

---

## stixfonts

OpenType Unicode fonts for Scientific, Technical, and Mathematical texts

### Overview

- See <https://www.stixfonts.org/> for background on the the STIX Fonts project.
- Download zip files with some or all of the fonts.
- View code charts for the
  - Math,
  - Regular text, fonts.

### Type 1 fonts (STIX 2.0.0 only)

The STIX Two fonts are OpenType fonts and are meant to be used in that format. For the benefit of LaTeX users who are unable to use XeTeX or luaTeX, we have also provided version 2.0.0 of the STIX fonts as a set of TFM files and Type 1 fonts.

Note that **no further updates** are planned to the Type 1 distribution; future development efforts will focus on improving the OpenType fonts.

### About the STIX fonts.

The Scientific and Technical Information eXchange (STIX) fonts are intended to satisfy the demanding needs of authors, publishers, printers, and others working in the scientific, medical, and technical fields. They combine a comprehensive Unicode-based collection of mathematical symbols and alphabets with a set of text faces suitable for professional publishing. The fonts are available royalty-free under the SIL Open Font License.

Version 2 of the STIX fonts, now known as “STIX Two”, is a thorough revision undertaken by the renowned type house Tiro Typeworks Ltd. (<https://www.tiro.com>). The STIX Two fonts consist of one Math font, two variable text fonts (STIXTwoTextVF-Roman and STIXTwoTextVF-Italic), and eight static text fonts (Regular, Italic, Medium, Medium Italic, SemiBold, SemiBold Italic, Bold, and Bold Italic) derived from the variable fonts. Together, they provide a uniform set of fonts that can be used throughout the production process, whether that be a traditional print-only process, an entirely electronic one, or a combination of the two.

---

The STIX project began through the joint efforts of American Mathematical Society (AMS), American Institute of Physics (AIP), American Physical Society (APS), American Chemical Society (ACS), The Institute of Electrical and Electronic Engineers (IEEE), and Elsevier. These companies are collectively known as the STI Pub companies.

## **A Fresh Take on Times Roman**

The original version of STIX was based on Times Roman, which has now been updated for the digital age.

As is well known, Times Roman was originally intended for printing the *London Times*. What is not generally appreciated is that the production quality of the *Times* was atypically high: It was printed on unusually high-quality paper on presses that operated more slowly than most newspaper presses. This allowed for the design of a typeface that could exploit this level of care: serifs could be much finer and counters (enclosed areas such as that in the lowercase e) could be much smaller than in other newspaper typefaces. These features of the font have not always fared well in less exacting environments. At the same time, a notable quirk of the Times Roman family is that the bold font is, in many respects, strikingly dissimilar to the roman font.

Tiro Typeworks explain their approach to updating the Times Roman basis of STIX as follows:

“Our principal goal in approaching STIX Two was to address several inherent deficiencies in the Times New Roman model as well as expand the typographic features. This process necessarily involved diverging somewhat from Times as familiar to people who have only known the common digital versions, while simultaneously restoring to that typeface aspects of the size-appropriate design characteristics that made it so successful in newspaper, book, and journal publishing in its metal type incarnation. The essential ‘Times-ness’ remains, but are with greater harmonisation of style across the family.

“Most digital versions of Times have been based on an optical size model that appears too light and fine when scaled down to typical text sizes. In the design of STIX Two, we went back to specimens of size-specific designs from the metal era, and adapted proportions, weights, and spacing of the 10pt and 12pt designs. The oft-noted mismatch between the style of different weights of Times has been resolved with a new bold design that matches the construction of the regular weight.”

## **Font implementation decisions**

- The STIX fonts do not contain fixed-width or sans serif text faces.

- The sans serif, fraktur, script, etc., alphabets in Plane 1 (U+1D400-U+1D4FF) are intended to be used only as technical symbols.
- These fonts are designed to support left-to-right typesetting in Latin-based scripts, with additional support for Greek and Cyrillic text. Extensions to support other writing directions have been considered, but are currently deemed to be outside the scope of the STIX project.

### Note to TeX users

These fonts have been tested with both XeTeX and luaTeX with good results. For best results, XeTeX users will want to use version 0.999992 or later of XeTeX, which ships with TeXLive 2020. This version fixes a number of bugs that were present in earlier versions. Our thanks go out to Jonathan Kew and Khaled Hosny for their generous help in identifying and fixing these bugs. LaTeX users should also make sure they have the latest version of the amsmath package.

### Summary of OpenType Features and Scripts

Further details these features can be found in the font charts.

The text fonts implement the following OpenType script tags:

1	Regular	Bold	Italic	BoldItalic	
2					
3	DFLT	DFLT	DFLT	DFLT	Default
4					
5	cyr1	cyr1	cyr1	cyr1	Cyrillic
6			cyr1.MKD	cyr1.MKD	Cyrillic/Macedonian
7			cyr1.SRB	cyr1.SRB	Cyrillic/Serbian
8					
9	grek	grek	grek	grek	Greek
10					
11	latn	latn	latn	latn	Latin
12	latn.LTH	latn.LTH	latn.LTH	latn.LTH	Latin/Lithuanian
13	latn.ROM	latn.ROM	latn.ROM	latn.ROM	Latin/Romanian
14	latn.TRK	latn.TRK	latn.TRK	latn.TRK	Latin/Turkish

and the following features

1	c2sc	Small Capitals from Capitals
2	<b>case</b>	Case-Sensitive Forms
3	ccmp	Glyph Composition/Decomposition
4	dnom	Denominators
5	frac	Fractions
6	kern	Kerning
7	liga	Standard Ligatures -- latn only

---

8	locl	Localized Forms	-- latn.ROM and Italic/BoldItalic cyrl.MKD only
9	numr	Numerators	
10	onum	Oldstyle Figures	
11	pnum	Proportional Figures	
12	smcp	Small Capitals	
13	subs	Subscript	
14	supr	Superscript	

All four text fonts also support the following Character Variants:

1	cv01	U+019B Lambda with horizontal, not slanted stroke -- latn only
2	cv02	U+0264 Rams horn with serifs -- latn only
3	cv03	U+2423 OPEN BOX curved instead of straight

In addition, the Italic and BoldItalic faces support the following Stylistic Variants:

1	ss01	Replace two-story g by hooked g -- Italic/BoldItalic only
2	ss02	Upright parens, brackets, and braces -- Italic/BoldItalic only

STIX Two Math implements the following font features:

1	ccmp	Glyph Composition/Decomposition
2	dtls	Dotless forms of i and j
3	flac	Flattened accents
4	ssty	Math Script style alternates

and the following Character Variants (note the different meaning of cv03 compared to the text fonts):

1	cv01	U+019B Lambda with horizontal, not slanted stroke -- latn only
2	cv02	U+0264 Rams horn with serifs -- latn only
3	cv03	Replace U+2205 EMPTY SET by an oblate form
4	cv04	Replace U+2216 SET MINUS by a smaller form

and the following Stylistic Sets (again, note that ss01 and ss02 have different meanings compared to the text fonts):

1	ss01	Stylistic Set 1 -- Math chancery to roundhand ( $\backslash\mathrm{cal} \rightarrow \backslash\mathrm{mathscr}$ )
2	ss02	Stylistic Set 2 -- Alternate italic forms: g, u, v, w, z
3	ss03	Stylistic Set 3 -- Horizontal crossbar variants
4	ss04	Stylistic Set 4 -- Minute, second and primes to <b>long</b> variants
5	ss05	Stylistic Set 5 -- Short arrow variants
6	ss06	Stylistic Set 6 -- Short/narrow variants
7	ss07	Stylistic Set 7 -- Alternate math symbols (product, summation, etc)
8	ss08	Stylistic Set 8 -- Upright integral variants; XITS compatible
9	ss09	Stylistic Set 9 -- Vertical slash variants; XITS compatible

---

```
10 ss10 Stylistic Set 10 -- Diagonal greater/lesser combination
    variants
11 ss11 Stylistic Set 11 -- Long slash not-equal combination variants
12 ss12 Stylistic Set 12 -- Low contrast (sans-like) variants
13 ss13 Stylistic Set 13 -- Horizontally flipped sine wave glyph
14 ss14 Stylistic Set 14 -- Tall variants
15 ss15 Stylistic Set 15 -- Slab serif symbol variants
16 ss16 Stylistic Set 16 -- Circled operator variants
17 ss20 Stylistic Set 20 -- Miscellaneous variants
```

## Build instructions

After cloning the project, the fonts can be built using the `build.sh` script (use `--verbose` option for more detailed build log):

```
1 $ ./build.sh
```

This may take several minutes to complete. The first time the script is called, it will create a Python virtual environment that will be also used for subsequent builds. Each time the script is called, the fonts will be rebuilt from scratch. The built fonts will be in `build` subdirectory, and should be manually copied and committed to `fonts` subdirectory.

## Notes on source formats and build process

The design masters for the STIX Two Text fonts are the `.vfb` files, a json source format used by FontLab 7. These files contain the glyph outlines, spacing, mark anchors, kerning and associated classes, font info, and variable design space info. Changes or additions to any of these things should be made in the `.vfb` files.

The build script used to generate font files uses the `.ufo` and `.designspace` files, not the `.vfb` sources directly. These files can be exported from FontLab 7 using the default export profile for 'DesignSpace + UFO'.

The `.ren` files are glyph name management files used by the build script to manage the relationship of development names in the sources to the build names used in the post or CFF tables of the fonts.

Because of issues with editing and managing OpenType Layout GPOS in variable font sources, the OTL projects for the STIX Two Text fonts are built in Microsoft's Visual OpenType Layout Tool (VOLT). This means changes to OTL, including updates to mark anchors and kerning implemented in the `.vfb` sources need to be passed through VOLT, updated in the `.vtp` VOLT project files, and compiled in `.input.ttf` which are then used by the build script as a source for the OTL tables in the fonts.

---

Obviously, any changes or extension to the glyph set in the .vfj design sources needs to be reflected in each of the other sources used in the build process: in the .ufo files, the .ren file glyph name lists, and especially in the .input.ttf files and .vtp project files. Fresh .input.ttf files can be exported from FontLab 7, opened in VOLT, and the .vtp project files imported and updated.

Note that if changes or updates are made to mark anchors or kerning or associated classes in the .vfj sources, these need to be converted to VOLT format and imported into the projects, replacing or updating existing VOLT lookups and groups. This can be done using the vfj-to-volt.py tool.

The revised .vtp files should then be exported for future use, and the .input.ttf fonts *shipped* from VOLT (this is important, because although the fonts will work if just compiled and saved in VOLT, they will contain private VOLT source tables and unregistered OTL features that will be then end up in the fonts generated by the build script; so use the ‘Ship Font’ option in VOLT and overwrite the .input.ttf file (save a copy with the VOLT project, if you like, but so long as you remembered to export the updated .vtp you can always reimport as needed)).

**IMPORTANT** : the STIXTwoMath-Regular.input.ttf file is also the source for the MATH table and cmap table in the final font build. Care must be taken to preserve or extend these as necessary in this file when updating OpenType Layout or other aspects of the font.

Once all the source files are ready, run the **build.sh** as described above. The build script describes what it is doing as it runs, and verbose mode can be used to get more detail. In overview, this is what it does:

1. Pre-process the UFO files to:
  - a) remove all features and kerning groups from the UFOs; b) rename the glyphs to match the TTFs (otherwise the binary tables can’t be grafted in with FontTools easily); c) extract the binary tables and add them under data/com.github.fonttools.ttx/ in the UFO font where ufo2ft expects them; d) save the modified files in build/masters to keep the sources unchanged.
1. Build variable font with fontmake from build/masters UFOs.
2. Build binary masters with fontmake (needed for the next step) from UFOs.
3. Build static fonts with fontmake from UFOs, but telling it to interpolate OTL tables from the binary masters.
4. Post-process the fonts to fix the name tables and other final touchups.