIA** is metrically compatible with **ARIAL**, it faithfully emulates this apparent error.

Gara

KURINTO GARA uses **EB GARAMOND** as a source font, which is an exceptionally well-designed font. However, it does not perfectly align with the character metrics of the predominant implementation, the **GARAMOND** font produced by Monotype.

The side bearings of each glyph of **KURINTO GARA** have been set so that the advance width of each glyph aligns with Monotype’s **GARAMOND** font, version 2.40 (created January 2, 1996 and modified April 2, 2004). However, no changes in the contours of these characters have been made.

TRom

The advance widths of the Basic Latin characters of all weights and slants of **KURINTO TROM** match those of **TIMES NEW ROMAN** version 7.00. However, this is not true for some characters outside of the Basic Latin subset.

This means that, if your text has characters outside of Basic Latin, changing fonts from **TIMES NEW ROMAN** to **KURINTO TROM** may cause the text to reflow.

Application-Specific Issues

Here are a few issues related to specific applications. These are just the ones I know of ... I’m sure there are many more.

Final Draft

This information is based on the documentation for Courier Prime, retrieved on April 23, 2020 from <https://quoteunquoteapps.com/courierprime/faq.php>:

Final Draft for Windows uses a special version of their house font with a different line height. Your page breaks might change if you swap back and forth between **COURIER**

FINAL DRAFT and **KURINTO CNEW**. Likewise, if your writing partner is using **KURINTO CNEW** on *Final Draft for the Mac*, and you open that file on Windows, page breaks might be off.

It is a problem with *Final Draft*, not an issue with the fonts.

Note that PDFs work great, so the only hiccups come when passing around an FDX file.

Documents

PDF Files Generated by Unibook

The set of **Kurinto_CodeChart_XXX.pdf** documents provided in the /Doc directory are generated by the Unibook application (<https://unicode.org/unibook/>). While most other PDF files in the release package are created using the Adobe Print-to-PDF facility from Microsoft Word or are written directly from the perl-based fret tool, Unibook uses the Microsoft Print to PDF driver.

This arrangement causes unexpectedly large PDF files, since many elements of the file are rendered as embedded bitmaps rather than characters. The file sizes have been reduced by about 75% using Acrobat's Reduce File Size command, but this reduces the quality of those bitmaps and still leaves some rather substantially large files.

Contact

For more information, please visit the Kurinto web site at: <https://www.Kurinto.com/>.

You can access support at: <https://www.Kurinto.com/support.htm>.

Information for Developers/Contributors

This release of Kurinto under the *OFL* license provides a means for you to modify these fonts to meet your needs. It also provides a way to contribute to the project. For information on what you're allowed to change or modify, see the *OFL.txt* and *FAQ-OFL.txt* files.

Anyone can make their own modified version of Kurinto (using a different name), but I will continue to maintain and develop the canonical version of the Kurinto fonts. As the package maintainer, I welcome contributions. Here are some things to keep in mind:

Format

I am open to contributions in various formats, but if you want to maximize the chances of getting your work included in this project, please make it available to me (via email or a URL) as a *.ttf* font with TrueType outlines and OpenType tables.

Source Files

The primary source files for the fonts are the fonts themselves. They contain all the important data in the fonts and can be studied and modified using open font tools such as FontForge and TTX.

Technical Details

This section has information that is intended primarily for font developers and users with more technical background. It contains a range of topics on the font structure, internal font settings, and extended technical information. It also provides my rationale for certain design choices.

Font Settings

This section describes how various fields are set in Kurinto fonts. Rationale is provided in some cases.



UnitsPerEm

All Kurinto fonts use a `Head.UnitsPerEm` setting of 2048 (2¹¹). This was chosen based on:

- the general convention in the font design community,
- on the advice that a power of two may (still) have a performance advantage in some rendering engines, and
- to provide a reasonable level of detail when performing transformations on glyph contours (see the next section on Contour Degradation).

Contour Degradation

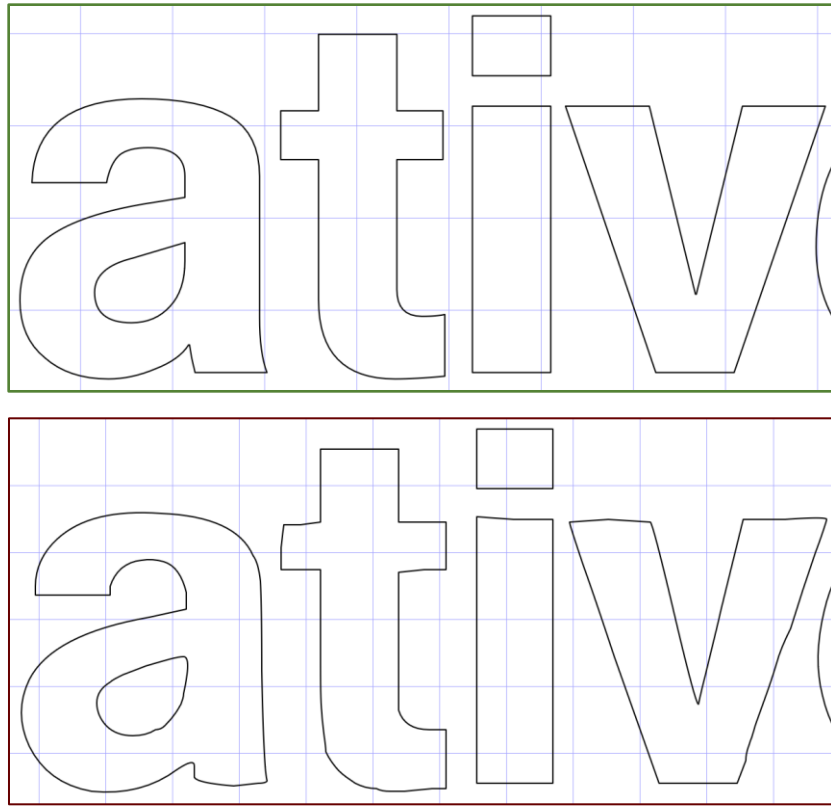
I have encountered situations where contours in glyphs have become noticeably damaged, probably due to changes in scale, alignment, rotation, or other transformations.

One extreme example of this issue is a font designed by Ryan Junell that contain icons for denoting various Creative Commons licenses. This font uses U+0043 (LATIN CAPITAL LETTER C in Unicode) for the logo (). The text of that glyph uses contours based on the Akzidenz Grotesk (and later released as **CC ACCIDENZ COMMONS** under ).

These two images below show a portion of the glyph for two different versions of the font. The images are snapshots of the glyph design mode of Font Creator 12.0.

The upper image shows the 2007 version of this font, called **CC ICONS**. This font uses a UnitsPerEm of 1400. Although the original contours of the letters were significantly reduced in size to fit in the single-character logo/glyph, the shapes of contours are preserved.

The lower image shows a 2014 version of the font, called **CREATIVE COMMONS ICONS**. The UnitsPerEm setting in this font is 1200, and the contours are significantly degraded. This may be a result of re-scaling:



Core Metrics

These settings are fixed across all Kurinto fonts:

- `Head.UnitsPerEm` = 2048

Name Table

Based on the SIL/NRSI recommendations for name table platforms⁴⁶, Kurinto fonts include name table entries for platform 3 (Windows) using the UniBMP encoding. Entries that contain line breaks use the Windows standard of the two characters CR and LF (U+000D and U+000A).

Head Table

The Flags field of the head table is set so that bits 0 and 1 are on. This indicates that the baseline for all glyphs in the font is at $y = 0$ and the left side bearing point is at $x = 0$.

The modification date of each font is set to the time that the font was generated in the [Kurinto pipeline](#). The creation date is set to Greenwich Mean Time 00:00 on the same date as the modification date.

⁴⁶ See https://silnrsi.github.io/FDBP/en-US/Font_Naming.html#about-name-strings-stored-in-opentype-fonts.

OS/2 Table

The achVendID field is set to “Goss”, for “Goss Typography”, a [registered foundry](#).

The Sample Text field is taken from Genesis 11:1 (*New International Version Bible*). In early versions of Kurinto, the sample text included English, Greek, Hindi, Cherokee, Japanese, and Malayalam:

Now the whole world had one language and a common speech.

καὶ ἦν πᾶσα ἡ γῆ χειλὸς ἓν καὶ φωνὴ μία πᾶσιν.

बाढ़ के बाद सारा संसार एक ही भाषा बोलता था। सभी लोग एक ही शब्द समूह का प्रयोग करते थे।

Dd RWhE ၵၵၵၵ ၵ4 Ddၵၵၵၵ, Dd ၵၵၵၵ ၵ4 Ddၵၵၵၵ.

初めのころ、人類はみな同じことばを話していました。

ഭൂമിയില് ഒക്കെയും ഒരേ ഭാഷയും ഒരേ വാക്കും ആയിരുന്നു.

However, this collection of languages caused instability in a number of applications. Sadly, I have contracted the sample text to simply:

Now the whole world had one language and a common speech.

DSIG Table

In addition to providing digital signatures in a few fonts, this table served to flag the font as having OpenType tables in Microsoft Word 2013 and before. Fonts without this table would not be recognized by those earlier version of Word as supporting OpenType features such as ligatures, alternate forms of figures (Lining, Old-style, Proportional, Tabular), or stylistic sets (that Kurinto uses for Small Caps, Petite Caps, etc.), even if the font has a GSUB table.

For this reason, the *fontbakery* testing tool used by Kurinto still flags FAIL for fonts without a DSIG table.

This shortcoming has been corrected in Microsoft Word 2016 and later. However, since Microsoft Word 2013 is still under extended support through May 2023, all Kurinto fonts provide a vestigial DSIG table to allow users of Word 2013 to access these features.

See <https://github.com/googlefonts/fontbakery/issues/1845> for an extended discussion of these issues.

Font Names

Long font names can raise issues in some application, especially *Microsoft Word*. These issues are described in <https://github.com/googlefonts/noto-fonts/issues/1566>, which reads in part:

Fonts with a family name that is too long cannot be selected on MS-WORD. The cause is that Name ID 1 is not unique within 31 characters. For example, typefaces that

shorten "Condensed" to "Cond" etc. can be used without problems with MS-WORD. My ideal is that all font's Name ID 1 fits within 31 characters. If font name ID 1 is 32 characters or more, even if the font can be used, it is not displayed correctly.

Because of these issues, the Font Family Name ('name' table ID 1) are no more than 28 characters long and the Full Font Name ('name' table ID 4) of all Kurinto fonts are unique even if truncated to the first 30 characters.

Range Bits

One holdover from the pre-Unicode era lives inside OpenType files in the form of the 64 Code Page Bits in the fields `OS/2.ulCodePageRange1` and `OS/2.ulCodePageRange2`. Some applications (including Microsoft Word as of February 2020) still use these bits to control font selection.

Each of the Code Page Range bits for Kurinto fonts is set if the font maps at least 70% of the code points for that range, based on the code page mappings files provided on Unicode.org as of Unicode version 12.1 (<https://www.unicode.org/Public/MAPPINGS/>).

OpenType files also have a set of `OS/2.ulUnicodeRange...` bits that indicate the presence of characters in various Unicode blocks. The Unicode Range bits are set based on the presence of any mapped code point in the corresponding Unicode block.

Glyph Names

Fonts in the release packages have glyph names based on this priority list:

- The name for the glyph in the Adobe Glyph List (the "AGL"). The AGL contains specific glyph names for 586 code points specified in the `aglfn.txt` file retrieved from <https://github.com/adobe-type-tools/agl-aglfn> on April 10, 2019.
- For glyphs mapped to a code point, a standard name composed from the lowest mapped code point for the glyph. The standard name has the format "uniXXXX" for code points U+FFFF and below and "uXXXXX" for higher code points. The XXXX is the code point in hexadecimal.
- For unmapped glyphs, the name "_NNNN" where NNNN is the index of the glyph (often called the "glyph ID") in decimal.

Variant Identifiers

Note that the variant identifiers I use – HK, SC, KM, etc. – follow no particular standard. This section contains some notes on alternate identifiers:

HK The language identifier for this language variant are variously "HK", "HC" (Hong Kong Chinese – used in Source Han Mono), and "ZHH" (OpenType language code).

- JP** The language identifiers for this language variant are variously “JP” and “JAN” (OpenType language code). The variant of the Source Han Mono that contains Japanese is simply missing the identifier – Japanese is the language default.
- KR** The language identifiers for this language variant are variously “KR”, “K” (used in Source Han Mono), “KOR” (OpenType language code).
- SC** The language identifiers for this language variant are variously “SC” (Kurinto and also used in Source Han Mono) and “ZHS” (OpenType language code).
- TC** The language identifiers for this language variant are variously “TC” for either “Traditional Chinese” or “Taiwanese Chinese”, “TW” (used by some fonts), and “ZHT” (OpenType language code).

Script Codes

The **Map.txt** in each release package provides a roster of the writing systems (scripts) that make up each Kurinto font. It also lists which specific source fonts contributed each of those writing systems.

Each writing system is identified by a script code. These script codes are specific to Kurinto and, while they are related to the script codes used by Unicode, there are significant differences. Kurinto adds script codes to handle the division of Unicode scripts along block boundaries, the subset of [Core Character Set](#), divisions across Kurinto font variants, allocation of non-Unicode blocks for auxiliary writing systems, and for convenience when composing fonts from various sources.

The script codes used in Kurinto are listed in the **Scripts.txt** file in each release package.

Kurinto script codes are organized into:

Unicode Scripts and Sub-scripts

Unicode Script: A script defined in Unicode. Note that many of these scripts have been segmented so that some (or all) of the characters have been relocated to other scripts.

Sub-script: A script composed of characters extracted from Unicode scripts for the purposed described above.

Common Sub-script: A sub-script made up of characters from the **Zyyy** (Common) and **Zinh** (Inherited) Unicode scripts.

WGL4 Sub-script: A sub-script containing characters from the *Windows Glyph List 4* character set. These characters are part of the Core Character Set that are present in all Core variant fonts.

NRSI Sub-script: A sub-script containing characters from the core set recommended by NRSI. These characters are part of the Core Character Set that are present in all Core variant fonts.

Google Sub-script: A sub-script containing characters from the core set recommended by Google. These characters are part of the Core Character Set that are present in all Core variant fonts.

Non-Unicode Scripts

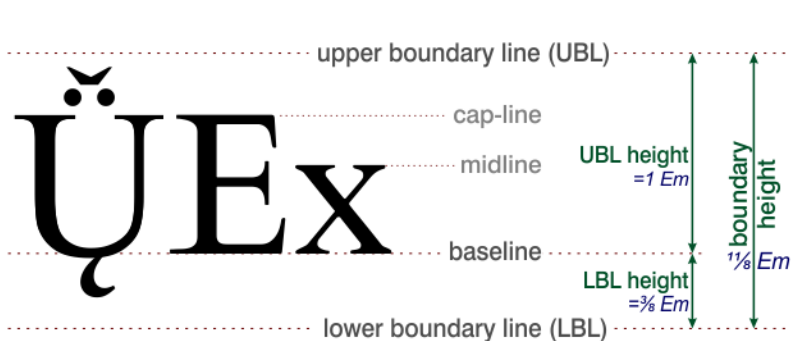
UCSUR Script: Blocks of code points allocated by the Under-ConScript Unicode Registry. These code points are all within the Private Use Areas set aside by the Unicode specification. Note the list of [USCUR scripts that are not present in the current version of Kurinto](#).

Kurinto PUA Script: Other blocks of code points allocated in the Private Use Area in Kurinto fonts.

Vertical Metrics Revisited

Settings for the eight OpenType vertical metrics parameters that control line height have caused much debate and heartburn among font designers. They have also been the source of untold issues with authors trying to control line spacing in their documents (see the [Surprise Layout Changes Pitfall](#)).

This section describes my approach and rationale for how the Kurinto line height parameters are set. It also looks at the related vertical metrics for cap-height, x-height, underlining, and strikethrough.



Boundary Lines

I have found it helpful to introduce two virtual design metrics: the Upper Boundary Line (UBL) and the Lower Boundary Line (LBL).

These lines – together with the corresponding UBL height and LBL height measurements – form the basis for how

word processors set interline spacing.

The expectation is that a word process will set the interline spacing to the boundary height for single-spaced text. In this scenario, the UBL of one line would coincide with the LBL of the previous line of text.

All Kurinto fonts set the UBL height to equal 1em, placing the upper boundary line 1em above the baseline. The LBL height is set at $\frac{3}{8}$ em. The resulting boundary height is $1\frac{1}{8}$ em. So, for 12pt text, the UBL is 12pt above the baseline, the LBL is $\frac{3}{8} \times 12\text{pt} = 4.5\text{pt}$ below the baseline, and the total boundary height is 16.5 pt. You can expect the line spacing of text set at 12pt to be 16.5pt ($\approx 0.23''$).

Since the UnitsPerEm of all Kurinto fonts is 2048, the UBL height is +2048 ($=2^{11}$) and the LBL height is 768 ($=3 \times 2^8$). The boundary height is fixed at 2816 – or 137.5% – of the UnitsPerEm setting.

This approach has some advantages:

- A boundary height of 137.5% fits nicely with various guidelines⁴⁷ which recommend a line height of 120%–145% of UnitsPerEm for contemporary fonts.
- Since these metrics are set the same across all fonts in the folio, switching fonts within the folio does not change the line height in any application I have tested.
- Use of powers of two may (still) have some performance advantages in some rendering engines.

Setting the Interline Spacing OpenType Parameters

Kurinto uses the approach recommended by SIL International for setting the eight OpenType vertical metrics parameters.⁴⁸ These settings are used across all the fonts in the folio:

- OS/2.sTypoAscender = OS/2.usWinAscent = hhea.Ascender = [UBL height](#) = 2048.
- OS/2.sTypoDescender = hhea.Descender = – [LBL height](#) = –768.
- OS/2.usWinDescent = [LBL height](#) = 768 (usWinDescent must be positive⁴⁹).
- OS/2.sTypoLineGap = hhea.LineGap = 0

All the applications I have tested render consistent line spacing with these settings.

However, setting the vertical metrics to fixed values does have some downsides. There are instances in non-Latin writing systems where the glyph bounding box extends above the UBL or below the LBL (i.e. has a $Y > 2048$ or $Y < -768$). In my experience, on-screen display in a few application settings do clip the glyph, but print and digital document renderings show no clipping. This issues was explored in detail in the [Clipping](#) section.

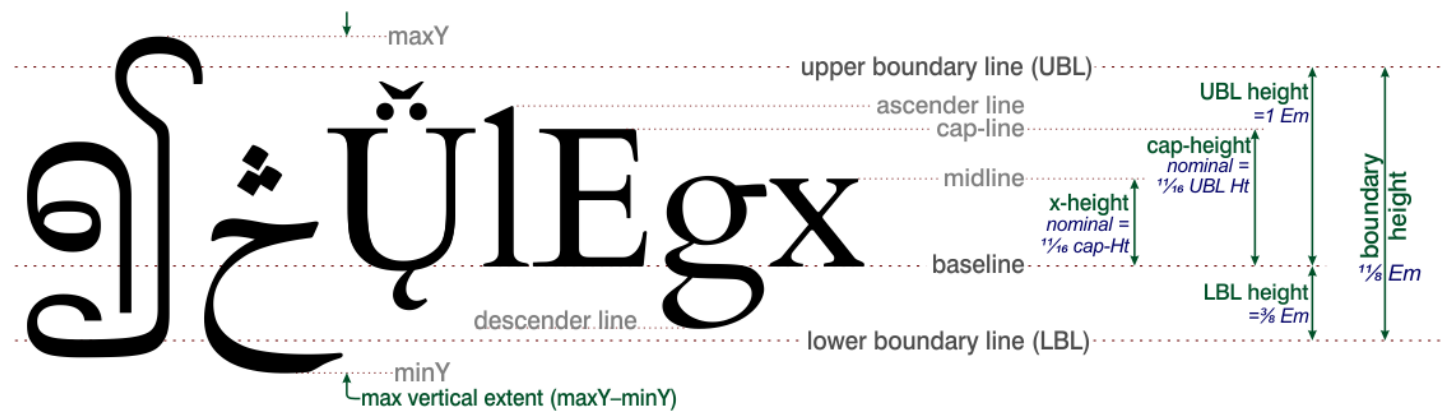
⁴⁷ For example, *Butterick’s Practical Typography, Version 2* at <https://practicaltypography.com/line-spacing.html> and also the discussion at <http://Typophile.com/node/77906>.

⁴⁸ See SIL International, *Font Development Best Practices*, section 3.4 *Line Metrics*, available at [https://silnrsi.github.io/FDBP/en-US/Line Metrics.html](https://silnrsi.github.io/FDBP/en-US/Line%20Metrics.html).

⁴⁹ The usWinDescent is given as a distance from the baseline and must be positive. According to <https://docs.microsoft.com/en-us/typography/opentype/spec/os2#uswindescent>:

“... the usWinDescent value treats distances below the baseline as positive values; thus, usWinDescent is usually a positive value, while sTypoDescender and hhea.descender are usually negative.”

Setting the Other Vertical Metrics



This diagram shows how the additional line metrics are set.

Note that the cap-height and x-height parameters should be considered **nominal** values. These are general guidelines across all Kurinto fonts. The actual OpenType cap-height and x-height values are set on for each typeface.

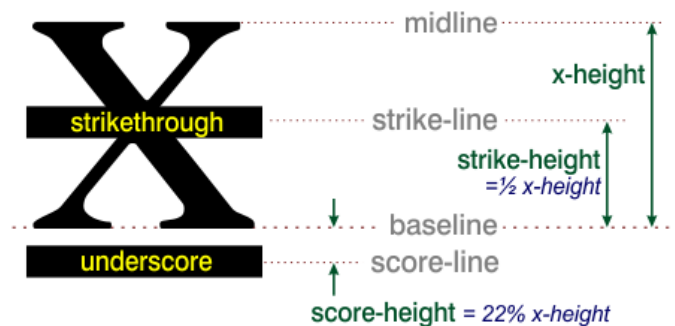
The nominal cap-height is set to $\frac{11}{16}$ em and the nominal x-height is set to $\frac{11}{16}$ cap-height.

Underlining and Strikethrough Revisited

The [Underlining and Strikethrough](#) section gave a general overview of how underlining and strikethrough are handled. This section describes the specifics.

The diagram at the right shows a lower-case “x” with both strikethrough and underscore set. I use two virtual metrics to control these adornments:

- The **strike-line** is located at the vertical **midpoint** of a strikethrough adornment. The strike-height parameter is the distance of the strike-line from the baseline.
- The **score-line** is located at the vertical **midpoint** of the underline (or “underscore”) adornment. The score-height is the distance of the score-line from the baseline.



The OpenType parameters for thickness are set to a single value that is based on the visual weight of the font. That value is then used to set the two OpenType parameters that control the vertical position of the strikethrough and underscore.

The behavior we might expect is that a word processing application would honor all four OpenType parameters – underline thickness, underline position, strikethrough thickness, and strikethrough

position – for each section of text on a line. *Microsoft Word* comes close to this ideal. My experience with *Word*, based on testing on Office 365 under Windows 10 on April 4, 2020 (and demonstrated in the samples above) is that:

- *Word* implements ~~striketrough~~ as expected, keeping the center of the adornment aligned across fonts of different typefaces ~~and styles but~~ varying the thickness.
- *Word* implements ~~double striketrough~~ based on the ~~single striketrough~~ parameters, using a vertical “sandwich” of three zones (solid, clear, solid), each of which is the same height as the single striketrough and with all of them centered on the striketrough height.
- *Word* implements underline using a complex algorithm: It uses a fixed underline thickness and vertical position across an entire line. *Word* seems to calculate the underline dimensions based on the lowest underline position of any font on the line.
- *Word* implements other underscoring adornments such as double underline, wavy line, and various Morse code styles based on the underlining parameters.

These rules do seem reasonable and produce acceptable typesetting in most situations.

However, other applications have mixed results with their implementation of the OpenType Standard. In tests published by Henrique Beier in July 2018,⁵⁰ Adobe Photoshop, CorelDraw X8, Google Chrome, Opera, and Safari ignored the OpenType settings, Adobe InDesign and Illustrator had partial implementations, and the Firefox and Edge browsers had full implementations.

OpenType Layout Features

The OpenType specification provides for an extensive array of tool for altering the layout and appearance of characters rendered from fonts. These are used to:

- automatically alter the layout of text, according to the rules and conventions of the writing system, and
- alter text by changing the layout or substituting glyphs on a discretionary basis – under the control of the user.

Kurinto fonts are composited from set of source fonts. A subset of the characters and glyphs of each source fonts is produced, and those subsets are merged to produce the specific Kurinto font.

The philosophy of Kurinto is to pass all OpenType directives from the source fonts through to the resulting Kurinto font. The subset operation preserves all OpenType directives appropriate to the particular subset being performed, together with all alternate glyphs needed for those directives.

However ...

⁵⁰ See *The State of Underlines and Strikethroughs* at <https://www.harbortype.com/blog/the-state-of-underlines-and-strikethroughs/>.

The primary authoring tool that is targeted by Kurinto is *Microsoft Word*. The current implementation of *Microsoft Word* implements only a small portion of the discretionary OpenType features that are defined. Despite the wide adoption by the user and industry communities of standard that has been in place for over two decades (since it was introduced by **Microsoft** in 1996⁵¹), users of *Word* are restricted to only a few basic OpenType features.

I have jury-rigged the fonts to use the 20 Supplementary Style choices to implement what I felt were the most pressing OpenType features – Small Caps, Petite Caps, Hybrid Figures, etc. (see [Alternate Characters and Layout Features](#)), but this is a poor substitute for a full OpenType implementation.

Hopefully, the good folks in Redmond will heed the needs of their users.

If you use Kurinto fonts with an authoring tool that allows full access to OpenType features, all the features present in the original source fonts should be available (although testing in this area has not been done). This means that many of the fonts have a massive collection of OpenType layout features, since they contain the union of all the directives for the selected writing systems from the source fonts. This is especially for the Main and Aux font variants.

Despite the size and complexity of the OpenType tables in some of the Kurinto fonts, I have rarely encountered any significant performance issues in the applications that I use. The one exception is when opening a large font in a font editing application, such as FontCreator. The parsing of the large OpenType tables can take a minute or three when opening some of the fonts – far slower than even the CJK fonts, which are many times larger but have far fewer OpenType directives.

Color

The introduction of emoticons in Unicode 6.0 (2010) spurred the introduction of color glyphs into the OpenType specification. Independent designs were created by Adobe, Apple, Google, and Microsoft in subsequent years, and all those designs have been incorporated into the OpenType specification as of version 1.8 (2016).

Kurinto fonts that use color are implemented with the Microsoft system that uses the COLR and CPAL SFNT tables. The SFNT tables used in the Adobe (SVG), Apple (sbix), and Google (CBDT and CBLT) implementations do not appear in Kurinto fonts.

Notes on Unicode Blocks and Writing Systems

This section has specific details on the implementation of various ranges of Unicode characters.

⁵¹ See <https://practicaltypography.com/opentype-features.html>.

Box Drawing and Block Elements

These Unicode blocks in the ranges U+2500–U+257F and U+2580–U+259F have legacy characters that provide compatibility with historical computing systems such as the cell graphics of the IBM PC. They are provided “for backward compatibility with the existing standards” and “the Unicode Standard does not encourage this kind of character-cell-based graphics model”.⁵²

All characters in these blocks are “designed to connect together into continuous lines, with no gaps between them”. As such, each of these characters “extend to the middle of the top, bottom, left, and/or right of the bounding box for the character cell”. They extend the full height of the boundary height extending from the LBL to the UBL. The width of these characters depends on the typeface:

- For monospaced and multi-spaced fonts (**KURINTO MONO**, **CMOD**, and **CNEW**), the width of these characters is the same as U+0020 SPACE.
- For **KURINTO TEXT**, these characters are designed on a square grid (W:H ratio = 1:1). In a rendering environment that maps to square pixels, these should render as square graphics.
- For all other typefaces, these characters are designed on a grid that is half as wide as it is high (W:H ratio = 1:2). Rendered graphics will appear squashed horizontally by 50%, compared with the **KURINTO TEXT**, rendering. However, the width of the lines themselves is preserved – it is only the overall dimensions of the graphics that are horizontally squashed.

Note, unlike many fonts that render all of the characters in these two blocks identically across the **RBIBI** styles, Kurinto fonts provide italicized versions of all these characters and bold versions of the Box Drawing characters.

The Pipeline

Versions of Kurinto are constructed in a pipeline that is controlled by a set of perl scripts. As of March 2020, a full build of the fonts take about 14 hours on a 10-core, Intel Core i9-9900X processor at 3.5GHz.

Very generally, the pipeline involves these technologies:



Microsoft Excel workbooks, for tables of information relating to Unicode (characters, glyph names, blocks, scripts), settings to be applied (e.g. the static contents of TrueType information tables), license management, and general control of the rest of the pipeline (e.g. the sequence of subset, merge, and transformations to be performed. This document contains some worksheets that are linked to those .xlsx documents.

⁵² All quotes in this section and design choices made for these two Unicode blocks are from [\[Unicode 2019\]](#), page 845.



Perl tools. The majority of work in running the pipeline, transforming fonts, producing releases, rights-management, database management, and various housekeeping functions is done by custom perl scripts I developed, using the Perl interpreter packaged by ActiveState. These tools are based on the open source FontUtils (Font : : TTF) library developed initially by SIL International (<https://scripts.sil.org/FontUtils>) and augmented by many contributors.



Python tools. Some use is made of the Python-based FontTools package (<https://github.com/fonttools/fonttools>), in particular the *merge* utility developed by Behdad Esfahbod and Roozbeh Pournader.



FontCreator. My main tool for font authoring and examination is the commercial FontCreator Pro application by High-Logic, developed by Erwin Denissen (<https://www.high-logic.com/>).



Microsoft Word.

Used for general documentation, development of the Specimen Book, and testing.



Corel Draw.

Used for authoring document that require extensive formatting, diagrams, and font control that is beyond the scope of *Word*.



FreeType. An open source software library to render outline fonts on raster displays. This is used by Font Validator. <http://freetype.org/>.

Font Validator. An open-source tool sponsored by Microsoft for verifying glyph consistency within a font. See <https://github.com/microsoft/Font-Validator>.

Font Bakery. An open-source tool sponsored by Google for reporting various consistency checks on fonts. See <https://github.com/googlefonts/fontbakery>.

While Kurinto is an open-source project in the sense that derivative work can be developed based on the fonts, I have not seen a clear way to release the font-production pipeline itself.

I believe that the complexity of the pipeline structure is too varied for me to reliably document (and support) how other developers might re-create the pipeline environment on their own systems with a likelihood of success.

CMAP Tables

Kurinto provides two cmap subtables in every font: Unicode BMP and Unicode FULL. Specifically:

- ‘cmap’ subtable 0 is configured for Windows (platform ID 3), language ID = 0, and Unicode BMP encoding (Basic Multi-Lingual Plane, restricted to U+0020–U+FFFF, encoding ID 1).
- ‘cmap’ subtable 1 is also configured for the Windows platform and language ID 0, but uses the “Unicode full repertoire” encoding (U+0020–U+10FFFF, encoding ID 10).

The Kurinto pipeline is configured to omit ‘cmap’ table 1 if all the characters in the generated font lie within the BMP. However, since all Kurinto fonts include the [block of characters in Plane 16](#), all fonts in the release package have two cmap subtables.

The format the ‘cmap’ subtables depends on the specific contents. However, it is typically Segmented Mapping (format 4) for subtable 0 and Segmented Coverage (format 12) for subtable 1.

CMAP Table Format

In order to address the pitfalls of [Font Selection](#), my initial intent was to provide a glyph for every code point in every font, using (highly multi-mapped) fallback glyphs that indicated the font variant to use to obtain the real glyph for that code point.

Although it is unusual for a font to map every code point, this approach places control back into the hands of the author. The distinctive fallback characters would appear in the document to flag the “no glyph available” situation to the author, who can make a conscious choice about which font to use. This feature would only require a handful of glyphs in each font, and (I believe) completely address the set of [Font Selection](#) pitfalls.

However ...

There is no cmap table format in the TrueType specification that serves this goal:

- The format 12 table – the standard for Unicode fonts – balloons linearly with the number of assigned code points, and would require entries for approximately each of the potential Unicode code points (1,114,112) in every font.
- Format 13 tables – intended for “Last Resort” fonts – use code point ranges for **every** glyph. This places a substantial overhead on each code point that is mapped to a distinct glyph in the font.

What is needed is a cmap table format that combines the features of format 12 and 13 tables. This new format table could use a flag field to indicate whether the code point range should be mapped onto a single glyph (as in format 13 tables) or a sequence of glyphs (format 12).

I personally believe that the strategy of having fonts assign every code point is valid in situations where you want to place the decisions of font selection for non-standard code points back into the hands of the author, rather than silently changing the font to one that the application deems “best”.

In place of the “glyph for every code point” approach, I now use a strategy for setting Panose values described in the next section.

Panose Settings

In order to address the pitfalls of [Font Selection](#), Kurinto fonts have Panose settings that emphatically nudge *Word* to use the appropriate font variant for a missing code point.

The Panose codes used in Kurinto are listed in the **Panose.txt** file in each release package.

Hinting

The current version of the Kurinto pipeline passes the original hinting information of the source fonts along to the target fonts.

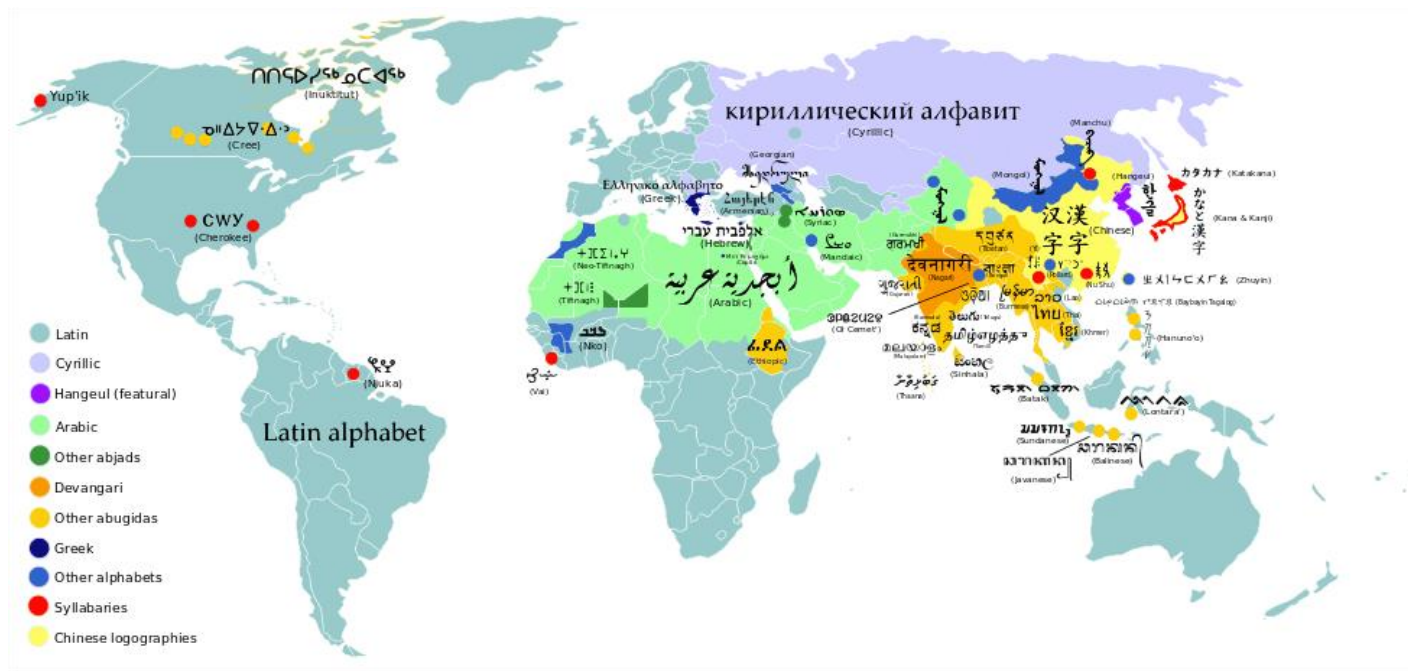
In the future, use of automated (re-)hinting process such as *ttfautohint* may be used.

Sample Texts

Throughout the documentation and web site for Kurinto, I use a set of sample texts demonstration of paragraph text. I also use specific representative characters to highlight particular aspects from various writing systems. In particular, the *Kurinto Specimen Book* (*Kurinto_SpecimenBook.pdf*, included in each release package) is composed mostly of sample texts and representative characters.

This section describes the sources and rationale for the sample texts and representative characters.

Predominant Writing Systems



World Writing Systems

The sample texts are chosen so that they cover the predominant writing systems in use today.⁵³ Each of these writing systems has more than 100 million users:

- **Latin.** This script is the basis for the largest number of alphabets of any writing system and the most widely adopted in the world (commonly used by about 70 percent of the world's population).⁵⁴
- **Chinese / 汉字漢字.** The Unicode term for this writing system is “Han”. It is used by an estimated 1.3 billion people.
- **Devanagari / देवनागरी.** Used by many languages on the Indian sub-continent, including Hindi. Sometimes abbreviated to “Nagari”.
- **Arabic / العربية.** Arabic script is a writing system used for writing Arabic and several other languages of Asia and Africa, such as Persian, Sorani Kurdish, Azerbaijani, Sindhi, Balochi, Pashto, Lurish, Urdu, and Mandinka.
- **Bengali-Assamese.** Bengali–Assamese script (also Eastern Nagari script) is the basis of the Bengali, Assamese, and Tirhuta alphabets as well as the alphabets for Bishnupriya Manipuri, Kokborok (Tripuri) and Meitei (Manipuri), Chakma, Santali, etc. The Unicode term for this writing system is “Bengali”.

⁵³ This information and the image of the *World Writing Systems* was obtained from Wikipedia’s *List of Writing Systems*, retrieved on February 8, 2020 from https://en.wikipedia.org/wiki/List_of_writing_systems.

⁵⁴ See Harald Haarmann. *Geschichte der Schrift* [History of Writing] (2nd ed., 2004). München: C. H. Beck. German. See also https://en.wikipedia.org/wiki/Latin_script.

- **Cyrillic / Кирилица.** Cyrillic script is used for various languages across Eurasia and is used as the national script in various Slavic-, Turkic- and Iranic-speaking countries in Eastern Europe, the Caucasus, Central Asia, and Northern Asia.
- **Kana / かなカナ.** Includes the Katakana and Hiragana writing systems used in Japanese, used by about 120 million native speakers with close to a 100% literacy rate.

Interestingly, these seven scripts span all the main flavors of writing systems (alphabets, logographics, abjads, abugidas, and syllabaries).

The sample texts are written in one or more widely used languages of each writing system. Since the example texts are written in a *language*, not a writing system, using English for Latin script is not sufficient. Since English uses an alphabet of only 26 letters, compared to the hundreds of letters in the Latin writing system, I use examples in four additional languages to get a better coverage of the characters in the Latin writing system:

- Icelandic / íslenska,
- Danish / dansk,
- Vietnamese / Tiếng Việt, and
- Turkmen / Türkmençe (Turkmenistan and Northern Iran).

Samples of the seven writing systems are provided in (at least) these twelve languages: English, Icelandic, Danish, Vietnamese, Turkmen, Traditional Chinese, Simplified Chinese, Hindi, Standard Arabic, Bengali, Russian, and Japanese.

“Kurinto”

For very brief examples, I use the word “Kurinto” transcribed (as best I can) for the writing system and language. The vertical word-cloud for “Kurinto” at the right uses this concept, in a smattering of languages and writing systems.

For the seven predominant writing systems and their representative written languages, I use the transcriptions:

- Kurinto (Latin / English)⁵⁵
- 库里特 (Han / Simplified Chinese ≈ Kù lǐ tè)
- 庫林托 (Han / Traditional Chinese ≈ “Kù lín tuō”)
- 库林托 (Han / Simplified Chinese ≈ “Kù lín tuō”)
- कूरेंटो (Devanagari / Hindi ≈ “koorento”)
- كورينتو (Standard Arabic ≈ “kurintu”)
- কুড়তে (Bengali-Assamese / Bangla ≈ “Kuraṭē”)
- КУРИНТО (Cyrillic / Russian ≈ “Kurynto”)
- クリント (Katakana / Japanese ≈ “Kurinto”)
- くりんと (Hiragana / Japanese ≈ “Kurinto”)

The above transcriptions were done with the assistance of Google Translate as of February 11 and December 17, 2018.

Additional writing systems and languages, sorted alphabetically by language, include:

- ካሩቶ (Ethiopic (Ge’ez) / Amharic)
- Ⴃၢၢၢ (UCAS / Blackfoot)
- ⠠⠠⠠⠠⠠⠠⠠⠠ (Braille)
- КУРИНТО (Cyrillic / Bulgarian)
- ကရိတ (Burmese)
- ᎠᎹᎠᎵ (Cherokee)
- ⲕϣⲣⲓⲛⲧⲟ (Coptic)
- ᖃᐅᐭᐅ (UCAS / Déné)
- კურინტო (Georgian)
- Κούριντο (Greek / Modern Greek)

⁵⁵ Using the extended Latin characters, this could be written Kürřntō (based on the ToPhonetics.com pronunciation guide). We could also imagine more whimsical spellings such as Ḳŭřĩņtŏ and Ḳŭřĩņtŏ that were done simply to include characters from various extended Latin Unicode blocks.

कूरेंटो
কুড়তে
कुरिंतो
वुरीन्टे
庫里特
쿠린토
クリント
КУРИНТО
Kurinto
Κούριντο
קוריןטו
קוריןטו
کورینتو
کتنا
ካሩቶ
કુરિન્ટો
કુરિનાહો
કુરિન્હો
კურინტო
കുരിന്തോ
කුරින්ටෝ
கேயுரிண்டோ

- [illegible]

Genesis

Various versions of Genesis 11:1:

1. Now the whole world had one language and a common speech.
2. Öll jörðin hafði eitt tungumál og ein og sömu orð.
3. Hele Menneskeheden havde eet Tungemål og samme Sprog.
4. Lúc đó cả thế giới chỉ có một ngôn ngữ, mọi người đều dùng một thứ tiếng mà thôi.

5. Tutuş ýer ýüzünde bir dil, birmeňzeş sözler bardy.
6. 那時，天下人都用同一種語言，講同一種話。
7. 那时，天下人的口音、言语都是一样。
8. अब सारी पृथ्वी पर एक ही भाषा, और एक ही बोली थी।
9. وَلَمْ يَكُنْ فِي الْأَرْضِ إِلَّا لُغَةٌ وَاحِدَةٌ لَهَا مُفْرَدَاتٌ مُحْدَوْدَةٌ.
10. Xপ্লাবনের পরে সমস্ত পৃথিবী এক ভাষাতে কথা বলত| সমস্ত মানুষ একই শব্দগুলি ব্যবহার করত|
11. Во всем мире был один язык и одно наречие.
12. 初めのころ、人類はみな同じことばを話していました。
13. ഭൂമിയില് ഒക്കെയും ഒരേ ഭാഷയും ഒരേ വാക്കും ആയിരുന്നു
14. καὶ ἦν πᾶσα ἡ γῆ χεῖλος ἓν καὶ φωνὴ μία πᾶσιν.
15. Dḍ RWhE ԵՎՎԷ Ի4 DhԺhԹE, Dḍ OՅԺՎԷ Ի4 DhԼET.

The corresponding languages and sources for the above translations are:

1. Latin / English: From the *New International Version Bible*.
2. Latin / Icelandic: Icelandic Bible, by the Icelandic Bible Society.
3. Latin / Danish: Danish, Det Gamle Testamente af 1931. Det Danske Bibelselskab (The Danish Bible Society).
4. Latin / Vietnamese: Vietnamese Bible: Easy-to-Read Version (BPT), 2010, World Bible Translation Center, BibleGateway.com
5. Latin / Turkmen: Onmigit.com, sourced from <http://ibt.org.ru/english/bible/tkm.htm>.
6. Han / Traditional Chinese: CCBT – Chinese Contemporary Bible (Traditional), courtesy of BibleGateway.
7. Han / Simplified Chinese: CUVS – Chinese Bible Union Version (Simplified), courtesy of BibleGateway.
8. Devanagari / Hindi. Translation by Nazish Qamar, Omniglot.com.
9. Arabic: Arabic Bible: Easy-to-Read Version, 2009, World Bible Translation Center.
10. Bengali: Omniglot.com.
11. Cyrillic / Russian: Holy Bible, New Russian Translation (НОВЫЙ ПЕРЕВОД НА РУССКИЙ ЯЗЫК) Copyright © 2006 by Biblica, Inc.® Used by permission.
12. Kana / Japanese: Japanese Living Bible, Copyright© 1978, 2011, 2016 by Biblica, Inc.® Used by permission.
13. Malayalam: Omniglot.com.

15. Cherokee: Edwin Archer, Cherokee Bible, Park Hill, 1856, courtesy of the Cherokee Bible Project.

[illegible]

10. Cyrillic / Russian: Clagnut. In English: “An enraged narrator selfishly beats a nimble fencer with five poles.”
11. Kana / Japanese: The Iroha (いろは) is an ancient Japanese poem that forms a perfect pangram. It contains each ancient Hiragana character exactly once (including the archaic ゐ and ゑ characters). An English-language translation by Ryuichi Abe: “Although its scent still lingers on, the form of a flower has scattered away. For whom will the glory of this world remain unchanged? Arriving today at the yonder side of the deep mountains of evanescent existence, We shall never allow ourselves to drift away, intoxicated, in the world of shallow dreams.” See <https://en.wikipedia.org/wiki/Iroha>.
12. Hebrew: Clagnut. All 22 in the Hebrew alphabet with all medial and final forms. In English: “A ‘dust bat’ escaped through the air conditioner, which exploded due to the heat.”
13. Klingon: Clagnut. In English: “Because of your apparent audacity the depressed conqueror is willing to fight you.”
14. Hangul / Korean: A Korean phrase that contains all 14 simple consonant letters, all 6 simple vowel letters, and all 4 iotized vowel letters, along with 1 of the 5 double consonant letters (ㄲ “gg”), 1 of 11 consonant clusters (ㄴㅎ “nh”), and 1 of 11 diphthongs (ㅟ “ui”). Clagnut. In English: “The essential condition for a kiss is that lips meet and there is no special technique required.”
15. Myanmar: Clagnut. In English: “The genius from Sri Lanka read the formula of elixir of life thoroughly in the almond tree next to Zalun market.”
16. Thai: From the Computer Association of Thailand under the Royal Patronage of His Majesty the King. Clagnut. In English: “Humans are most superb and worth more than any animal or beast. Do develop your academic expertise. Do not destroy or kill anyone. Do not be angry or execrate anyone. Practice forgiveness as you would good sportsmanship. Do behave under morals and rules. Speak and confer politely and with servility.”
17. Tibetan: Clagnut.

The UDHR

Article 1 of the *Universal Declaration of Human Rights* (the “UDHR”) provides another set of sample text. The status of the translations of this document are particularly interesting, in the world of omnilingual texts. From [Kellman 2016]:

It can be read ... in languages ranging from Abkhaz to Zulu. However, these versions are not conceived as translations but rather as equivalences, alternate embodiments of identical tenets. The Bible has, according to the Wycliffe Global Alliance, been translated in part into 2,932 languages, as a whole into 554 (Wycliffe 2015). However, in the case of the Bible, unlike the UDHR, it is meaningful to distinguish between the original and its derivatives. The Hebrew and Greek texts possess authority that English, Bengali, and Xhosa approximations do not. Nevertheless, although the Bible is

translated, the UDHR is, through the theology of international governance, transubstantiated into multiple tongues. No version has priority; none is the Ur-text. In principle, each is equally valid, transparent, and interchangeable. The utopian – and moot – premise is not only that all humans possess inalienable rights but also that all languages are capable of expressing the same set of fundamental propositions.

Here are translations in the predominant writing systems:

1. All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.
2. Hver maður er borinn frjálss og jafn öðrum að virðingu og réttindum. Menn eru gæddir vitsmunum og samvizku, og ber þeim að breyta bróðurlega hverjum við annan.
3. Alle mennesker er født frie og lige i værdighed og rettigheder. De er udstyret med fornuft og samvittighed, og de bør handle mod hverandre i en broderskabets ånd.
4. Tất cả mọi người sinh ra đều được tự do và bình đẳng về nhân phẩm và quyền. Mọi con người đều được tạo hoá ban cho lý trí và lương tâm và cần phải đối xử với nhau trong tình bằng hữu.
5. Adamlaryň hemmesi azat dogulýarlar we öz mertebesi hem-de hukuklary boýunça ilkibaşdan deň dirlir. Olara ozal-başdan aň, ynsap berlendir we biri-birine özara doganlyk ruhunda çemeleşmek olaryň ýaraşygydyr.
6. 人人生而自由，在尊嚴和權利上一律平等。他們賦有理性和良心，並應以兄弟關係的精神相對待。
7. 人人生而自由,在尊严和权利上一律平等。他们赋有理性和良心,并应以兄弟关系的精神相对待。
8. सभी मनुष्यों को गौरव और अधिकारों के मामले में जन्मजात स्वतन्त्रता और समानता प्राप्त है। उन्हें बुद्धि और अन्तरात्मा की देन प्राप्त है और परस्पर उन्हें भाईचारे के भाव से बर्ताव करना चाहिए।
9. يولد جميع الناس أحراراً متساوين في الكرامة والحقوق. وقد وهبوا عقلاً وضميراً وعليهم أن يعامل بعضهم بعضاً بروح الإخاء.
10. সমস্ত মানুষ স্বাধীনভাবে সমান মর্যাদা এবং অধিকার নিয়ে জন্মগ্রহণ করে। তাঁদের বিবেক এবং বুদ্ধি আছে; সুতরাং সকলেরই একে অপরের প্রতি ভ্রাতৃত্বসুলভ মনোভাব নিয়ে আচরণ করা উচিত।
11. Все люди рождаются свободными и равными в своем достоинстве и правах. Они наделены разумом и совестью и должны поступать в отношении друг друга в духе братства.
12. すべての人間は、生まれながらにして自由であり、かつ、尊厳と権利とについて平等である。人間は、理性と良心とを授けられており、互いに同胞の精神をもって行動しなければならない。

The translations are provided by the *UDHR in Unicode* project (<https://unicode.org/udhr/>) and were downloaded from the GitHub datastore for that project (<https://github.com/unicode-org/udhr>) on February 10, 2020.

These are the specific Writing System / Language (**code**) corresponding to the above translations. The **code** can be used to determine the specific *UDHR in Unicode* XML file, which has a name of the form: udhr_**code**.xml.

1. Latin / English (eng)
2. Latin / Icelandic (isl)
3. Latin / Danish (dan)
4. Latin / Vietnamese (vie)
5. Latin / Turkmen (tuk_latn)
6. Han / Traditional Chinese (cmn_hant)
7. Han / Simplified Chinese (cmn_hans)
8. Devanagari / Hindi (hin)
9. Arabic / Standard Arabic (arb)
10. Bengali / Bengali (ben)
11. Cyrillic / Russian (rus)
12. Kana / Japanese (jpn)

A translation in Hieroglyphs is offered by Omniglot.com. However, since [Microsoft Word does not implement the positioning directives needed](#), it is represented here as an image:



Other Writing Systems

In certain places, I have also included Georgian, Klingon, Voynich, Armenian, Hebrew, Hangul (한글, called Chosŏn'gŭl (조선글) in North Korea), Cherokee (GWY), or Thaana (Maldivian or Dhivehi) ... because we have worked with native speakers of these languages, or because I simply fancy them.

Representative Characters

In addition to the sample text, I show several blocks of individual characters that represent each of the writing systems. There are separate blocks of representative characters for each font variant:

Representative Character Blocks by Font Variant

Core

Kurinto Sans Core 16pt

A æ ó â ° π ✖ Ẅ ‰ € ™ ⅜ ⇅ √ ∟ ∣ ▩ ► ☺ fi ₪ ∅ ◉ ◌ Kurinto Font Family © Clint Goss

Main

Kurinto Sans 16pt

୧୧ ୧୨ ୧୩ ୧୪ ୧୫ ୧୬ ୧୭ ୧୮ ୧୯ ୨୦ ୨୧ ୨୨ ୨୩ ୨୪ ୨୫ ୨୬ ୨୭ ୨୮ ୨୯ ୩୦ ୩୧ ୩୨ ୩୩ ୩୪ ୩୫ ୩୬ ୩୭ ୩୮ ୩୯ ୪୦ ୪୧ ୪୨ ୪୩ ୪୪ ୪୫ ୪୬ ୪୭ ୪୮ ୪୯ ୫୦ ୫୧ ୫୨ ୫୩ ୫୪ ୫୫ ୫୬ ୫୭ ୫୮ ୫୯ ୬୦ ୬୧ ୬୨ ୬୩ ୬୪ ୬୫ ୬୬ ୬୭ ୬୮ ୬୯ ୭୦ ୭୧ ୭୨ ୭୩ ୭୪ ୭୫ ୭୬ ୭୭ ୭୮ ୭୯ ୮୦ ୮୧ ୮୨ ୮୩ ୮୪ ୮୫ ୮୬ ୮୭ ୮୮ ୮୯ ୯୦ ୯୧ ୯୨ ୯୩ ୯୪ ୯୫ ୯୬ ୯୭ ୯୮ ୯୯ ୧୦୦

Aux

Kurinto Sans Aux 16pt

𐀀𐀁𐀂𐀃𐀄𐀅𐀆𐀇𐀈𐀉𐀊𐀋𐀌𐀍𐀎𐀏𐀐𐀑𐀒𐀓𐀔𐀕𐀖𐀗𐀘𐀙𐀚𐀛𐀜𐀝𐀞𐀟𐀠𐀡𐀢𐀣𐀤𐀥𐀦𐀧𐀨𐀩𐀪𐀫𐀬𐀭𐀮𐀯𐀰𐀱𐀲𐀳𐀴𐀵𐀶𐀷𐀸𐀹𐀺𐀻𐀼𐀽𐀾𐀿𐁀𐁁𐁂𐁃𐁄𐁅𐁆𐁇𐁈𐁉𐁊𐁋𐁌𐁍𐁎𐁏𐁐𐁑𐁒𐁓𐁔𐁕𐁖𐁗𐁘𐁙𐁚𐁛𐁜𐁝𐁞𐁟𐁠𐁡𐁢𐁣𐁤𐁥𐁦𐁧𐁨𐁩𐁪𐁫𐁬𐁭𐁮𐁯𐁰𐁱𐁲𐁳𐁴𐁵𐁶𐁷𐁸𐁹𐁺𐁻𐁼𐁽𐁾𐁿𐂀𐂁𐂂𐂃𐂄𐂅𐂆𐂇𐂈𐂉𐂊𐂋𐂌𐂍𐂎𐂏𐂐𐂑𐂒𐂓𐂔𐂕𐂖𐂗𐂘𐂙𐂚𐂛𐂜𐂝𐂞𐂟𐂠𐂡𐂢𐂣𐂤𐂥𐂦𐂧𐂨𐂩𐂪𐂫𐂬𐂭𐂮𐂯𐂰𐂱𐂲𐂳𐂴𐂵𐂶𐂷𐂸𐂹𐂺𐂻𐂼𐂽𐂾𐂿𐃀𐃁𐃂𐃃𐃄𐃅𐃆𐃇𐃈𐃉𐃊𐃋𐃌𐃍𐃎𐃏𐃐𐃑𐃒𐃓𐃔𐃕𐃖𐃗𐃘𐃙𐃚𐃛𐃜𐃝𐃞𐃟𐃠𐃡𐃢𐃣𐃤𐃥𐃦𐃧𐃨𐃩𐃪𐃫𐃬𐃭𐃮𐃯𐃰𐃱𐃲𐃳𐃴𐃵𐃶𐃷𐃸𐃹𐃺𐃻𐃼𐃽𐃾𐃿𐄀𐄁𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋𐄌𐄍𐄎𐄏𐄐𐄑𐄒𐄓𐄔𐄕𐄖𐄗𐄘𐄙𐄚𐄛𐄜𐄝𐄞𐄟𐄠𐄡𐄢𐄣𐄤𐄥𐄦𐄧𐄨𐄩𐄪𐄫𐄬𐄭𐄮𐄯𐄰𐄱𐄲𐄳𐄴𐄵𐄶𐄷𐄸𐄹𐄺𐄻𐄼𐄽𐄾𐄿𐅀𐅁𐅂𐅃𐅄𐅅𐅆𐅇𐅈𐅉𐅊𐅋𐅌𐅍𐅎𐅏𐅐𐅑𐅒𐅓𐅔𐅕𐅖𐅗𐅘𐅙𐅚𐅛𐅜𐅝𐅞𐅟𐅠𐅡𐅢𐅣𐅤𐅥𐅦𐅧𐅨𐅩𐅪𐅫𐅬𐅭𐅮𐅯𐅰𐅱𐅲𐅳𐅴𐅵𐅶𐅷𐅸𐅹𐅺𐅻𐅼𐅽𐅾𐅿𐆀𐆁𐆂𐆃𐆄𐆅𐆆𐆇𐆈𐆉𐆊𐆋𐆌𐆍𐆎𐆏𐆐𐆑𐆒𐆓𐆔𐆕𐆖𐆗𐆘𐆙𐆚𐆛𐆜𐆝𐆞𐆟𐆠𐆡𐆢𐆣𐆤𐆥𐆦𐆧𐆨𐆩𐆪𐆫𐆬𐆭𐆮𐆯𐆰𐆱𐆲𐆳𐆴𐆵𐆶𐆷𐆸𐆹𐆺𐆻𐆼𐆽𐆾𐆿𐇀𐇁𐇂𐇃𐇄𐇅𐇆𐇇𐇈𐇉𐇊𐇋𐇌𐇍𐇎𐇏𐇐𐇑𐇒𐇓𐇔𐇕𐇖𐇗𐇘𐇙𐇚𐇛𐇜𐇝𐇞𐇟𐇠𐇡𐇢𐇣𐇤𐇥𐇦𐇧𐇨𐇩𐇪𐇫𐇬𐇭𐇮𐇯𐇰𐇱𐇲𐇳𐇴𐇵𐇶𐇷𐇸𐇹𐇺𐇻𐇼𐇽𐇾𐇿𐈀𐈁𐈂𐈃𐈄𐈅𐈆𐈇𐈈𐈉𐈊𐈋𐈌𐈍𐈎𐈏𐈐𐈑𐈒𐈓𐈔𐈕𐈖𐈗𐈘𐈙𐈚𐈛𐈜𐈝𐈞𐈟𐈠𐈡𐈢𐈣𐈤𐈥𐈦𐈧𐈨𐈩𐈪𐈫𐈬𐈭𐈮𐈯𐈰𐈱𐈲𐈳𐈴𐈵𐈶𐈷𐈸𐈹𐈺𐈻𐈼𐈽𐈾𐈿𐉀𐉁𐉂𐉃𐉄𐉅𐉆𐉇𐉈𐉉𐉊𐉋𐉌𐉍𐉎𐉏𐉐𐉑𐉒𐉓𐉔𐉕𐉖𐉗𐉘𐉙𐉚𐉛𐉜𐉝𐉞𐉟𐉠𐉡𐉢𐉣𐉤𐉥𐉦𐉧𐉨𐉩𐉪𐉫𐉬𐉭𐉮𐉯𐉰𐉱𐉲𐉳𐉴𐉵𐉶𐉷𐉸𐉹𐉺𐉻𐉼𐉽𐉾𐉿𐊀𐊁𐊂𐊃𐊄𐊅𐊆𐊇𐊈𐊉𐊊𐊋𐊌𐊍𐊎𐊏𐊐𐊑𐊒𐊓𐊔𐊕𐊖𐊗𐊘𐊙𐊚𐊛𐊜𐊝𐊞𐊟𐊠𐊡𐊢𐊣𐊤𐊥𐊦𐊧𐊨𐊩𐊪𐊫𐊬𐊭𐊮𐊯𐊰𐊱𐊲𐊳𐊴𐊵𐊶𐊷𐊸𐊹𐊺𐊻𐊼𐊽𐊾𐊿𐋀𐋁𐋂𐋃𐋄𐋅𐋆𐋇𐋈𐋉𐋊𐋋𐋌𐋍𐋎𐋏𐋐𐋑𐋒𐋓𐋔𐋕𐋖𐋗𐋘𐋙𐋚𐋛𐋜𐋝𐋞𐋟𐋠𐋡𐋢𐋣𐋤𐋥𐋦𐋧𐋨𐋩𐋪𐋫𐋬𐋭𐋮𐋯𐋰𐋱𐋲𐋳𐋴𐋵𐋶𐋷𐋸𐋹𐋺𐋻𐋼𐋽𐋾𐋿𐌀𐌁𐌂𐌃𐌄𐌅𐌆𐌇𐌈𐌉𐌊𐌋𐌌𐌍𐌎𐌏𐌐𐌑𐌒𐌓𐌔𐌕𐌖𐌗𐌘𐌙𐌚𐌛𐌜𐌝𐌞𐌟𐌠𐌡𐌢𐌣𐌤𐌥𐌦𐌧𐌨𐌩𐌪𐌫𐌬𐌭𐌮𐌯𐌰𐌱𐌲𐌳𐌴𐌵𐌶𐌷𐌸𐌹𐌺𐌻𐌼𐌽𐌾𐌿𐍀𐍁𐍂𐍃𐍄𐍅𐍆𐍇𐍈𐍉𐍊𐍋𐍌𐍍𐍎𐍏𐍐𐍑𐍒𐍓𐍔𐍕𐍖𐍗𐍘𐍙𐍚𐍛𐍜𐍝𐍞𐍟𐍠𐍡𐍢𐍣𐍤𐍥𐍦𐍧𐍨𐍩𐍪𐍫𐍬𐍭𐍮𐍯𐍰𐍱𐍲𐍳𐍴𐍵𐍶𐍷𐍸𐍹𐍺𐍻𐍼𐍽𐍾𐍿𐎀𐎁𐎂𐎃𐎄𐎅𐎆𐎇𐎈𐎉𐎊𐎋𐎌𐎍𐎎𐎏𐎐𐎑𐎒𐎓𐎔𐎕𐎖𐎗𐎘𐎙𐎚𐎛𐎜𐎝𐎞𐎟𐎠𐎡𐎢𐎣𐎤𐎥𐎦𐎧𐎨𐎩𐎪𐎫𐎬𐎭𐎮𐎯𐎰𐎱𐎲𐎳𐎴𐎵𐎶𐎷𐎸𐎹𐎺𐎻𐎼𐎽𐎾𐎿𐏀𐏁𐏂𐏃𐏄𐏅𐏆𐏇𐏈𐏉𐏊𐏋𐏌𐏍𐏎𐏏𐏐𐏑𐏒𐏓𐏔𐏕𐏖𐏗𐏘𐏙𐏚𐏛𐏜𐏝𐏞𐏟𐏠𐏡𐏢𐏣𐏤𐏥𐏦𐏧𐏨𐏩𐏪𐏫𐏬𐏭𐏮𐏯𐏰𐏱𐏲𐏳𐏴𐏵𐏶𐏷𐏸𐏹𐏺𐏻𐏼𐏽

The last three characters are from the Hieroglyphica-A, -B, -C blocks and are only implemented in the Aux variant of sans serif fonts: **KURINTO SANS AUX**, **KURINTO SERI AUX**, **KURINTO ARIA AUX**, and **KURINTO CALI AUX**.

CJK

Kurinto Sans JP and KR 16pt

尤𠂇電ノ(四)夜翎甲子点⦿ㄣ一&!。£↑囧永亥越魚


KM

Kurinto Sans KM 16pt: ကု ကို

Kurinto Sans TB 16pt: 卐 ૐ

Descriptions

This section (beginning on the next page) lists the representative characters in more detail. Each entry shows the code point, the Kurinto script name and [Script Code], and the name of the representative character for that script. Each section also lists the font used to show the characters.

A	U+0041: Basic Latin [S_Lat0] - LATIN CAPITAL LETTER A.
Æ	U+00E6: Latin 1 Supplement [S_Lat1] - LATIN SMALL LETTER AE.
Ó	U+0151: Latin Extended - A [S_LatA] - LATIN SMALL LETTER O WITH DOUBLE ACUTE.
ă	U+01FB: WGL4 Latin Extended-B [W_LatB] - LATIN SMALL LETTER A WITH RING ABOVE AND ACUTE.
°	U+02DA: WGL4 Spacing Modifier Letters - RING ABOVE.
Π	U+03C0: WGL4 Greek - GREEK SMALL LETTER PI.
Ж	U+0416: WGL4 Cyrillic - CYRILLIC CAPITAL LETTER ZHE.
Ẅ	U+1E85: WGL4 Latin Extended Additional - LATIN SMALL LETTER W WITH DIAERESIS.
‰	U+2030: WGL4 General Punctuation - PER MILLE SIGN.
€	U+20AC: WGL4 Currency Symbols - EURO SIGN.
™	U+2122: WGL4 Letterlike Symbols - TRADE MARK SIGN.
¾	U+215C: WGL4 Eighth Fractions - VULGAR FRACTION THREE EIGHTHS.
↕	U+2195: WGL4 Arrows - UP DOWN ARROW.
√	U+221A: WGL4 Mathematical Symbols - SQUARE ROOT.
⌊	U+2310: WGL4 Miscellaneous Technical - REVERSED NOT SIGN.
⌋	U+2560: WGL4 Box Drawing - BOX DRAWINGS DOUBLE VERTICAL AND RIGHT.
▒	U+2592: WGL4 Block Elements - MEDIUM SHADE.
►	U+25BA: WGL4 Geometric Shapes - BLACK RIGHT-POINTING POINTER.
😊	U+263A: WGL4 Miscellaneous Symbols - WHITE SMILING FACE.
fi	U+FB01: WGL4 Ligatures - LATIN SMALL LIGATURE FI.
┐	U+2423: NRSI Control Pictures - OPEN BOX.
◯	U+25CC: NRSI Geometric Shapes - DOTTED CIRCLE.
◌̂	U+10FB21: Alt Caps and Figures - PETITE CAP A.
	U+10FF00: Font Identification - FOLIO IDENTIFICATION.

ſt	U+FB06: Latin - LATIN SMALL LIGATURE ST.
Ბ	U+1E920: Adlam - ADLAM CAPITAL LETTER KPO.
ش	U+0634: Arabic - ARABIC LETTER KASHMIRI YEH.
Ճ	U+0543: Armenian - ARMENIAN CAPITAL LETTER CHEH.
ꦲ	U+1B10: Balinese - BALINESE LETTER AIKARA.
Ი	U+A6B8: Bamum - BAMUM LETTER SHEUX.
ᯀ	U+1BC5: Batak - BATAK LETTER BA.
ঐ	U+0990: Bengali (Bangla) - BENGALI LETTER AI.
⠄	U+289D: Braille - BRAILLE PATTERN DOTS-13458.
ᨁ	U+1A12: Buginese - BUGINESE LETTER LA.
Ჟ	U+1750: Buhid - BUHID LETTER SA.
ᑭ	U+1571: Unified Canadian Aboriginal Syllabics - CANADIAN SYLLABICS TYI.
Ლ	U+1111A: Chakma - CHAKMA LETTER NAA.
ᩈ	U+AA01: Cham - CHAM LETTER I.
ᯊ	U+13DC: Cherokee - CHEROKEE LETTER DLA.
Ж	U+0416: Cyrillic - CYRILLIC CAPITAL LETTER ZHE.
Პ	U+10427: Deseret (Mormon) - DESERET CAPITAL LETTER EW.
औ	U+0912: Devanagari (Nagari) - DEVANAGARI LETTER SHORT O.
ᯚ	U+1BC58: Duployan shorthand, Duployan stenography - DUPLOYAN LETTER SLOAN U.
ሐ	U+1210: Ethiopic (Ge'ez) - ETHIOPIC SYLLABLE HHA.
ლ	U+10DA: Georgian (Mkhedruli and Mtavruli) - GEORGIAN LETTER LAS.
Ϡ	U+03E2: Greek and Coptic - COPTIC CAPITAL LETTER SHEI.
ૐ	U+0AD0: Gujarati - GUJARATI OM.
᳚	U+11D85: Gunjala Gondi - GUNJALA GONDI LETTER PA.

ਐ	U+0A10: Gurmukhi - GURMUKHI LETTER AI.
Ⴄ	U+10D10: Hanifi Rohingya - HANIFI ROHINGYA LETTER SHA.
ᮊ	U+1730: Hanunoo (Hanunóo) - HANUNOO LETTER SA.
א	U+05D0: Hebrew - HEBREW LETTER ALEF.
あ	U+3042: Hiragana - HIRAGANA LETTER A.
ꦒ	U+A98B: Javanese - JAVANESE LETTER NGA LELET RASWADI.
ಉ	U+0C8A: Kannada - KANNADA LETTER UU.
ボ	U+30DC: Katakana - KATAKANA LETTER BO.
ᆢ	U+A914: Kayah Li - KAYAH LI LETTER NA.
𑌆	U+112B3: Khudawadi, Sindhi - KHUDAWADI LETTER II.
໙	U+0EDD: Lao - LAO HO MO.
ꠘ	U+1C3F: Lepcha (Róng) - LEPCHA PUNCTUATION TSHOOK.
ᱠ	U+1913: Limbu - LIMBU LETTER BHA.
ᱵ	U+A4E4: Lisu (Fraser) - LISU LETTER ZA.
ഇ	U+0D07: Malayalam - MALAYALAM LETTER I.
ܐ	U+0851: Mandaic, Mandaean - MANDAIC LETTER ASZ.
Ა	U+11D00: Masaram Gondi - MASARAM GONDI LETTER A.
᱅	U+AAE0: Meitei Mayek (Meithei, Meetei) - MEETEI MAYEK LETTER E.
ᱫ	U+1E800: Mende Kikakui - MENDE KIKAKUI SYLLABLE M001 KI.
ᄒ	U+16F24: Miao (Pollard) - MIAO LETTER NGHHA.
ᠡ	U+1826: Mongolian - MONGOLIAN LETTER UE.
Ⴄ	U+16A40: Mro, Mru - MRO LETTER TA.
ᨁ	U+1980: New Tai Lue - NEW TAI LUE LETTER HIGH QA.
ꠘ	U+11400: Newa, Newar, Newari, Nepāla lipi - NEWA LETTER A.
᱌	U+07D0: N'Ko - NKO LETTER O.
᱆	U+1C58: Ol Chiki (Ol Cemet', Ol, Santali) - OL CHIKI DIGIT EIGHT.

ꠘ	U+0B1E: Oriya (Odia) - ORIYA LETTER NYA.
ꠘ	U+104B0: Osage - OSAGE CAPITAL LETTER A.
ꠘ	U+10480: Osmanya - OSMANYA LETTER ALEF.
ꠘ	U+16B00: Pahawh Hmong - PAHAWH HMONG VOWEL KEEB.
ꠘ	U+11AC0: Pau Cin Hau - PAU CIN HAU LETTER PA.
ꠘ	U+A930: Rejang (Redjang, Kaganga) - REJANG LETTER KA.
ꠘ	U+0800: Samaritan - SAMARITAN LETTER ALAF.
ꠘ	U+A88E: Saurashtra - SAURASHTRA LETTER AI.
ꠘ	U+1118C: Sharada, Śāradā - SHARADA LETTER VOCALIC LL.
ꠘ	U+1047B: Shavian (Shaw) - SHAVIAN LETTER ERR.
ꠘ	U+0D94: Sinhala - SINHALA LETTER OYANNA.
ꠘ	U+110D0: Sora Sompeng - SORA SOMPENG LETTER SAH.
ꠘ	U+1BBD: Sundanese - SUNDANESE LETTER BHA.
ꠘ	U+A800: Syloti Nagri - SYLOTI NAGRI LETTER A.
ꠘ	U+071E: Syriac - SYRIAC LETTER YUDH HE.
ꠘ	U+1760: Tagbanwa - TAGBANWA LETTER A.
ꠘ	U+1950: Tai Le - TAI LE LETTER KA.
ꠘ	U+1A20: Tai Tham (Lanna) - TAI THAM LETTER HIGH KA.
ꠘ	U+AA80: Tai Viet - TAI VIET LETTER LOW KO.
ꠘ	U+11680: Takri, Ṭākrī, Ṭāṅkrī - TAKRI LETTER A.
ꠘ	U+0B87: Tamil - TAMIL LETTER I.
ꠘ	U+0C34: Telugu - TELUGU LETTER LLLA.
ꠘ	U+0780: Thaana - THAANA LETTER HAA.
ꠘ	U+0E01: Thai - THAI CHARACTER KO KAI.
ꠘ	U+2D65: Tifinagh (Berber) - TIFINAGH LETTER YAZZ.
ꠘ	U+11482: Tirhuta - TIRHUTA LETTER AA.

ᵱ	U+A500: Vai - VAI SYLLABLE EE.
Ა	U+1E2C1: Wancho - WANCHO LETTER A.
᳚	U+118A5: Warang Citi (Varang Kshiti) - WARANG CITI CAPITAL LETTER YO.
ᵱ	U+A000: Yi - YI SYLLABLE IT.
ᵇ	U+0180: Latin Extended - B - LATIN SMALL LETTER B WITH STROKE.
ɐ	U+0250: IPA Extensions - LATIN SMALL LETTER TURNED A.
᳚	U+1CF1: Vedic Extensions - VEDIC SIGN ANUSVARA UBHAYATO.
Ɑ	U+1D00: Phonetic Extensions - LATIN LETTER SMALL CAPITAL A.
Ɑ̇	U+1E00: Latin Extended Additional - LATIN CAPITAL LETTER A WITH RING BELOW.
	U+2016: General Punctuation - DOUBLE VERTICAL LINE.
⁰	U+2070: Superscripts and Subscripts - SUPERSCRIPT ZERO.
⅐	U+2150: Vulgar Fraction - VULGAR FRACTION ONE SEVENTH.
€	U+20A0: Currency Symbols - EURO-CURRENCY SIGN.
ℳ	U+2100: Letterlike Symbols - ACCOUNT OF.
I	U+2160: Number Forms - ROMAN NUMERAL ONE.
↖	U+2196: Arrows - NORTH WEST ARROW.
∀	U+2200: Mathematical Symbols - FOR ALL.
∅	U+2300: Miscellaneous Technical - DIAMETER SIGN.
␣	U+2400: Control Pictures - SYMBOL FOR NULL.
ℴ	U+2440: Optical Character Recognition - OCR HOOK.
①	U+2460: Enclosed Alphanumerics - CIRCLED DIGIT ONE.
⌄	U+256C: Box Drawing - BOX DRAWINGS DOUBLE VERTICAL AND HORIZONTAL.
░	U+2591: Block Elements - LIGHT SHADE.
◈	U+25C8: Geometric Shapes - WHITE DIAMOND CONTAINING BLACK SMALL DIAMOND.
☀	U+2600: Miscellaneous Symbols - BLACK SUN WITH RAYS.
✂	U+2700: Dingbats - BLACK SAFETY SCISSORS.











Ł	U+2C60: Latin Extended - C - LATIN CAPITAL LETTER L WITH DOUBLE BAR.
☞	U+2E19: Supplemental Punctuation - PALM BRANCH.
Ω	U+A7B6: Latin Extended - D - LATIN CAPITAL LETTER OMEGA.
≡	U+A835: Common Indic Number Forms - NORTH INDIC FRACTION THREE SIXTEENTHS.
Ɑ	U+AB30: Latin Extended - E - LATIN SMALL LETTER BARRED ALPHA.
♩	U+1D11E: Musical Symbols - MUSICAL SYMBOL G CLEF.
ᲀ	U+1D200: Ancient Greek Musical Notation - GREEK VOCAL NOTATION SYMBOL-1.
℔	U+1D4DA: Mathematical Alphanumeric Symbols - MATHEMATICAL BOLD SCRIPT CAPITAL K.
฿	U+0E3F: Common Thai - THAI CURRENCY SYMBOL BAHT.
ⴁ	U+10FB: Common Georgian - GEORGIAN PARAGRAPH SEPARATOR.
∥	U+1736: Common Philippine - PHILIPPINE DOUBLE PUNCTUATION.
᠌	U+1805: Common Mongolian - MONGOLIAN FOUR DOTS.
ㇿ	U+309F: Common Katakana - HIRAGANA DIGRAPH YORI.
Ꞇ	U+A92E: Common Kayah Li - KAYAH LI SIGN CWI.
ꦗ	U+A9CF: Common Javanese - JAVANESE PANGRANGKEP.
(U+FD3E: Common Ornate Parentheses - ORNATE LEFT PARENTHESIS.
ㄥ	U+16FE2: Common Old Chinese - OLD CHINESE HOOK MARK.

ᮊ	U+1708: Tagalog (Baybayin, Alibata) - TAGALOG LETTER NA.
ᨿ	U+1173F: Ahom, Tai Ahom - AHOM SYMBOL VI.
𐎶	U+14400: Anatolian Hieroglyphs (Luwian Hieroglyphs, Hittite Hieroglyphs) - ANATOLIAN HIEROGLYPH A001.
𐬵	U+10B34: Avestan - AVESTAN LETTER SSHE.
Რ	U+16AD4: Bassa Vah - BASSA VAH LETTER MBE.
ᱠ	U+11C03: Bhaiksuki - BHAIKSUKI LETTER II.
𑀵	U+1104D: Brahmi - BRAHMI PUNCTUATION LOTUS.
Რ	U+102C5: Carian - CARIAN LETTER II.
Ⴁ	U+1054A: Caucasian Albanian - CAUCASIAN ALBANIAN LETTER CHI.
Ⲛ	U+03E2: Coptic - COPTIC CAPITAL LETTER SHEI.
𐎶	U+12000: Cuneiform, Sumero-Akkadian - CUNEIFORM SIGN A.
𐎶	U+10800: Cypriot syllabary - CYPRIOT SYLLABLE A.
ᱠ	U+11800: Dogra - DOGRA LETTER A.
𐀀	U+13000: Egyptian hieroglyphs - EGYPTIAN HIEROGLYPH A001.
Რ	U+10500: Elbasan - ELBASAN LETTER A.
Რ	U+10FE0: Elymaic - ELYMAIC LETTER ALEPH.
Რ	U+2C00: Glagolitic - GLAGOLITIC CAPITAL LETTER AZU.
Რ	U+10338: Gothic - GOTHIC LETTER THIUTH.
ᱠ	U+11306: Grantha - GRANTHA LETTER AA.
Რ	U+108E0: Hatran - HATRAN LETTER ALEPH.
Რ	U+10840: Imperial Aramaic - IMPERIAL ARAMAIC LETTER ALEPH.
Რ	U+10B60: Inscriptional Pahlavi - INSCRIPTIONAL PAHLAVI LETTER ALEPH.
Რ	U+10B40: Inscriptional Parthian - INSCRIPTIONAL PARTHIAN LETTER ALEPH.
ᱠ	U+1108C: Kaithi - KAITHI LETTER AU.

𐌀	U+10A13: Kharoshthi - KHAROSHTHI LETTER GHA.
𑀭	U+11216: Khojki - KHOJKI LETTER DDA.
𐎶𐎵	U+10629: Linear A - LINEAR A SIGN AB048.
𐎶𐎵	U+10010: Linear B - LINEAR B SYLLABLE B044 KE.
𐌆	U+1028C: Lycian - LYCIAN LETTER Q.
𐌆	U+10934: Lydian - LYDIAN LETTER TT.
𑀧	U+11150: Mahajani - MAHAJANI LETTER A.
𐎶𐎵	U+10ADF: Manichaean - MANICHAEAN LETTER XOPH.
𐌆	U+11C70: Marchen - MARCHEN HEAD MARK.
𐎶𐎵	U+109DE: Meroitic Cursive - MEROITIC CURSIVE NUMBER FOUR THOUSAND.
𐎶𐎵	U+10980: Meroitic Hieroglyphs - MEROITIC HIEROGLYPHIC LETTER A.
𑀭	U+11600: Modi, Modī - MODI LETTER A.
𑀭	U+11280: Multani - MULTANI LETTER A.
𐎶𐎵	U+10880: Nabataean - NABATAEAN LETTER FINAL ALEPH.
𐎶𐎵	U+1698: Ogham - OGHAM LETTER IFIN.
𐎶𐎵	U+10CB1: Old Hungarian (Hungarian Runic) - .
𐎶𐎵	U+10300: Old Italic (Etruscan, Oscan, etc.) - OLD ITALIC LETTER A.
𐎶𐎵	U+10A8F: Old North Arabian (Ancient North Arabian) - OLD NORTH ARABIAN LETTER ES-3.
𐎶𐎵	U+10350: Old Permic - OLD PERMIC LETTER AN.
𐎶𐎵	U+103A0: Old Persian - OLD PERSIAN SIGN A.
𐎶𐎵	U+10F00: Old Sogdian - OLD SOGDIAN LETTER ALEPH.
𐎶𐎵	U+10A60: Old South Arabian - OLD SOUTH ARABIAN LETTER HE.
𐎶𐎵	U+10C11: Old Turkic, Orkhon Runic - OLD TURKIC LETTER ORKHON AD.
𐎶𐎵	U+1086C: Palmyrene - PALMYRENE LETTER MEM.
𐎶𐎵	U+A840: Phags-pa - PHAGS-PA LETTER KA.
𐎶𐎵	U+10900: Phoenician - PHOENICIAN LETTER ALF.

𐭪	U+10B90: Psalter Pahlavi - PSALTER PAHLAVI LETTER SHIN.
ᚱ	U+16A0: Runic - RUNIC LETTER FEHU FEOH FE F.
𑖀	U+11580: Siddham, Siddham, Siddhamāṭṭkā - SIDDHAM LETTER A.
𐭥	U+10F54: Sogdian - SOGDIAN NUMBER ONE HUNDRED.
Ბ	U+11A50: Soyombo - SOYOMBO LETTER A.
𐞪	U+17000: Tangut - TANGUT L2008-0008.
𐎠	U+10384: Ugaritic - UGARITIC LETTER DELTA.
ᠠ	U+11A00: Zanabazar Square (Zanabazarin Dörböljin Useg, Xewtee Dörböljin Biciḡ, Horizontal Square Script) - ZANABAZAR SQUARE LETTER A.
𐆑	U+1012A: Agean Numbers - AEGEAN NUMBER NINE THOUSAND.
Ϟ	U+1016E: Ancient Greek Numbers - GREEK ACROPHONIC THESPIAN FIVE HUNDRED.
HS	U+10198: Ancient Symbols - ROMAN SESTERTIUS SIGN.
𐌀	U+101F8: Phaistos Disc - PHAISTOS DISC SIGN FLUTE.
Ϡ	U+102ED: Coptic Epact Numbers - COPTIC EPACT NUMBER FORTY.
𐍮	U+1D037: Byzantine Musical Symbols - BYZANTINE MUSICAL SYMBOL KATAVA TROMIKON.
ꠐ	U+1D2E0: Mayan Numerals - MAYAN NUMERAL ZERO.
䷌	U+1D33A: Tai Xuan Jing Symbols - TETRAGRAM FOR ETERNITY.
正	U+1D376: Counting Rod Numerals - IDEOGRAPHIC TALLY MARK FIVE.
東	U+1F000: Mahjong Tiles - MAHJONG TILE EAST WIND.
𐞓	U+1F05A: Domino Tiles - DOMINO TILE HORIZONTAL-05-06.
♠	U+1F0CF: Playing Cards - PLAYING CARD BLACK JOKER.
😄	U+1F600: Emoticons - GRINNING FACE.
🚀	U+1F680: Transport and Map Symbols - ROCKET.
⚗	U+1F70A: Alchemical Symbols - ALCHEMICAL SYMBOL FOR VINEGAR.
♔	U+1FA00: Chess Symbols - NEUTRAL CHESS KING.
✚	U+16ED: Common Runic - RUNIC CROSS PUNCTUATION.

ᑭ	U+E000: Tengwar - TENGWAR LETTER TINCO.
ᑭ̃	U+E080: Cirth - CIRTH LETTER P.
ᑭ̇	U+E150: Kinya - KINYA LETTER K.
ᑭ̈	U+E200: Verdurian - VERDURIAN CAPITAL LETTER U.
ᑭ̉	U+E270: aUI - AUI LETTER A.
ᑭ̊	U+E290: Amman-Iar - AMMAN-IAR LETTER P.
ᑭ̋	U+E3B0: Olaetyan - OLAETYAN LETTER A.
ᑭ̌	U+E5C0: Gargoyle - GARGOYLE LETTER P.
ᑭ̍	U+E630: Seussian Latin Extensions - SEUSS LETTER YUZZ.
ᑭ̎	U+E650: Sylabica - SYLABICA LETTER I.
ᑭ̏	U+E684: Ewellic - EWELLIC LETTER NASAL AA.
ᑭ̐	U+E6D0: Amlin - AMLIN VOWEL ARAL.
ᑭ̑	U+E6F0: Unifon - UNIFON SMALL LETTER CHILL.
ᑭ̒	U+E770: Solresol - SOLRESOL SYLLABLE DO.
ᑭ̓	U+E780: Visible Speech - VISIBLE SPEECH CONSONANT LETTER BACK PRIMARY.
ᑭ̔	U+E830: D'Ni - DNI LETTER V.
ᑭ̕	U+E890: Aurebesh - AUREBESH LETTER AUREK.
ᑭ̖	U+E8E9: Tonal - TONAL HEXADECIMAL DIGIT NINE.
ᑭ̗	U+E949: Glaitha-A - GLAITHA-A CAPITAL LETTER AURA WITH DIAERESIS.
	U+E98E: Glaitha-B - GLAITHA-B NUMBER COMPONENT SIXTY.
ᑭ̘	U+EAA0: Wanya - WANYA CAPITAL LETTER B.
ᑭ̙	U+EB00: Orokin - OROKIN LETTER P.
ᑭ̚	U+ED22: Deini - DEINI LETTER J.
ᑭ̛	U+F4C0: Ath - ATH DIGIT ZERO.
ᑭ̜	U+F8A0: Aiha - AIHA LETTER K.
ᑭ̝	U+F8D0: Klingon - KLINGON LETTER A.

	U+F0000: Kinya Syllables - KINYA SYLLABLE KAK.
	U+E000: Standard Music Font Layout Version 1.3 - .
	U+F700: Creative Commons - CREATIVE COMMONS ICON CC.
	U+F5000: Regular Icons - ICON BLACKHEARTSUIT.
	U+F50A0: Solid Icons - ICON SOLID GLASSMARTINI.
	U+F5450: Brand Icons - ICON BRAND TWITTERSQUARE.
	U+FF45C: Voynich - VOYNICH LETTER EVA-156.
	U+103434: Hieroglyphica - A - AUXILIARY HIEROGLYPHIC A1. (Kurinto Sans Aux only)
	U+104680: Hieroglyphica - B - AUXILIARY HIEROGLYPHIC B1. (Kurinto Sans Aux only)
	U+104460: Hieroglyphica - C - AUXILIARY HIEROGLYPHIC C1. (Kurinto Sans Aux only)

KURINTO SANS JP, unless otherwise indicated.

ㄸ	U+3124: Bopomofo - BOPOMOFO LETTER ANG.
ㅏ	U+1185: Hangul (Hangŭl, Hangeul) - HANGUL JUNGSEONG YO-YAE. [KR]
𪛗	U+2EEA: Han (Hanzi, Kanji, Hanja) - CJK RADICAL C-SIMPLIFIED FROG.
ノ	U+3034: CJK Symbols and Punctuation - VERTICAL KANA REPEAT WITH VOICED SOUND MARK UPPER HALF.
(四)	U+3223: Enclosed CJK Letters and Months 1 - PARENTHESESIZED IDEOGRAPHIC FOUR.
㝱	U+32B0: Enclosed CJK Letters and Months 2 - CIRCLED IDEOGRAPHIC NIGHT.
令和	U+32FF: Reiwa - SQUARE ERA NAME REIWA. Added in Unicode version 12.1
甲	U+3199: Kanbun - IDEOGRAPHIC ANNOTATION FIRST MARK.
ㄣ	U+31E1: CJK Strokes - CJK STROKE HZZZG.
0点	U+3358: Ideographic Compatibility - IDEOGRAPHIC TELEGRAPH SYMBOL FOR HOUR ZERO.
䷀	U+4DC0: Yijing Hexagram Symbols - HEXAGRAM FOR THE CREATIVE HEAVEN.
ㄱ	U+FE18: Vertical Forms - PRESENTATION FORM FOR VERTICAL RIGHT WHITE LENTICULAR BRACKET.
┌	U+FE3C: CJK Compatibility Forms - PRESENTATION FORM FOR VERTICAL RIGHT BLACK LENTICULAR BRACKET.
&	U+FE60: Small Form Variants - SMALL AMPERSAND.
!	U+FF01: Fullwidth Forms - FULLWIDTH EXCLAMATION MARK.
。	U+FF61: Common Halfwidth - HALFWIDTH IDEOGRAPHIC FULL STOP.
£	U+FFE1: Common Fullwidth - FULLWIDTH POUND SIGN.
↑	U+FFEA: Common Halfwidth Vertical - HALFWIDTH UPWARDS ARROW.
コ	U+1F201: Enclosed Ideographic Supplement - SQUARED KATAKANA KOKO.

永	U+6C38: CJK Unified Ideographs - CJK UNIFIED IDEOGRAPH-6C38.The Kanji symbol for “eternal”. This character has each of the eight basic strokes exactly once. See https://en.wiktionary.org/wiki/%E6%B0%B8 .
亥	U+20014: CJK Unified Ideographs Extension B - UCS2003.
𪛗	U+2A717: CJK Unified Ideographs Extension C - GCYY-00061.
𪛖	U+2B756: CJK Unified Ideographs Extension D - H-JTC064.
◻	U+2B837: CJK Unified Ideographs Extension E - V4-4039.
◻	U+2CED2: CJK Unified Ideographs Extension F - JMJ-056834.
◻	U+2F80C: CJK Compatibility Ideograph Supplement - TF-594D.

KM

Kurinto Sans KM

ញ	U+1789: Khmer - KHMER LETTER NYO.
ဣ	U+1024: Myanmar (Burmese) - MYANMAR LETTER II.

TB

Kurinto Sans TB

卐	U+0FD5: Common Svasti - RIGHT-FACING SVASTI SIGN.
ཱ	U+0F00: Tibetan - TIBETAN SYLLABLE OM.

Mojibake

Text representations of mojobake were produced from the Russian-language version of [Article 1 of the UDHR](#), converted from UTF-8 to ISO-8859-5 (code page 28595) using the GNU iconv utility.

Visual Centering and Alignment

One of the elements of the *Character Design Standards* published by Microsoft ([CDS 2018]) is the visual centering of specific characters. For example, “@” should be visually centered between “H” and “O” ... i.e. the display of “H@O” should appear visually centered.

The *Kurinto Specimen Book* (Kurinto_SpecimenBook.pdf, included in each release package) has a section of *Visual Centering Displays* which implement each of the numerous recommendations of the *Character Design Standards* for every typeface, roughly in the order specified in the *Standard*.

Here is an example, for Kurinto Text. Refer to the *Character Design Standards* ([CDS 2018]) for the specifics of each display:

Kurinto Text – Visual Centering and Alignment

OHOHAHBHCOAO ononanbncnoaobocodo
0^a01^a2^a3^a4^a5^a6^a7^a8^a9^a0^o1^o2^o3^o4^o5^o6^o7^o8^o9^o0
0102030405060708090 n.n,n:n;no.o,o;o;o
H₂O A₀B⁰C₁D¹E₂F²G₃H³I₄J⁴K₅L⁵M₆N⁶O₇P⁷Q₈R⁸S₉T⁹
H(H)H[H]H{H}HO(O)O[O]O{O}O n.n,n:n;no.o,o;o;o
0•0n...n...o...o...n...n...o...o...
HTMHTMOTMOTM_E_ x-x—x O©O®O®O©O©O
H@H@O@O H•H•O•O H/H/O/O/O\H\H
o¿ig?!O 0|0\$0¢0c0£0¤0¥0f0F0£0Pts0€0
0+0–0±0×0•0~0^0°0≈0=0<0≤0≥0>0

Windows Font Installation

The [font installation instructions for Windows 10](#) will work in most cases. However, users who repeatedly install and uninstall fonts may encounter frustrating situations:

- Fonts files are loaded into /Windows/Fonts, but do not show up in an application's font list.
- Font installation says that a font is already installed when it is not, and clicking **Yes** to replace the font fails to install the font.
- You find many duplicated fonts in /Windows/Fonts with extensions such as _0, _1, ...
- Windows shows only some of the installed fonts in a typeface.
- And (**particularly** frustrating) your application winds up accessing the **wrong version** of an installed font.

There are many Web-based sources of information on these situations which provide more detailed and timely information than I can cover in this document. However, here are some basic things to note:

- Current versions of Windows use a completely different directory structure for fonts that are installed for a single user versus for all users.
- When fonts are installed, Windows copies the font file into a system folder and also updates a list of available fonts in the Windows registry. Many bizarre issues arise if the list of fonts in the registry gets out of step with the font files in the system folder.
- Windows does not delete superseded fonts – it renames the font files. It can also restore previously “replaced” fonts, making the job of actually deleting a font difficult.
- Fonts listed in the registry with incorrect information (such as referring to a .ttf font when the actual file name is .otf) can render the font inaccessible.

Here are some pointers to resources that I have come across:

- “Fonts Seem to Install, but Don’t” - <https://forums.adobe.com/thread/951333>
- How to Delete Fonts using the Windows Registry – <https://www.maketecheasier.com/delete-fonts-using-windows-registry/>
- What is the purpose of the Fonts key in the Registry? – <https://superuser.com/questions/813039/what-is-the-purpose-of-the-fonts-key-in-the-registry>

In addition, the [font installation instructions for Windows 10](#) will **only** work if the default viewer for fonts is set to Windows Font Viewer. If the default view has been changed to another application, the context menu items **Install** and **Install for all users** will not be available.

Testing

Testing is done using a set of Microsoft Word document and a set of automated font testing tools.

Testing Documents

The testing documents include this User Guide and related user documentation, the array of Specimen Book documents, and the Publication Sample documents included in the /Misc directory of the distribution package.

Testing Tools

All fonts are tested using these tools. Public versions of Kurinto are not released until all tests are passed.

- **FontValidator.** See <https://github.com/HinTak/Font-Validator/>
- **Font Bakery.** The Google open source fontbakery tool is executed with -check-universal. See <https://github.com/googlefonts/fontbakery>
- **kval (kmet -v)** – Kurinto’s own testing software, which primarily applies checks against the recommendations in the *Character Design Standards* section of Microsoft Typography’s *OpenType Font Development* standards. See <https://docs.microsoft.com/en-us/typography/develop/character-design-standards/>.
- **kismet** – A tool specific to Kurinto that tests for inconsistencies in metric compatible fonts against the model fonts.

References

[CDS 2018] Microsoft Typography, *Character Design Standards*, retrieved February 2, 2019 from <https://docs.microsoft.com/en-us/typography/develop/character-design-standards/>.

[Evans 2013] Lorna Evans, *Recommended Characters for Non-Roman Fonts*, retrieved January 21, 2018 from <https://scriptsource.org/entry/gg5wm9hhd3>. See also subsequent comments and updates on the cited web page that modify the recommendations of the posted document.

[FF 2017] FontForge Community, *Design with FontForge – A Book About How to Create New Typefaces Using FontForge*, 205 pages, 2017, retrieved May 19, 2020 from <http://designwithfontforge.com/ebook/design-with-fontforge-en-US.pdf>.

[GPO 2016] U.S. Government Publishing Office, *Style Manual: An Official Guide to the Form and Style of Federal Government Publications*, 2016, 461 + xiii pages. ISBN 978-0-16-093601-2, retrieved from <https://www.govinfo.gov/content/pkg/GPO-STYLEMANUAL-2016/pdf/GPO-STYLEMANUAL-2016.pdf> on April 9, 2020.

[Kellman 2016] Steven G. Kellman, *Omnilingual Aspirations: The Case of the Universal Declaration of Human Rights*, *Critical Multilingualism Studies*, Volume 4, Number 1 (2016), pages 5–24.

[PRC-Edu 2013] Ministry of Education of the People’s Republic of China, 通用规范汉字表 «Table of General Standard Chinese Characters», published June 18, 2013,

<http://www.gov.cn/gzdt/att/att/site1/20130819/tygfzhzb.pdf>. See also
https://en.wikipedia.org/wiki/Table_of_General_Standard_Chinese_Characters.

[SIL 2020] SIL / NRSI, *Font Development Best Practices, Technical Guidance Regarding Font Development And Production*, <https://silnrsi.github.io/FDBP/en-US/>, retrieved March 15, 2020.

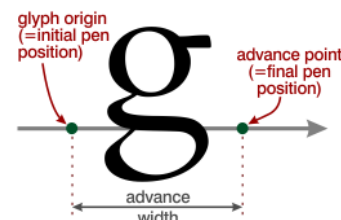
[Unicode 2019] The Unicode Consortium, *The Unicode® Standard, Version 12.0 – Core Specification*, March 2019, 1018 + xxviii pages, ISBN 978-1-936213-22-1,
<http://www.unicode.org/versions/Unicode12.0.0/>.

Glossary and Acronyms

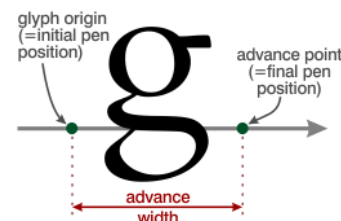
Terms shown in **red bold** are my own inventions for this project. [TG] indicates entries that are sourced from <http://Typography.Guru>.

advance point A point that indicates the right side of a glyph. The advance point sits on the baseline and is spaced from the glyph origin by the advance width.

Sometimes called the “final pen position”. Also called the “right side bearing point” in FontCreator.

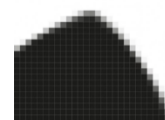


advance width The linear distance between the start of one character and the start of the next sequential character, in the reading direction of the text and in the absence of kerning and tracking. The reading direction could be a horizontal (left-to-right or right-to-left) or vertical (top-to-bottom) orientation. Often abbreviated “ADW”.



The term “character width” is sometimes use colloquially for advance width. However, this term can be confused with the bounding box of the contours of a character.

anti-aliasing A method used to minimize aliasing (jagged and pixelated edges) from (for example) text outlines on pixel-based displays. [TG]

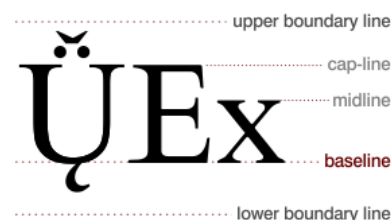


ASCII A character encoding for 128 characters – the largest character set that can be represented in 7 bits of data.

ASCII comprises 95 printable characters and 33 control characters. The printable characters correspond to the 95 Basic Latin characters of Unicode. The See <https://en.wikipedia.org/wiki/ASCII>.

baseline In certain scripts (e.g. Latin, Cyrillic, Greek), the baseline is the line upon which letters (without descenders) sit. [TG]

More specifically called the “alphabetic baseline” to distinguish it from the ideographic baseline.

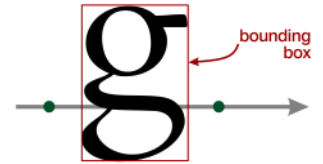


body text The text that forms the main content of designs (like ads) or entire publications (like magazines, books and so on). Also called “body copy”. [TG]

boundary height The distance from the Lower Boundary Line to the Upper Boundary Line: (UBL – LBL).

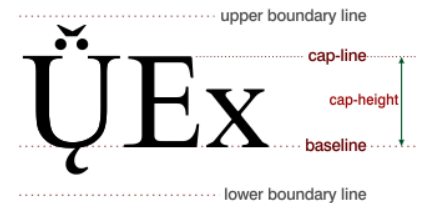


bounding box A virtual box that surrounds a set of contours as tightly as possible. Bounding boxes apply to single contours, glyphs (except empty ones), and all the glyphs of a font taken together. The bounding box is often expressed numerically by four values, often labeled xMin, yMin, xMax, and yMax. “Bounding box” is often abbreviated as “bbox”.



Since contours in OpenType fonts have curves that can extend beyond the points that define the curve, calculating a bounding box based on the “inked area” of the contours is complex and computationally expensive. Because of this, the actual bounding box data in an OpenType font is based on the control points (both “on-curve” and “off-curve”) that define the contours.

cap-height The distance from the baseline to the cap-line.



cap-line A virtual line that intersects the top of the upper-case letters with flat tops, such as “X”, “Z” or “E”. Letters with rounded tops, such as “O”, are not considered, since they typically “overshoot” the midline. The cap-line defines the cap-height metric of the font.



character A character (as used in *The Unicode Standard*) is an abstract entity, such as LATIN CAPITAL LETTER A or BENGALI DIGIT FIVE. Characters have associated code points, such as U+0041 and U+09EB.

Characters are all about meaning, not appearance. Characters have no specific relationship with the visual representation of the character on screen or paper.⁵⁶ So, for example, the characters U + 03A9 GREEK CAPITAL LETTER OMEGA and U + 2126 OHM SIGN may have identical appearances (“Ω” and “Ω”). However, a text-to-speech reader should pronounce “a 339 Ω resistor” as “a three hundred and thirty-nine **Ohm** resistor” and not “a three hundred and thirty-nine **upper-case omega** resistor”.

character width See advance width.

CJK Chinese–Japanese–Korean – a collective term for these languages, each of which includes variations of Chinese characters that are part of the Unicode Han writing system.

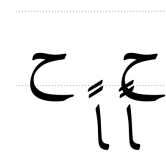
clipping A situation where a portion of a character is not rendered or not visible. The situation usually involves a straight boundary above or below a character beyond which the character is clipped.

⁵⁶ This definition is based on *The Unicode Standard, Version 12.0 – Core Specification*, page 6.

code page A scheme for mapping a limited set of code points onto a collection of characters. Each code page has a number (e.g. code page 1250) that identifies the particular mapping. Many of the code pages are limited to 256 code points so that they could be represented internally by 8 bits. This limited scheme of switching code pages when different characters are needed predates the Unicode era and was fraught with issues, annoyances, confusion, and errors.

code point A hexadecimal (base 16) number assigned to a Unicode character. This document writes code points as, for example, U+1234.

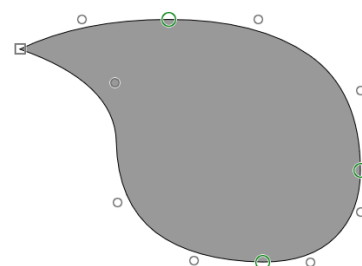
collision A situation where characters on neighboring lines intersect. This happens when a character with a very low descender intersects with a character on the subsequent line that has a very high ascender.



In Kurinto fonts, this only happens when at least one of the characters extends past the [UBL](#) or [LBL](#) boundary lines, as is shown with the two Arabic characters on the right.

contours Graphic shapes. In OpenType fonts, contours are defined by a set of control points that define a shape as a sequence of connected lines and curves.

control points The data points that define the shape of a contour. A control point can define an “on-curve” control point, through which the contour passes, or an “off-curve” control point, which affects the curvature of a segment between two on-curve control points.



In the contour shown at the right, the on-curve points are represented by the squares and larger green circles. The off-curve points are shown as the smaller circles. See https://en.wikipedia.org/wiki/Bézier_curve for a deep dive into how vector graphics work.

diacritics Additional contours (graphic shapes) added above a letter (Å), below a letter (À), within a letter – such as the double tilde (ö), or between two letters – such as the interpunct (◌◌). Also called “diacritical marks” or “accents”, for diacritics such as the acute (ó) and the grave (ò).

digits See figures.

dingbats A set of ornamental characters that are often used to represent concepts. Some examples: ✂✚✳➡✓

duospace See also monospace and multi-space.

em A unit of measure equal to the point size of the text being rendered. Since an inch has 72 points, one em will equal one inch for 72pt text.

Historically, the line height was often equal to one em and the width of a capital letter M was often one em. With the inclusion of languages that contain stacked diacritics and larger descenders, line height is often larger than one em square in order to avoid collisions. Line height is fixed at $1\frac{1}{8}$ em for all Kurinto fonts.

em space A space character that has the width of one em.

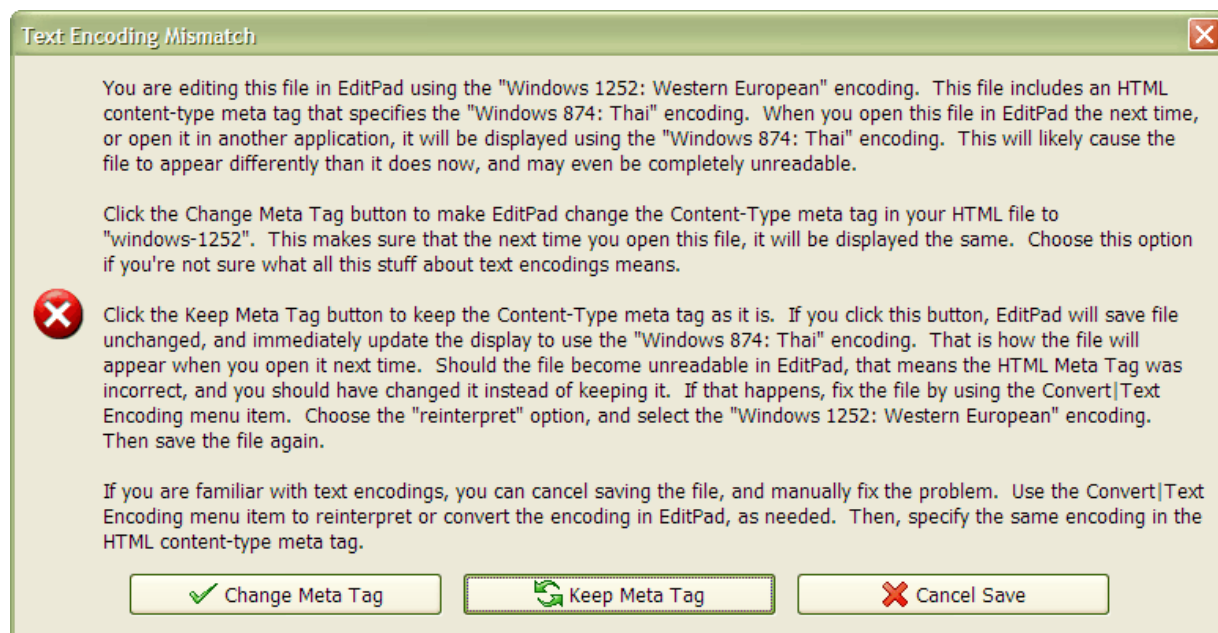
em square A square of size 1em × 1em.

en Half of one em.

encoding An encoding defines how to represent a set of characters digitally – i.e. how to “map” a set of bits to a collection of characters. The full term is “character encoding”.

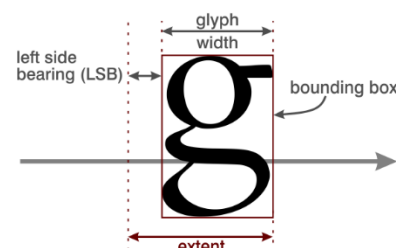
Encoding systems typically represent each character using one or a few bytes of data. Issues arise when a person or system does not know the encoding system that was used to encode the original text. This can lead to incorrectly interpreting the text and rendering it as [mojibake](#).

A wide range of text encodings were prevalent prior to the adoption of Unicode as the predominant encoding scheme. This caused persistent problems with users who had to deal with confusing system and corrupted text. This is an example of a dialog box from the *EditPad* application when it detected a possible encoding mismatch:



extent The distance from the glyph origin (initial pen position) to the right side (maximum X value) of the bounding box around the glyph. The extent = LSB + glyph width.

This is the definition used by the TrueType/OpenType specifications and is used in this User Guide. Note that, in certain other context, the extent matches the glyph width.



See, for example, the FreeType documentation at <https://www.freetype.org/freetype2/docs/glyphs/glyphs-3.html>.

figures The digits “0”–“9”. Also called “digits” and “numerals”.

fixed-pitch See pitch and monospace.

font A member of a typeface for a specific style (e.g. **NARROW BOLD ITALIC**) and variant (e.g. **TC** for “Traditional Chinese”). In Kurinto, each font is implemented by a single TTF font file with a `.ttf` extension.

font family See Typeface.

font folio A collection of typefaces (font families) that are designed to work together in a coordinated way. The fonts in the folio have coordinated naming, styles, metrics, embedding permissions, Panose settings, and typographic standards. The font families in the folio may fall into multiple classifications (e.g. serif, sans-serif, calligraphic, monospaced, etc.).

The term “font folio” is newly invented (as far as I know) for this project. In some circles, the term “font superfamily” is used for this concept.

FUnits Font Units. Glyphs are composed of contours. Each contour is laid out on an abstract coordinate system using a set of points that describe the location and curvature of that contour. An FUnit is the smallest unit of measure on that coordinate system.

Kurinto fonts use a system of 2,048 FUnits per em. So, when rendering 12pt text in a Kurinto font, each inch (72 points) of distance represents 12,288 FUnits.

glyph The visual appearance given to a character by a font. In practice, the font only specifies an intended visual appearance (using points and curves). It is up to the computer system’s rendering engine to display the shape of the character on the output medium (screen, printer, plotter, retina-implanted rasterizer, direct-brain desktop, etc. In OpenType fonts, the shape of a glyph is defined by a set of contours.

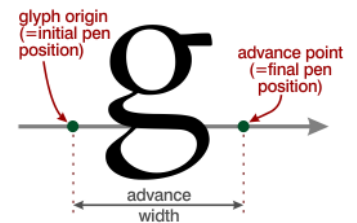
glyph-compatible license A term I use for a font license that indicates whether glyphs in one font can be used in the development of another font (see [Legal Disclaimer](#)).

I say that a license, **A**, is “glyph-compatible” with license, **B**, if both **A** and **B** allow code and data (typically glyph data) to be extracted from a font released under **A** and incorporated into a font released under **B**.

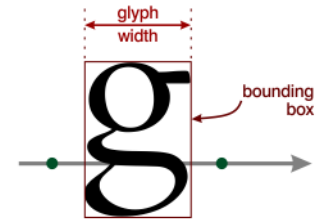
Source fonts used to compose Kurinto fonts must be released under licenses that are glyph-compatible with the OFL. See also the `OFL_LicenseCompatibility.pdf` document in the `/Doc` directory of the release package.

glyph origin A point that indicates the left side of a glyph. The glyph origin sits on the baseline and is spaced from the left edge of the glyphs's bounding box by the left side bearing. Note that the glyph origin is not necessarily located at X=0 (see Offset).

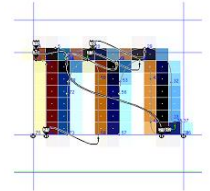
Sometimes called the “initial pen position”. Also called the “left side bearing point” in FontCreator.



glyph width The horizontal size of the bounding box of a glyph. This is the distance between the leftmost inked contour of a glyph to the rightmost inked contour of that glyph. It is often expressed as xMax-xMin.



hinting Display screens often have lower resolution compared with printing systems such as laser printers. When characters are displayed at smaller sizes, legibility is reduced since fewer screen pixels are available to render the characters. Hinting is a process of augment a font with specialized instructions to improve on-screen legibility. The specialized instructions are used by the rendering system for the screen display to adjust the brightness or color of pixels on the edges of curves to make the characters more recognizable. (The image at the right is from [TG].)



ideograph See logograph.

ideographic baseline The baseline is the line upon which logographs (ideographs) sit. Distinguished from “alphabetic baseline”.

interline spacing The vertical space between successive lines of text. If text is written vertically (top-to-bottom or bottom-to-top), then interline spacing is the horizontal space between vertical columns of text. Also called “leading”.

kerning The process of fine-tuning the space between characters to improve readability and visual appeal. See the [Kerning Pitfall](#) for an example.

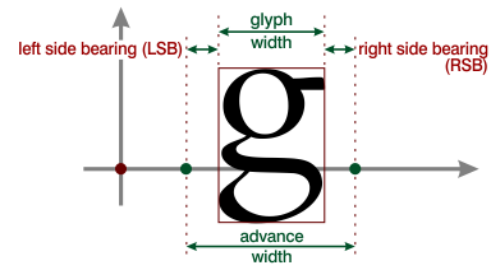


LBL Lower Boundary Line. A horizontal line that represents the intended vertical boundary between a line of single-spaced text and the start of the next line of text. See [Boundary Lines](#).



leading See interline spacing.

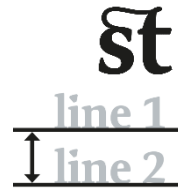
left side bearing (LSB) The distance between the left side of the glyph's width and the left side of the visual glyph design. If the glyph design exceeds the width of the glyph, the side bearing values can also be negative. [TG].



left side bearing point See [glyph origin](#).

ligature A visual and/or technical connection of two or more letters or glyphs. [TG].

line spacing Line spacing, interline spacing, line height, or “leading” is the distance between the baselines of successive lines of type. The term leading has its roots in letterpress printing where it referred to the lead strips added between the lines [TG].



Line spacing is fixed at $\frac{11}{8}$ of an em for all Kurinto fonts.

lining figures Figures (digits) with a uniform height similar or equal to the cap height of the font. [TG].

163

logograph A character that represents a word, phrase, or concept. For example, the Chinese character 永 (U+6C38) corresponds to the concept of “eternal” in English.⁵⁷

The Han writing system is composed of logographs. Han is the primary writing system of Chinese and is used in combination with other writing systems in Japanese and Korean, is composed of logographs.

Also commonly called “ideograph” in Unicode and font development, although the term “ideograph” has a slightly different meaning.⁵⁸ Also called “logogram”.

LTR Left-to-Right. This can pertain to text direction or the sequence of vertical lines of text on the page – as in “TTB-LTR” for “Top-to-Bottom – Left-to-Right”.

metric compatible font A font that can replace another original font without changing the layout of your document. The original font is usually a popular font such as **ARIAL** or **TIMES NEW ROMAN**. Each character in the metric compatible font takes the same space (horizontal and vertical) as the original font.

metrics model A font whose metrics – line height, advance widths, underline positioning, etc. – are used as a model for building a metric compatible font.

⁵⁷ See <https://en.wiktionary.org/wiki/%E6%B0%B8>.

⁵⁸ See <https://en.wikipedia.org/wiki/Ideogram> for the distinction.

midline A line that intersects the top of lower-case letters with flat tops, such as “x”, “z” or “v”. Letters with rounded tops, such as “o”, are not considered, since they typically “overshoot” the midline. Likewise, letters with ascenders, such as “l”, are not considered. The midline defines the x-height metric of the font.



mojibake A Japanese-language phrase (文字化け – Moh-Jee-Bah-Kay) used for garbled characters. The phrase translates to English as “character transformation” and is often pronounced “Moh-jee-Bake”.

Garbled characters often arise when the recipient of text incorrectly guesses the [encoding scheme](#) used on the original text and uses the wrong decoding scheme. Here is an example of a display of the Japanese-language version of a Creative Commons license:

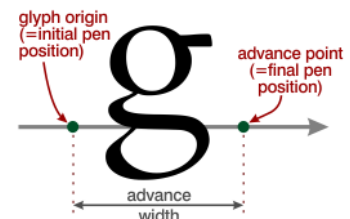
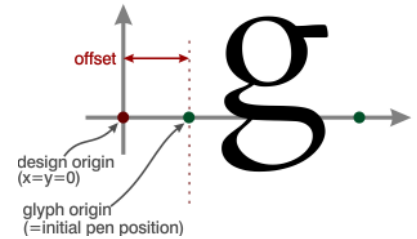


monospace In the strict sense, a monospaced typeface or font has the same [advance width](#) for all characters. However, this term is often used loosely when referring to duospace or multi-space fonts. There are also a class of “faux-monospace” fonts – see the discussion in the description of KURINTO TYPE.

multi-space A font that uses a small set of possible [advance widths](#) for its characters, with large classes of characters sharing the same advance width. See also monospace and duospace.

numerals See figures.

- NRSI** The Non-Roman Script Initiative – an effort by SIL International to develop a set of recommendations and fonts for fonts that use writings systems outside of the Pan-European region.
- offset** The horizontal distance from the design origin – located on the baseline at $X=0$ – to the glyph origin. This glyph metric is often, but not necessarily, set to zero.
- Some fonts are designed so that the offsets of all the glyphs in the font are zero, and this may provide some efficiencies when rendering text in that font.
- OFL** The *SIL Open Font License Version 1.1* – the open source license under which the Kurinto font package (including this document) is released. See <http://scripts.sil.org/OFL>.
- oldstyle figures** Figures (digits) with ascenders and descenders like lowercase letters. [TG]. **163**
- OTF** **OpenType Format** – a digital font file format that is based on the TTF. It was developed by Microsoft and Adobe Systems in the 1990s. See <https://en.wikipedia.org/wiki/OpenType>.
- Panose** A system developed by Benjamin Bauermeister for classifying typefaces solely on their visual characteristics. It can be used to match a known font to its closest visual neighbor from a font pool. This name of this system is often written in all capitals – “PANOSE”.
- PDF** **Portable Document Format** – a file format for documents that display reliably and consistently on all prevalent computer and operating systems. It was developed by Adobe System in the 1990s and is now an open standard that requires no royalties for its implementation and use. See <https://en.wikipedia.org/wiki/PDF>.
- PDF viewer** A program, such as Adobe Acrobat Reader, that is used to view PDF files on-screen.
- pen position** A virtual point located on the baseline that is used to track the placement of glyphs in the horizontal axis. Conceptually, one glyph is placed at an initial pen position and then the pen position moves forward by the advance width of that glyph to a final pen position. The final pen position (adjusted for kerning and tracking) becomes the initial pen position of the next sequential glyph.
- petite caps** Small caps will often be slightly or significantly larger than the x-height of the font. Some typefaces have additional small caps which match the x-height and they are called petite caps. [TG] **Aap**
- pitch** The number of characters that fit horizontally in one inch (or, potentially, some other horizontal unit). This metric is often used with monospaced (or “fixed-pitch”) fonts. For



example, this monospaced text set in Kurinto Mono 12pt is 10-pitch, since each horizontal inch fits 10 characters.

point $\frac{1}{72}$ of an inch, or about 0.353 mm. Historically this has varied by $\approx \pm 51\%$, but in the era of digital typography, this is now a fixed measure.

Postscript A system for drawing graphics as curves, which can then be rendered on raster displays at any resolution. See <https://en.wikipedia.org/wiki/PostScript>.

proportional The attribute that characters widths are independent of one another. This could apply to proportional typeface designs, proportional text, and proportional figures (digits). The characters have varying [advance widths](#), typically based on the visual design of each character. The term “typographic” is sometimes used for “proportional”, especially in relation to typographic figures. (The image is from [TG].)



PUA **Private Use Area.** Blocks of code points that contain characters outside of the Unicode specification. These code points are in the ranges: U+E000–U+F8FF, U+F0000–U+FFFFD, and U+100000–U+10FFFD.

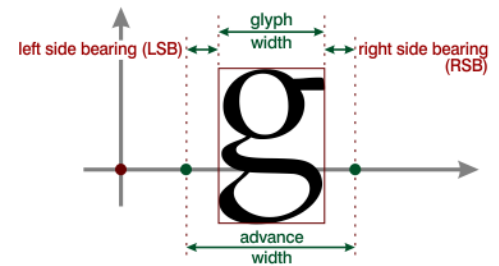
RBIBI A shorthand for “**R**egular, **B**old, **I**tallic, and **B**old **I**tallic” – the standard quartet of basic font styles that most fonts (and all Kurinto fonts) provide. Where possible, it is shown as a stylized “**RBIBI**” rather than plaintext “RBIBI”.

reflow The repositioning of text and graphic elements resulting from a change in the sizes of the elements being rendered. Changing a font in a document typically causes the whole document to reflow, because of changes in the character widths or line height. This typically causes the locations of line breaks and page breaks to change, affecting the position of images, frames, and tables. Reflow often necessitates the page layout of the entire document to be adjusted manually.

rendering engine The component responsible for converting a font to a usable format. Typically, a rendering engine will convert a stream of characters and font into an on-screen or on-printer display, often “rasterizing” the contours of the glyph outlines for the given display.

There are many in use today: Uniscribe (Windows), Core Type (Apple), FreeType (open source), to name a few. Since different rendering engines can produce significantly different results, the actual, pixel-by-pixel display can noticeably vary depending on the rendering engine in use.

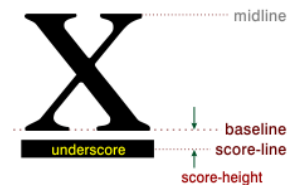
right side bearing (RSB) The distance between the right side of the glyph's width and the right side of the visual glyph design. If the glyph design exceeds the width of the glyph, the side bearing values can also be negative [TG].



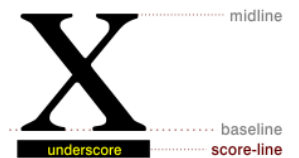
right side bearing point See [advance point](#).

RTL Right-to-Left. This can pertain to text direction or the sequence of vertical lines of text on the page – as in “TTB-RTL” for “Top-to-Bottom – Right-to-Left”.

score-height The distance from the baseline to the score-line.



score-line A virtual line located at the vertical midpoint of the underline (or “underscore”) adornment.



script A term that *The Unicode Standard* uses for a collection of characters that make up a writing system. While Unicode uses the term “script”, this document uses the more descriptive term “writing system”.

serif An ornamental aspect of a character. A serif typeface is one whose characters contain serifs (such as Times Roman or Courier). [from the Oracle Documaker, Fonts Reference]

side bearings See left side bearing and right side bearing.

small caps Small capitals (or just small caps) can be used in bicameral scripts. They use the design of capital letters, but are smaller (roughly the size of the lowercase letters) and usually replace the lowercase letters. [TG]



strike-height The distance from the baseline to the strike-line.



strike-line A virtual line located at the vertical midpoint of a strikethrough adornment.



tabular figures A set of figures (digits) that share the same [advance width](#). Also called fixed-pitch, fixed-width, or monospaced figures. (The image is from [TG].)

- tofu** A slang term for the replacement character that is shown for an undisplayable code point (character). The typical reason the character cannot be displayed is that the available fonts do not have a glyph for the code point. The actual replacement character might appear as, for example: □, □, □, or ♦.
- tracking** Uniformly increasing or decreasing the horizontal spacing between characters. Tracking provides an overall control of letterspacing. Tracking can be adjusted »to make letterspacing tighter (½pt tighter, in this case)«, »to make letterspacing looser (½ point looser in this case)«, or »really space things out (1 ½ pts)«.
- TTB** Top-to-Bottom. This can pertain to text direction or the sequence of vertical lines of text on the page – as in “TTB-RTL” for “Top-to-Bottom – Right-to-Left”. TTB text has two possible orientations: TTB/sideways (characters rotated 90°) and TTB/upright.
- TTF** **TrueType Format** – a digital font file format developed by Apple in the late 1980s. TrueType fonts are prevalent on the classic Mac OS, macOS, and Microsoft Windows operating systems. See <https://en.wikipedia.org/wiki/TrueType>.
- Type 1** A font format developed by Adobe that uses Postscript outlines to describe the shape of glyphs and adds hinting to control the rendering of glyphs on raster displays. The primary purpose of the hinting is to improve rendering on (relatively) low-resolution displays at (relatively) small point sizes. See https://en.wikipedia.org/wiki/PostScript#Font_handling.
- typeface** A coordinated design for a collection of characters. Some examples are **TIMES NEW ROMAN**, **ARIAL**, and **KURINTO TEXT**. A typeface is also called a font family.
- A particular typeface may be available in various styles, such as **BOLD**, **BOLD ITALIC**, **NARROW REVERSE OBLIQUE**, and **WIDE SMALL CAPS**.
- UBL** Upper Boundary Line. A horizontal line that represents the intended vertical boundary between a single-spaced line of text and the end of the previous line of text. See [Boundary Lines](#).
- Unicode** See [What is Unicode?](#)
- UnitsPerEm** A basic design parameter of a font that specifies the number of FUnits per em. Often abbreviated “UPM”. There are 2,048 UnitsPerEm in all Kurinto fonts.
- UTF-8** A system for representing all the possible code points of Unicode – ranging from U+0000 through U+10FFFF – in a stream of bytes. The width of each code point in UTF-8 is variable, with each character taking from 1 to 4 bytes. UTF-8 is the predominant encoding system on the World Wide Web and text documents on many systems. documents.



- UTF-16** A system that represents Unicode code points in one or two 16-bit words. Each character uses 2 or 4 bytes of data. UTF-16 is used internally by Microsoft Windows, the Java and JavaScript programming languages, and for plain-text files on Windows.
- variant** To accommodate every human language, Kurinto typefaces are segmented into font variants. Most writing systems in current use are included in the primary (or “Main”) font variant. For example, the font for the bold version of **KURINTO TEXT** that includes the Main variant is called **KURINTO TEXT BOLD**. Historical languages (e.g. Phoenician) and constructed languages (e.g. Klingon) are included in the “Aux” variant – **KURINTO TEXT AUX BOLD**. Asian writing systems with very large character sets or composition rules are included in specific font variants – for example Japanese (**KURINTO TEXT JP BOLD**), Traditional Chinese (**KURINTO TEXT TC BOLD**), and Tibetan (**KURINTO TEXT TB BOLD**). There are also variants for certain large, specialized character sets, including **MUSIC** and **UFI** (including the Medieval Unicode Font Initiative and Cyrillic Font Initiative).
- WGL4** **Windows Glyph List 4.** A set of 657 characters that can represent the many European languages. It is sometimes referred to at the “Pan-European character set”. It includes characters for Albanian, Bosnian, Bulgarian, Croatian, Czech, English, Estonian, German, Hungarian, Latvian, Lithuanian, Macedonian, Modern Greek, Polish, Romanian (except Ș, ș, Ț, and ț – U+0218–U+021B), Russian, Serbian, Slovak, Slovene, and Turkish. It also contains a limited set of box-drawing, block elements, and mathematical symbols. It does not contain characters for Arabic, Hebrew, Thai, Vietnamese, or any language that uses logographic characters.
- writing system** A collection of characters that are used by a set of languages. The languages that use a writing system might only use a subset of the characters. For example, both Russian and Ukrainian use the Cyrillic writing system, but only Ukrainian uses the character U+20B4 € HRYVNIA SIGN.
- Note that the general term “writing system” includes additional information about how the language(s) are written (e.g. “right-to-left” and how to place diacritics). However, in this document, we focus on just the characters.

x-height The distance from the baseline and the midline.



Appendix 1 – The Core Character Set

Each of the Core variant fonts has a set of 810 characters from the Core Character Set. The set of characters is a union of character sets from these sources:

- The WGL4 character set, retrieved from https://en.wikipedia.org/wiki/Windows_Glyph_List_4 on January 22, 2020. WGL4 was designed by Microsoft to cover ranges of characters for Eastern Europe, Cyrillic, ANSI, Greek, and Turkish.⁵⁹
- *Recommended Characters for Non-Roman Fonts*, by NRSI (the Non-Roman Script Initiative) of SIL International, retrieved from <https://scriptsource.org/entry/gg5wm9hhd3> on February 28, 2020. See also subsequent comments and updates on the cited web page that modify the recommendations of the posted document.
- The Core Latin glyph set from *Google Fonts 2016 Glyph Sets*, retrieved from <https://github.com/googlefonts/gftools/> on February 19, 2020.
- A few additional characters in the BMP that I consider useful.
- Two ligatures in the Private Use Area (PUA) that have become widely used standards. See Note 4 below.
- A block of additional characters in the Supplementary Private Use Area-B that provide a range of capabilities. See [Font Identification](#) and [Visual Font Metrics](#) for a description of these characters.

Note that characters from the Unicode IPA Extensions block are not included in the Core character set. This is based on the SIL / NRSI recommendation: Since the IPA Extensions block is designed for linguistic phonetics, individual characters should not be included in a core font – the entire block should be sourced from a single font so that the characters work together.

The core character set supports these languages: **Latin:** Afrikaans, Albanian, Alsatian, Arumanian, Asturian, Basque, Bosnian, Breton, Catalan, Cebuano, Chichewa, Cornish, Corsican, Croatian, Czech, Danish, Dutch, English, Esperanto, Estonian, Faroese, Filipino, Finnish, French, Frisian, Friulian, Gaelic, Gagauz (Latin), Galician, German, Greenlandic, Hungarian, Icelandic, Indonesian, Irish, Italian, Karelian, Ladin, Latin (Lingua Latina), Latvian, Lithuanian, Luba, Maltese, Moldavian (Latin), Norwegian, Occitan, Polish, Portuguese, Retho-Romance, Romanian, Sámi (Lule), Sámi (Northern), Sámi (Southern), Serbian, Slovak, Slovenian, Sorbian, Spanish, Swahili, Swedish, Turkish, Turkmen (Latin), Vepsian, Welsh, Wolof, Zulu. **Cyrillic:** Agul, Avar, Balkar, Belarusian, Bulgarian, Chechen, Erzya, Gagauz, Ingush, Karachay, Khvarshi, Komi, Komi-Permyak, Lezgian, Macedonian, Moldavian, Nenets Tundra, Ossetian, Russian, Rutul, Serbian, Ukrainian. **Greek.**

The roadmap charts on the next four pages identify each character in the core font. Except where noted, characters can be cut from these charts and pasted into your document.

⁵⁹ Windows code pages 1250 (Central European), 1251 (Cyrillic), 1252 (Western European), 1253 (Greek), 1254 (Turkish), as well the MS-DOS glyphs from code page 437. WGL4 is equivalent to the Win95 character set.

Character Roadmap of All **Core** Variant Fonts

U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Unicode Block
002_	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	Basic Latin
003_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
004_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
005_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	
006_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
007_	p	q	r	s	t	u	v	w	x	y	z	{		}	~		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Latin-1 Supplement
00A_	NB SP	ı	ç	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯	
00B_	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	
00C_	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	
00D_	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	
00E_	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	
00F_	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	
010_	Ā	ā	Ă	ă	Ą	ą	Ć	ć	Ĉ	ĉ	Č	č	Ď	ď			Latin Extended-A
011_	Đ	đ	Ē	ē	Ĕ	ĕ	Ė	ė	Ę	ę	Ě	ě	Ĝ	ĝ	Ğ	ğ	
012_	Ġ	ġ	Ģ	ģ	Ĥ	ĥ	Ħ	ħ	Ĩ	ĩ	Ī	ī	Ĵ	ĵ	Ķ	ķ	
013_	İ	ı	IJ	ij	Ĵ	ĵ	Ķ	ķ	κ	Ĺ	ĺ	Ł	ł	Œ	œ	Œ	
014_	Ł	ł	Œ	œ	Œ	œ	Œ	œ	Œ	œ	Œ	œ	Œ	œ	Œ	œ	
015_	Œ	œ	Œ	œ	Œ	œ	Œ	œ	Œ	œ	Œ	œ	Œ	œ	Œ	œ	
016_	Š	š	Ť	ť	Ŧ	ŧ	Ũ	ũ	Ū	ū	Ŭ	ŭ	Ű	ű			
017_	Ů	ů	Ű	ű	Ų	ų	Ŷ	ŷ	Ÿ	Ž	ž	Ž	ž	Ž	ž	ſ	
019_			f														Latin Extended-B
01F_											Á	á	Ą	ą	Ǽ	Ǿ	
02B_																	Spacing Modifier Letters
02C_																	
02D_	ˆ																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
A	White = core characters. You can usually cut these characters and paste them into your document.																
space	Blue indicates a special spacing or separator character.																
																	= character not included

Character Roadmap of All Core Variant Fonts - page 2

U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Unicode Block	
034_																CGJ	Comb. Diacritical Marks	1
038_					'	ˆ	Α	·	Έ	Η	Ι		Ό		Υ	Ω	Greek and Coptic	
039_	ι	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο		
03A_	Π	Ρ		Σ	Τ	Υ	Φ	Χ	Ψ	Ω	İ	ÿ	ά	έ	ή	ί		
03B_	Ů	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο		
03C_	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	İ	ÿ	ό	ύ	ώ			
040_	È	Ë	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	Ў	Ѓ	Ѕ	Cyrillic	
041_	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П		
042_	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я		
043_	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п		
044_	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я		
045_	è	ë	ђ	ѓ	є	ѕ	і	ї	ј	љ	њ	ћ	ќ	ў	џ	џ	Latin Extended Additional	
049_	Ґ	ґ																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
1E8_	Ẁ	ẁ	Ẃ	ẃ	Ẅ	ẅ												
1EF_			Ỳ	ỳ														
200_	NO SP	MO SP	EN SP	EM SP	3/M SP	4/M SP	6/M SP	F SP	P SP	TH SP	H SP	ZW SP	ZW NJ	ZW J	LRM	RLM	General Punctuation (note 1)	
201_	-	-	—	—	—		=	'	'	,	\	"	"	"	"			
202_	†	‡	•			...	•	L SEP	P SEP	LRE	RLE	PDF	LRO	RLO	NNB SP			
203_	‰		'	"						<	>		!!		—			
204_					/													
205_																MM SP		
206_	WJ	f0	×	,	+		LRI	RLI	FSI	PDI	ISS	ASS	IFS	AFS	NADS	NODS	67	
207_					4											n	Superscripts & Subscripts	2
20A_				₣	£				Pt				€				Currency Symbols	4
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
A	White = core characters. You can usually cut these characters and paste them into your document.																	
space	Blue indicates a spacing or separator char. The abbrev. in red indicates the function of the char. See below.																= character not included	
	Tan characters serve a special function. These will not usually display as they are shown on this chart.																= unassigned code point	

Note 1: The 17 spacing characters are: En Quad, Em Quad, En Space, Em Space, Three-per-Em Space, Four-per-Em Space, Six-per-Em Space, Figure Space, Punctuation Space, Thin Space, Hair Space, Zero Width Space, Line Separator, Paragraph Separator, Narrow No-Break Space, Medium Mathematical Space, and Word Joiner.

Character Roadmap of All **Core** Variant Fonts - page 3

U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Unicode Block
210_																	Letterlike symbols
211_																	
212_																	
215_																	
219_																	Arrows
21A_																	
220_																	Mathematical Operators
221_																	
222_																	
224_																	
226_																	
230_																	Miscellaneous Technical
231_																	
232_																	
242_																	Control Pictures
250_																	Box Drawing
251_																	
252_																	
253_																	
255_																	
256_																	
258_																	Block Elements
259_																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	U+2423 is a standard character for showing a visible space. It takes the same width as the space character (U+0020).																= character not included
	White = core characters. You can usually cut these characters and paste them into your document.																= unassigned code point

Character Roadmap of All Core Variant Fonts - page 4

U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Unicode Block	
25A_	■	□									▪	▫	▬				Geometric Shapes	
25B_			▲								▶	◻	▼					
25C_				◀							◊	◯	⊙			●		
25D_								◻	◐									
25E_						○												
263_											☺	☹	☀				Miscellaneous Symbols	
264_	♀		♂															
266_	♠			♣		♥	♦				♪	♫						
F00_		fi	fl														Private Use Area (note 4)	
FB0_		fi	fl														Alphabetic Presentation Forms	
FE0_	VS 1	VS 2	VS 3	VS 4	VS 5	VS 6	VS 7	VS 8	VS 9	VS 10	VS 11	VS 12	VS 13	VS 14	VS 15	VS 16	Variation Selectors	
FEF_	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Arabic Presentation Forms-B	
FFF_											IA A	IA S	IA T	OBJ	↻		Specials (note 2)	
10FA00 – 10FE7F are the ACF (Alt Caps and Figures) block ... see Appendix 2																		
10FF0_	Kurinto Font Folio 2.195																Kurinto (note 3)	
10FF1_																		
10FF2_																		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Total chars with ACF: ###	
A	White = core characters. You can usually cut these characters and paste them into your document.																	Total chars without ACF: 757
zwid	Blue indicates a spacing or separator character. The abbreviation in red indicates the function of the character.																	
	Tan characters serve a special function. These will not usually display as they are shown on this chart.																	

Note 2: The three spacing characters are the InterLinear Annotation Anchor, Separator, and Terminator characters used internally for Japanese Ruby (furigana).

Note 3: The Kurinto block provides characters for font identification, visual metrics, and access to characters that would otherwise not be visible since they serve special functions.

Note 4: The Private Use Area (PUA) is generally not used for standard characters. However, the two ligatures at U+F001 and U+F002 – copies of the ligatures at U+FB01 and U+FB02 – have become so widely used in the design community that they have become a de-facto standard. It has been commented that “Populating the fonts with different glyphs on these positions is support suicide and will make the fonts useless to a professional design community.”⁶⁰ For these reason, I have included them in the Core Character Set.

⁶⁰ Comment by Bruno Maag on October 1, 2010. See <https://bugs.launchpad.net/ubuntu-font-family/+bug/652154>, retrieved April 30, 2020. See also <https://scripts.sil.org/PUACharsInMSSoftware>.

Appendix 2 – The Alt Caps and Figures Character Set

Some fonts support an “ACF” (Alt Caps and Figures) block of characters in the high Private User Area. These 911 characters provide Small Caps, Petite Caps, and Titling Caps for Latin, Greek, and Cyrillic capital letters. They also provide 8 styles of figures.

These characters are available by OpenType layout features. They can also be accessed directly by the code points shown in the tables on the next three pages:

Character Roadmap of the **Alt Caps and Figures** (ACF) Block

U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	SPUA-B Block
10FA0_	space	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	Small Caps
10FA1_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
10FA2_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
10FA3_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	
10FA4_	`	{		}	~	ı	¢	£	¤	¥	¦	§	¨	©	®	q	
10FA5_	¼	½	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	
10FA6_	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	
10FA7_	Ü	Ý	Þ	ß	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	
10FA8_	Ẽ	Ė	Ĝ	Ğ	Ĥ	Ħ	Ĩ	Ĵ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	
10FA9_	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	
10FAA_	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	
10FAB_	Ÿ	Ž	ž	Ž	Ž	ž	Ž	ž	Ž	ž	Ž	ž	Ž	ž	Ž	ž	
10FAC_	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	
10FAD_	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	Ϊ	Ϋ	Ϊ	Ϋ	Ϊ	Ϋ	Ϊ	Ϋ	
10FAE_	⅛	⅜	⅝	⅞													
10FB0_	space	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	Petite Caps
10FB1_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
10FB2_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
10FB3_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	
10FB4_	`	{		}	~	ı	¢	£	¤	¥	¦	§	¨	©	®	q	
10FB5_	¼	½	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	
10FB6_	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	
10FB7_	Ü	Ý	Þ	ß	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	
10FB8_	Ẽ	Ė	Ĝ	Ğ	Ĥ	Ħ	Ĩ	Ĵ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	Ĳ	
10FB9_	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	Ł	ł	
10FBA_	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	Ś	ś	
10FBB_	Ÿ	Ž	ž	Ž	Ž	ž	Ž	ž	Ž	ž	Ž	ž	Ž	ž	Ž	ž	
10FBC_	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	
10FBD_	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	Ϊ	Ϋ	Ϊ	Ϋ	Ϊ	Ϋ	Ϊ	Ϋ	
10FBE_	⅛	⅜	⅝	⅞													
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	space	Blue = space characters					Σ	Red = Greek				Grey = character not included					

Character Roadmap of the **Alt Caps and Figures** (ACF) Block - Page 2

U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	SPUA-B Block
10FC0_	space	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	Titling Caps
10FC1_	0	1	2	3	4	5	6	7	8	9	:	:	<	=	>	?	
10FC2_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
10FC3_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	
10FC4_	`	{		}	~	ı	¢	£	¤	¥	¦	§	¨	©	®	q	
10FC5_	¼	½	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	
10FC6_	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	
10FC7_	Ü	Ý	Þ	ß	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	
10FC8_	Ě	Ě	Ĝ	Ĝ	Ĝ	Ĝ	Ĥ	Ĥ	Ĥ	Ĥ	Ĥ	Ĥ	Ĥ	Ĥ	Ĥ	Ĥ	
10FC9_	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	Ĺ	
10FCA_	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š	
10FCB_	Ÿ	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	Ž	
10FCC_	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	
10FCD_	Τ	Υ	Φ	Χ	Ψ	Ω	İ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	
10FCE_	⅛	⅜	⅝	⅞													
10FD0_	È	Ë	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	Ў	Ѓ	Ѓ	Cyrillic Small Caps
10FD1_	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	
10FD2_	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	
10FD3_	Г																
10FD4_	È	Ë	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	Ў	Ѓ	Ѓ	Cyrillic Petite Caps
10FD5_	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	
10FD6_	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	
10FD7_	Г																
10FD8_	È	Ë	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	Ў	Ѓ	Ѓ	Cyrillic Titling Caps
10FD9_	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	
10FDA_	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	
10FDB_	Г																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	space	Blue = space characters					Σ	Red = Greek			Ж	Purple = Cyrillic					Grey = character not included

Character Roadmap of the Alt Caps and Figures (ACF) Block - Page 3																			
U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	SPUA-B Block		
10FE0_	0	1	2	3	4	5	6	7	8	9							Proportional Lining Figures		
10FE1_	o	1	2	3	4	5	6	7	8	9							Proportional OldStyle Figures		
10FE2_	0	1	2	3	4	5	6	7	8	9							Proportional Hybrid Figures		
10FE3_	0̄	1̄	2̄	3̄	4̄	5̄	6̄	7̄	8̄	9̄							Proportional Overscore Figures		
10FE4_	0	1	2	3	4	5	6	7	8	9							Tabular Lining Figures		
10FE5_	o	1	2	3	4	5	6	7	8	9							Tabular OldStyle Figures		
10FE6_	0	1	2	3	4	5	6	7	8	9							Tabular Hybrid Figures		
10FE7_	0̄	1̄	2̄	3̄	4̄	5̄	6̄	7̄	8̄	9̄							Tabular Overscore Figures		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
												Grey = character not included							

911

Appendix 3 – Ligatures

For a demonstration of all the ligatures in each Kurinto font, see the Specimen Book PDF document included in each distribution package.