

mtp2otf

Conversion and validation specification

1 Scope

mtp2otf converts locally supplied MathTime™ Professional 2 (MTPro2) Type 1 fonts and TeX metrics into a single-face OpenType MATH font. The Full and Lite editions have distinct installation identities. CFF-based OTF is the canonical generated font; TrueType-based TTF is derived from it for Word. Both formats retain the OpenType math layout data.

The interface to TeX engines is `unicode-math`. There is no companion package, Lua callback, or runtime-generated TeX sidecar. The converter does not resolve TeX trees, download inputs, install dependencies, or install generated fonts. Full is the default. Missing Full inputs never select Lite automatically.

2 Inputs and build interface

2.1 Source staging

Place the source files directly in `mtp2/`. Each active source font requires its PFB and TFM pair; `mtp2.sty` supplies package policy. Full additionally requires `umt2ms.fd`, `umt2mf.fd`, `umt2bb.fd`, and `umt2hrb.fd`. Lite does not require these Full-only declarations. The source registry and required-file calculation are defined in `tools/source_policy.py`. Empty inputs receive setup guidance; partially supplied inputs receive a missing-file list.

The Lite source set contains the text, script, and scriptscript sizes of `mt2mi`, `mt2sy`, and `mt2mb`, plus `mt2exa`, `mt2exe`, `mt2exf`, `mt2exg`, `mt2xl`, and `mt2xxx1`. The three optical-size suffixes are `t`, `s`, and `f`. Full includes the additional math alphabets and symbol sources in the registry.

2.2 Roman donors

The default inputs in `donors/` are `NimbusRoman-Regular.otf` and `NimbusRoman-Bold.otf` from Artifex urw-base35. Another static CFF OTF or TrueType TTF pair can be selected explicitly. Both faces must be upright, proportional, from the same family, and have the required Latin and digit coverage and appropriate weights. Coordinates and advances are normalized to the output units per em.

Regular supplies upright Latin and permitted additional text characters; Bold supplies mathematical bold Latin and digits. Mathematical italic, bold italic, and non-Latin-shaped Greek come from MTPro2. Optional donor text coverage can vary. A donor must not fill missing Full-only math semantics in Lite.

2.3 Commands and configuration

```
mkdir -p mtp2 donors
# Copy the source fonts, declarations, and donors.
```

```
./build.sh
./build.sh --edition=lite
./build.sh --roman-regular=/path/to/Regular.otf \
--roman-bold=/path/to/Bold.otf
```

The public options are `--edition`, `--roman-regular`, and `--roman-bold`; help is available with `--help`. Unknown options and abbreviated option names are rejected. Relative donor paths are resolved from the caller’s working directory. All project directories are resolved from the location of `build.sh`.

Python 3.10 or later with `fontTools`, `FontForge` with Python 3.10 or later support, and `tftopl` are required. Dependency and input failures stop the build early. Users create and populate the input directories; the build creates only its working and output directories.

3 Working state and final outputs

Location	Purpose
<code>mtpro2/</code>	Local Type 1 fonts, TeX metrics, and declarations.
<code>donors/</code>	Local Regular and Bold Roman donors.
<code>build/full/</code> , <code>build/lite/</code>	Edition-specific intermediate fonts, converted PL files, calculated values, and one diagnostic log.
<code>out/</code>	Validated OTF and TTF deliverables only.

Full produces `MTPro2Math.otf` and `MTPro2Math.ttf`. Lite produces `MTPro2MathLite.otf` and `MTPro2MathLite.ttf`. Both editions coexist in `out/`. No font is built directly into this directory.

After every required check succeeds, each validated candidate is copied to a temporary file beside its destination and renamed into place. This avoids exposing a partially copied individual font. It is not a two-file filesystem transaction: an interruption during final promotion can leave a previously validated OTF/TTF pair from different builds. Run the build again to replace the pair together in the normal sequence.

Input, construction, conversion, or validation failure leaves previous outputs intact. Their presence does not establish that the latest build succeeded: use the command’s exit status. Full and Lite may be built in sequence; simultaneous builds of the same edition are not supported. The entire `build/` directory is disposable and can be regenerated from the local inputs.

3.1 Progress and diagnostics

Normal stdout contains one line: `building MTPro2 Math`, or `building MTPro2 Math Lite`. Successful audits are silent. An unsuccessful subprocess reports its captured diagnostics on stderr. FontForge output and warnings remain available in `build/<edition>/build.log` even when the subprocess succeeds. The log is replaced for each new build that reaches this stage, rather than accumulated across runs.

Exit status is zero on success, two for command usage and preflight failures handled by the frontend, and one for build or validation failure. There is no automatic installation, clean command, verification bypass, or verbose-mode configuration.

4 Conversion contracts

4.1 Source metrics and ordinary spacing

An untransformed MTPro2 glyph retains TFM CHARWD as its horizontal advance and CHARIC as its MATH italic correction. The ordinary adjacency adjustment is

$$\text{CHARIC}(\text{left}) + \text{TFMkern}(\text{left}, \text{right}).$$

It is encoded in cumulative GPOS layers. A missing TFM pair contributes zero, not a reason to discard italic correction. Skewchar-right pairs are excluded from ordinary kerning according to the local declarations. Script, prime, and accent placement are separate layout paths.

For source math alphabets, top-accent attachment is based on

$$\frac{\text{CHARWD} + \text{CHARIC}}{2} + \text{skewkern}.$$

Combining accents and synthesized forms follow their own coordinate transformations. Widths and attachment coordinates are not interchangeable.

4.2 Optical sizes, aliases, and variants

Text, script, and scriptscript source outlines are exposed through OpenType math substitutions. Full-only option alphabets preserve their optical-size reachability in both supported feature application orders. Latin-shaped Greek aliases share glyph IDs with their canonical donors; the aliases do not duplicate outlines or acquire separate metrics.

TFM larger-size chains, extensible recipes, and local outline geometry determine variant and assembly construction. Repeatable parts retain both connector endpoints. Brace cusp geometry follows the source halves; the cusp inside a part is distinct from an overlap between assembly parts.

4.3 Primes and compatibility forms

The public prime forms and contextual substitutions provide the retained raw-prime and superscript behavior. Their padding is derived from local outline geometry. Scriptscript prime alternates use the native optical source forms, with their corresponding source widths. Backprime follows its own source identity. These are conversion policies, not a claim that every prime transformation is a direct TFM assignment.

Public vector and bar accents are fixed. Private wide forms are distinct: U+E286 supplies finite variants, while U+E287 supplies an assembly. Standard commands do not automatically select these private paths. Swash z remains available through `cv01` or U+E000 where supported; there is no default swash or Word command registration.

4.4 MATH constants and provenance

`tools/math_constants.py` calculates the MATH constants from local source metrics and explicitly retained conversion policies. `tools/source_policy.py` extracts package and font-definition policy and the geometry needed by synthesized forms. The calculation covers the 56 MathConstants fields and the separate common connector overlap.

Not every field has a direct TFM counterpart. Reference-derived thresholds and engine-compatibility choices are documented as such; they are not relabelled as original-source metrics. In particular, upper-limit placement retains a compatibility policy rather than a universal one-to-one TeX parameter mapping. Next-style baseline-drop values are converted into the parent font’s coordinate scale.

The appendix gives every scalar assignment and its source classification. It describes formulas and explicit conversion policy, not evaluated local MTPro2 measurements. The same assignments apply to Full and Lite.

The symbol TFM supplies axis height and script/fraction shifts; the extension TFM supplies rule thickness and large-operator spacing. The package supplies script ratios. TeX-style gap formulas combine those inputs. OpenType-only thresholds and skewed-fraction gaps are retained migration policies rather than recovered TFM parameters. Rendered comparisons support their retention in tested engines, not universal identity with every original TeX expression.

Ordinary superscripts select the source sup2 branch and subscripts select sub2; OpenType’s common fields do not encode every TeX style branch. Upper-limit and stretch-stack upper-side assignments preserve the reviewed Word placement policy. Lower stretch-stack controls reuse the lower-limit inputs and are not claimed to be independently optimized. Baseline drops use script-size symbol parameters converted to the first-level parent’s em; nested engine scaling can differ. The flattened-accent threshold does not imply a dedicated flattened-accent substitution.

5 Validation methodology

Validation is mandatory. Reuse of parsing and identity definitions does not replace comparisons of serialized output with local source metrics. OTF establishes source semantics. Cross-format invariants transfer those guarantees to TTF; structural and naming checks run on both formats.

Check	Responsibility
Source declarations	Required source set, unique usable policy, accent semantics, and Full optical-family ordering.
Roman donors	Input type, family, weight, required coverage; output bounding boxes and advances, plus an extra-import heuristic.
Structural validator	Tables, cmap, aliases, clipping bounds, assemblies, features, prime contexts, and serialization.
Edition naming	Installation names, technical version metadata, and CFF naming constraints for both formats.
Source contract	Advances, italic correction, top accents, and ordinary IC plus TFM pair adjustments in the OTF; converted accent centers are checked against TTF outlines.
Edition capabilities	Full option-alphabet optical sizes or Lite’s required and forbidden features and code points.
Cross-format check	Glyph-ID mapping, advances and sidebearings, MATH/GSUB/GPOS/GDEF table bytes, and outline origins.
MATH values	Each calculated scalar against the serialized OTF, including the common connector overlap.

The Roman output check compares normalized bounding boxes and advances; it is not a proof of contour identity. The OTF/TTF check permits the intended cubic-to-quadratic approximation and checks origin shifts separately. Engine rendering is not proven solely by table validation.

5.1 Local evidence

`build/<edition>/source-policy.json` contains parsed policy, geometry, and input hashes. `mtp2-source-contract.json` contains the glyph relationships and source-derived records used by output audits. `math-constants.json` contains evaluated scalars and their provenance descriptions. These snapshots are working data, not extra deliverables. Repeated PASS summaries and copies of the same snapshots are not written beside the final fonts.

When changing the converter, build both editions from the same inputs before and after the change. Compare corresponding OTFs and TTFs with build timestamps and dependent checksums normalized; retain all other table differences for investigation. A layout change additionally requires focused engine tests. Identical nonvolatile output can reuse the earlier rendering evidence for that exact font content.

6 Specimens and engine limits

`word.docx` and `lualatex.tex` are editable specimens. The included PDFs show the Full edition with Nimbus donors. Word uses the TTF as its math font and Times New Roman for body text; LuaLaTeX uses the OTF and Nimbus Roman for body text. Close Word before replacing an installed font.

To rebuild the LuaLaTeX specimen, first build Full, then run `lualatex lualatex.tex` from `doc/`. The specimen needs `amsmath`, `fontspec`, `unicode-math`, and all four Nimbus Roman body styles discoverable by TeX. These specimen dependencies are not additional converter dependencies. This specification itself can be compiled with `lualatex mtp2otf.tex`, without MTPro2 fonts.

Word can segment mathematical runs before GPOS processing. LuaHBTeX may not reproduce the complete Type 1 ordinary italic-correction plus pair-kern combination. A correct font table therefore does not imply identical spacing in every engine. No Lua option or callback is added to hide these differences. Compare the included specimens with the original Type 1 output in the same layout context when distinguishing source behavior, conversion behavior, and engine behavior.

7 Publication and references

The public tree contains the shell entry point, Python tools, README, LICENSE, and documentation. Local input and working directories are not part of the archive. Publication inspection is available as `python3 tools/audit_public_source.py` on a clean source-only tree. It checks prohibited paths and selected metric patterns; it is not a general legal or confidentiality audit of arbitrary PDF contents.

Mappings are interoperability metadata created for this project by comparing TeX and Unicode semantics, local-source behavior, and rendered results. Type 1 names alone do not determine the required mappings. They are not mechanical copies of MTPro2 files.

Project-authored code and documentation use 0BSD. MTPro2, donor fonts, and external tools retain their own terms. MathTime is a trademark of Publish or Perish, Inc. This is an independent project, not affiliated with or endorsed by Publish or Perish, Inc., Personal TeX, Inc., or Artifex Software.

- Repository and implementation: <https://github.com/satom-zk/mtp2otf>
- Bug reports: <https://github.com/satom-zk/mtp2otf/issues>

A Scalar assignments

The following expressions follow `tools/math_constants.py` in its actual evaluation order. `round` is Python nearest-integer rounding (ties to even); `//` is floor division. `Fraction(a,b)` is an exact rational. Intermediate rounding is intentional and must not be algebraically reordered.

`upm` is the output units per em. `s(i)` means `round(sy[i] * upm)`, with `sy` from `mt2syt.tfm`; `e(i)` uses `mt2exa.tfm` in the same way. `theta = e(8)` and `xh = s(5).script_sy` comes from `mt2sys.tfm`. The two ratios are the local `mtpro2.sty` defaults; `_pct(r)` evaluates `round(r.numerator * 100 / r.denominator)`.

`display_operator_min_height` is the floor of the average of the separately rounded height-plus-depth values of extension TFM slots 0x50 and 0x58, in output units. The source provides the sizes; choosing their midpoint is a conversion policy. The delimited subformula threshold reuses this value and is not TeX's `deliml`.

Source classes distinguish inputs from assignment choices: MT-TFM denotes local TFM parameters; MT-STY denotes package declarations; MT-PFB denotes local outlines; OT-SUG denotes an OpenType suggested formula; TeX denotes the script-space convention. OT-MIG and MT-MIG denote explicit migration policy. Combined labels mean that source data and an assignment policy both contribute. A source label does not establish exact layout equivalence in every engine.

Values are in font units except the two script percentages and the radical-degree percentage. Numeric literals in migration rows are project policy, not a table of measurements copied from MTPro2. The initial connector-overlap value is reduced to the nonnegative integer minimum joining run when local geometry requires a smaller overlap. Both ends of repeatable parts participate; fixed outside ends do not.

Field	Expression and source class
AxisHeight	<code>s(22)</code> MT-TFM
FractionNumeratorShiftUp	<code>s(9)</code> MT-TFM
FractionNumeratorDisplayStyleShiftUp	<code>s(8)</code> MT-TFM
FractionDenominatorShiftDown	<code>s(12)</code> MT-TFM
FractionDenominatorDisplayStyleShiftDown	<code>s(11)</code> MT-TFM
FractionNumeratorGapMin	<code>theta</code> OT-SUG+MT
FractionNumDisplayStyleGapMin	<code>3 * theta</code> OT-SUG+MT
FractionDenominatorGapMin	<code>theta</code> OT-SUG+MT
FractionDenomDisplayStyleGapMin	<code>3 * theta</code> OT-SUG+MT
FractionRuleThickness	<code>theta</code> MT-TFM
StackTopShiftUp	<code>s(10)</code> MT-TFM
StackTopDisplayStyleShiftUp	<code>s(8)</code> MT-TFM
StackBottomShiftDown	<code>s(12)</code> MT-TFM

Field	Expression and source class
StackBottomDisplayStyleShiftDown	s(11) MT-TFM
StackGapMin	3 * theta OT-SUG+MT
StackDisplayStyleGapMin	7 * theta OT-SUG+MT
SuperscriptShiftUp	s(14) MT-TFM+OT-MIG
SuperscriptShiftUpCramped	s(15) MT-TFM
SubscriptShiftDown	s(17) MT-TFM+OT-MIG
SubscriptTopMax	round(Fraction(4,5) * s(5)) OT-SUG+MT
SuperscriptBottomMin	round(Fraction(1,4) * s(5)) OT-SUG+MT
SuperscriptBottomMaxWithSubscript	round(Fraction(4,5) * s(5)) OT-SUG+MT
SubSuperscriptGapMin	4 * theta OT-SUG+MT
UpperLimitBaselineRiseMin	e(9) OT-MIG+MT
UpperLimitGapMin	e(11) OT-MIG+MT
LowerLimitGapMin	e(10) MT-TFM
LowerLimitBaselineDropMin	e(12) MT-TFM
StretchStackTopShiftUp	e(9) OT-MIG+MT
StretchStackGapAboveMin	e(11) OT-MIG+MT
StretchStackGapBelowMin	e(10) OT-MIG+MT
StretchStackBottomShiftDown	e(12) OT-MIG+MT
OverbarRuleThickness	theta MT-TFM
OverbarVerticalGap	3 * theta OT-SUG+MT
OverbarExtraAscender	theta OT-SUG+MT
UnderbarRuleThickness	theta MT-TFM
UnderbarVerticalGap	3 * theta OT-SUG+MT
UnderbarExtraDescender	theta OT-SUG+MT
RadicalRuleThickness	theta MT-TFM
RadicalExtraAscender	theta OT-SUG+MT
RadicalVerticalGap	theta + theta // 4 OT-SUG+MT
RadicalDisplayStyleVerticalGap	theta + xh // 4 OT-SUG+MT

Field	Expression and source class
RadicalKernBeforeDegree	<code>round(Fraction(5,18) * upm)</code> OT-SUG
RadicalKernAfterDegree	<code>-round(Fraction(10,18) * upm)</code> OT-SUG
RadicalDegreeBottomRaisePercent	60 OT-SUG
ScriptPercentScaleDown	<code>_pct(script_ratio)</code> MT-STY
ScriptScriptPercentScaleDown	<code>_pct(scriptscript_ratio)</code> MT-STY
MinConnectorOverlap	100 OT-MIG+MT-PFB
AccentBaseHeight	477 OT-MIG
FlattenedAccentBaseHeight	656 OT-MIG
DelimitedSubFormulaMinHeight	<code>int(display_operator_min_height)</code> MT-MIG+MT
DisplayOperatorMinHeight	<code>int(display_operator_min_height)</code> MT-TFM
MathLeading	154 OT-MIG
SkewedFractionHorizontalGap	350 OT-MIG
SkewedFractionVerticalGap	102 OT-MIG
SpaceAfterScript	<code>round(Fraction(1,20) * upm)</code> TeX
SubscriptBaselineDropMin	<code>round(script_sy[19] * float(script_ratio) * upm)</code> MT-TFM+OT-MIG
SuperscriptBaselineDropMax	<code>round(script_sy[18] * float(script_ratio) * upm)</code> MT-TFM+OT-MIG