

mtp2otf

Build guide and technical reference

1 Overview

mtp2otf converts MathTime™ Professional 2 (MTPro2) Type 1 fonts and TeX metrics to OpenType MATH. It preserves the source designs, including optical script sizes, and supplies the layout data used by Unicode math engines. Upright Latin letters and additional text characters come from a separate Roman font pair.

The converter produces OTF for LuaLaTeX and TTF for Microsoft Word. Full and Lite use separate family names and can be installed together. The selected edition requires the corresponding MTPro2 source files.

No additional mtp2otf package is needed to use the generated fonts. Unicode characters and OpenType features provide access to the designs. The original mtp2 package's custom commands are not included. See `font-guide.pdf` for illustrated font usage and character variants.

2 Building the fonts

2.1 Requirements

The build requires a POSIX shell, Python 3.10 or later with fontTools, FontForge with Python 3.10 or later support, and `tftopl` from TeX Live. These commands must be on PATH.

Create `mtp2/` and `donors/` beside `build.sh`. Supply the font files separately:

- `mtp2/`: MTPro2 PFB/TFM files and `mtp2.sty`. Full also requires `umt2ms.fd`, `umt2mf.fd`, `umt2bb.fd`, and `umt2hrb.fd`.
- `donors/`: `NimbusRoman-Regular.otf` and `NimbusRoman-Bold.otf` from Artifex urw-base35.

Place the files directly in these directories. The build reports missing inputs for the selected edition.

Another Roman pair can be selected with the options below. The faces must be static CFF OTF or TrueType TTF, upright and proportional, from the same family, and have the required Latin and digit coverage and Regular/Bold weights.

2.2 Build options

The default build uses Full and minimal donor coverage:

```
./build.sh
```

Select Lite, extended donor coverage, or a different Roman pair as needed:

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

The edition and donor options can be combined. Relative donor paths are resolved from the current directory; project paths are relative to `build.sh`.

Minimal coverage uses a fixed selection of Roman characters. Extended coverage adds eligible characters available in the Regular donor, without replacing MTPro2's mathematical glyphs. It does not add Full-only features to Lite. Appendix D lists the selection rules.

2.3 Output

The generated fonts are placed in `out/`:

Edition	OTF	TTF
Full	MTPro2Math.otf	MTPro2Math.ttf
Lite	MTPro2MathLite.otf	MTPro2MathLite.ttf

Install the generated fonts manually. Errors are reported on `stderr`; generation diagnostics are recorded in `build/<edition>/build.log`. Check the command's exit status: a failed build may leave previous output files in place. Do not run concurrent builds of the same edition.

3 Using the fonts

3.1 Microsoft Word

Install the TTF and select `MTPro2 Math` or `MTPro2 Math Lite` as the equation font. For the equation text, enable *Kerning for fonts* at 1 pt and above in the Font dialog. In Microsoft 365 Word for Windows, as tested on 22 October 2026, italic correction is not applied to ordinary character spacing, so spacing may differ slightly from the original Type 1 output.

Use Word's equation editor for mathematical alphabets, accents, delimiters and operators. It does not expose the same OpenType feature controls as LuaLaTeX. The explicit feature examples in `font-guide.pdf` are for LuaLaTeX.

3.2 LuaLaTeX

Load the OTF with `unicode-math`:

```
\usepackage{unicode-math}
\setmathfont{MTPro2Math.otf}
```

For Lite, use `MTPro2MathLite.otf`. If the font is not installed, specify its directory with the `Path` option.

Use `unicode-math` commands for alphabets and symbols, and the usual math structures for scripts, fractions, accents and delimiters. See `font-guide.pdf` for input examples and visual comparisons; Appendix A lists the exact feature inputs.

Luaotfload 3.29 can omit pair kerning between script-size variants.

The font can also be used with `lua-unicode-math`.

4 License and project

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

Repository: <https://github.com/satom-zk/mtp2otf>

Issues: <https://github.com/satom-zk/mtp2otf/issues>

5 Release history

v1.0 Initial release.

v1.1 Added large glyphs, missing mathematical alphabets, and character-variant access.

v1.2 Corrected TFM kern interpretation and removed supplemental Word-oriented spacing adjustments, improving LuaLaTeX output.

v1.3 Revised MATH constants and prime conversion against the original sources, and reorganized the build tools.

v1.4 Revised Unicode mappings and standard input routes, reorganized character variants, and separated ordinary delimiter growth from optional large ready-made forms. Removed unused private-use entries.

A Character variants

The inputs below are the standard Unicode characters selected by the mathematical alphabet or symbol command. Features also follow the corresponding script-size designs. F = Full; L = Lite.

Feature	Design	Standard inputs	Edition
cv01	Swash z	U+1D467 U+1D49B	F/L F
cv02	Alternative beta and delta, upright and italic	U+03B2; U+03B4; U+1D6FD; U+1D6FF U+1D6C3; U+1D6C5; U+1D737; U+1D739	F/L F
cv03	Alternative hbar (original hslash)	U+210F	F/L
cv11	Slanted product, coproduct and sum	U+220F--U+2211	F/L
cv12	Alternative small and large conjunction signs	U+2227; U+22C0	F/L
cv21	Straight braces	U+007B; U+007D	F/L
cv22	Large prebuilt parentheses	U+0028--U+0029	F/L
cv23	Large prebuilt curly braces	U+007B; U+007D	F/L
cv24=1	Plain overbrace and underbrace	U+23DE--U+23DF	F/L
cv24=2	Curly overbrace and underbrace	U+23DE--U+23DF	F/L
cv31	Script alternatives, regular and bold	Script alternate inputs listed below	F
cv32	Fraktur alternatives, regular and bold	Fraktur alternate inputs listed below	F
cv33	Heavier upright Blackboard	Upright Blackboard Latin and digits	F
cv34	Holey Blackboard, including the standard italic forms	Upright Blackboard Latin and digits; U+213C--213F; U+2145--2149	F
cv41	Alternative proper-subset and proper-superset signs	U+228A--U+228B; U+2ACB--U+2ACC	F
cv42	Alternative less/greater-than-but-not-equal signs	U+2268--U+2269	F
cv43	Short divides and parallel signs, including their negations	U+2223--U+2226	F
cv44	Thick similar and approximately-equal signs	U+223C; U+2248	F
cv45	Alternative proportional-to, smile and frown signs	U+221D; U+2322--U+2323	F
cv46=1	Slanted-equality negated comparisons; alternative negated inclusions	U+2270--U+2271; U+2288--U+2289	F
cv46=2	Double-equality negated comparisons and inclusions	U+2270--U+2271; U+2288--U+2289	F
cv47	Alternative not-similar sign	U+2241	F
cv51	Alternative empty set	U+2205	F
cv52	Shaded spade and club	U+2660; U+2663	F/L
cv53	Alternative yen and registered signs	U+00A5; U+00AE	F
ss01	Curly Script	Regular Script alphabet	F
ss02	Curly Script alternate series	Regular Script alphabet	F

cv33 and cv34 together select the heavier Holey upright Blackboard design. cv33 leaves the Greek and standard italic Blackboard characters unchanged. cv34 selects their corresponding Holey designs.

ss01 and cv31 can be combined. ss02 selects the Curly alternate series without requiring ss01; it overrides the regular Script choices of ss01 and cv31. cv32 remains independent.

For `cv24` and `cv46`, choose one numbered value per tag; disabling the feature restores the default. `cv47` affects only U+2241 and can be combined with either `cv46` choice.

Script alternate inputs: U+2110; U+2112; U+1D49E; U+1D4A2; U+1D4AC; U+1D4AE;
U+1D4B4--U+1D4B5; U+1D4C7; U+1D4CF; U+1D4D2; U+1D4D6; U+1D4D8; U+1D4DB; U+1D4E0;
U+1D4E2; U+1D4E8--U+1D4E9; U+1D4FB; U+1D503.

Fraktur alternate inputs: U+2111; U+211C; U+1D51C; U+1D535--U+1D536; U+1D584;
U+1D59D--U+1D59E.

B Growing symbols

The table lists standard inputs with MATH size data in both editions. V and H denote vertical and horizontal growth. Var. counts the records in the prebuilt variant list; Asm. indicates an assembly. Zero variants with an assembly means that growth uses repeatable parts.

Use paired `\left` and `\right` for delimiters, `\sqrt` for radicals, and accent or over/under commands for horizontal constructions. The table describes font data; a symbol command alone need not request growth. For U+20D7, `\vec` selects a fixed accent and `\overrightarrow` requests a wide accent. `\bar` and `\overbar` use the fixed U+0304 accent; `\wideoverbar` uses U+0305.

Unicode	Input	Selection	V/H	Var.	Asm.
U+0028	<code>\lparen</code>	default	V	5	yes
U+0028	<code>\lparen</code>	<code>cv22=1</code>	V	24	yes
U+0029	<code>\rparen</code>	default	V	5	yes
U+0029	<code>\rparen</code>	<code>cv22=1</code>	V	24	yes
U+002F	<code>\mathslash</code>	default	V	5	no
U+005B	<code>\lbrack</code>	default	V	5	yes
U+005C	<code>\backslash</code>	default	V	5	no
U+005D	<code>\rbrack</code>	default	V	5	yes
U+007B	<code>\lbrace</code>	default	V	5	yes
U+007B	<code>\lbrace</code>	<code>cv23=1</code>	V	24	yes
U+007B	<code>\lbrace</code>	<code>cv21=1</code>	V	5	yes
U+007C	<code>\vert</code>	default	V	1	yes
U+007D	<code>\rbrace</code>	default	V	5	yes
U+007D	<code>\rbrace</code>	<code>cv23=1</code>	V	24	yes
U+007D	<code>\rbrace</code>	<code>cv21=1</code>	V	5	yes
U+0302	<code>\widehat {...}</code>	default	H	37	no
U+0303	<code>\widetilde {...}</code>	default	H	37	no
U+0305	<code>\wideoverbar {...}</code>	default	H	3	yes
U+030C	<code>\widecheck {...}</code>	default	H	35	no
U+2016	<code>\Vert</code>	default	V	1	yes
U+20D6	<code>\overleftarrow {...}</code>	default	H	0	yes
U+20D7	<code>\overrightarrow {...}</code>	default	H	2	yes
U+2190	<code>\leftarrow</code>	default	H	2	yes
U+2191	<code>\uparrow</code>	default	V	1	yes
U+2192	<code>\rightarrow</code>	default	H	2	yes
U+2193	<code>\downarrow</code>	default	V	1	yes
U+2194	<code>\leftrightarrow</code>	default	H	2	yes
U+2195	<code>\updownarrow</code>	default	V	1	yes
U+21A6	<code>\mapsto</code>	default	H	2	no
U+21D0	<code>\Leftarrow</code>	default	H	2	yes
U+21D1	<code>\Uparrow</code>	default	V	1	yes
U+21D2	<code>\Rightarrow</code>	default	H	2	yes
U+21D3	<code>\Downarrow</code>	default	V	1	yes
U+21D4	<code>\Leftrightarrow</code>	default	H	2	yes
U+21D5	<code>\Updownarrow</code>	default	V	1	yes
U+220F	<code>\prod</code>	default	V	2	no
U+220F	<code>\prod</code>	<code>cv11=1</code>	V	2	no
U+2210	<code>\coprod</code>	default	V	2	no

Unicode	Input	Selection	V/H	Var.	Asm.
U+2210	\coprod	cv11=1	V	2	no
U+2211	\sum	default	V	2	no
U+2211	\sum	cv11=1	V	2	no
U+221A	\sqrt {...}	default	V	5	yes
U+2223	\mid	default	V	1	yes
U+2225	\parallel	default	V	1	yes
U+222B	\int	default	V	2	no
U+222C	\iint	default	V	2	no
U+222D	\iiint	default	V	2	no
U+222E	\oint	default	V	2	no
U+222F	\oiint	default	V	2	no
U+2230	\oiint	default	V	2	no
U+2231	\intclockwise	default	V	2	no
U+2232	\varointclockwise	default	V	2	no
U+2233	\ointctrclockwise	default	V	2	no
U+22C0	\bigwedge	default	V	2	no
U+22C0	\bigwedge	cv12=1	V	2	no
U+22C1	\bigvee	default	V	2	no
U+22C2	\bigcap	default	V	2	no
U+22C3	\bigcup	default	V	2	no
U+2308	\lceil	default	V	5	yes
U+2309	\rceil	default	V	5	yes
U+230A	\lfloor	default	V	5	yes
U+230B	\rfloor	default	V	5	yes
U+23AA	\vbraceextender	default	V	1	yes
U+23B0	\lmoustache	default	V	1	yes
U+23B1	\rmoustache	default	V	1	yes
U+23DC	\overparen {...}	default	H	24	yes
U+23DE	\overbrace {...}	default	H	0	yes
U+23DE	\overbrace {...}	cv24=1	H	0	yes
U+23DE	\overbrace {...}	cv24=2	H	35	yes
U+23DF	\underbrace {...}	default	H	0	yes
U+23DF	\underbrace {...}	cv24=1	H	0	yes
U+23DF	\underbrace {...}	cv24=2	H	35	yes
U+27E8	\langle	default	V	5	no
U+27E9	\rangle	default	V	5	no
U+27EE	\lgroup	default	V	1	yes
U+27EF	\rgroup	default	V	1	yes
U+2A00	\bigodot	default	V	2	no
U+2A01	\bigoplus	default	V	2	no
U+2A02	\bigotimes	default	V	2	no
U+2A03	\bigcupdot	default	V	2	no
U+2A04	\biguplus	default	V	2	no
U+2A06	\bigsqcup	default	V	2	no
U+2A0D	\intbar	default	V	2	no
U+2A0F	\fint	default	V	2	no

The engine compares its size request with MATH AdvanceMeasurement. This is a selection dimension and need not equal the glyph's ink extent. Default delimiters and radicals use the ordinary source sequence; cv22/cv23 select larger delimiter series.

C Metrics and MATH constants

C.1 Glyph metrics

The output uses 1000 units per em. For an untransformed MTPro2 glyph, the TFM width supplies its advance and the TFM italic correction supplies its MATH italic correction. Ordinary GPOS pair adjustments store the TFM pair kern without an added italic-correction term. Skewchar pairs are excluded from

ordinary kerning; scripts, primes and accents have separate positioning data. Top-accent attachment for source math alphabets is based on

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

For untransformed source alphabets, the calculation uses the exact TFM fixed-point values and is rounded once to output units. Combining accents and synthesized forms use the corresponding coordinate transformations and geometric attachment positions. Text, script and scriptscript designs use the source optical sizes.

C.2 Constants

The following assignments define the 56 MATH constants and the common connector overlap in both editions. They combine source metrics, package script ratios and conversion policies.

The expressions follow `tools/native_math.py` in evaluation order. `round` uses nearest-integer rounding, ties to even; `//` is floor division and `Fraction(a, b)` is exact rational arithmetic. Intermediate rounding is part of each assignment.

`upm` is the output units per em. `s(i)` is `round(sy[i] * upm)`, with `sy` from `mt2sytfm`; `e(i)` uses `mt2exa.tfm`. `theta = e(8)` and `xh = s(5)`. `script_sy` comes from `mt2sys.tfm`. The script ratios are the `mtpro2.sty` defaults; `_pct(r)` is `round(r.numerator * 100 / r.denominator)`.

`display_operator_min_height` is computed from the 30 mapped text/display operator pairs in `mt2exa.tfm`. Round each height-plus-depth to output units, then take `(max(text sizes) + min(display sizes)) // 2`. The source values give `(1103 + 1203) // 2 = 1153`. The delimited-subformula threshold uses the same value. This is a conversion policy; TFM box dimensions and OpenType glyph bounds need not select the same design in every series.

`cap_height` is the TFM height of A (slot 0x41) in `mt2mit.tfm`, before conversion to output units. Where MATH provides only one shift for display and inline superscripts, the display value is used. Fields with separate display values retain both. `MathLeading` uses the median of the em-normalized Latin Modern Math, STIX Two Math and Cambria Math reference values. These fonts are not build dependencies.

Source labels: MT-TFM, TFM parameters; MT-STY, package declarations; MT-PFB, outlines; OT-SUG, OpenType suggested formulas; TeX, the script-space convention; OT-MIG and MT-MIG, conversion policies. Combined labels indicate that both source data and an assignment policy contribute.

Values are in font units except the script and radical-degree percentages. The initial connector overlap is reduced to the nonnegative integer minimum joining run when required by outline geometry. Both connector ends of repeatable parts participate; fixed outside ends do not.

Field	Expression and source class
<code>AxisHeight</code>	<code>s(22)</code> MT-TFM
<code>FractionNumeratorShiftUp</code>	<code>s(9)</code> MT-TFM
<code>FractionNumeratorDisplayStyleShiftUp</code>	<code>s(8)</code> MT-TFM
<code>FractionDenominatorShiftDown</code>	<code>s(12)</code> MT-TFM
<code>FractionDenominatorDisplayStyleShiftDown</code>	<code>s(11)</code> MT-TFM

Field	Expression and source class
FractionNumeratorGapMin	theta OT-SUG+MT
FractionNumDisplayStyleGapMin	3 * theta OT-SUG+MT
FractionDenominatorGapMin	theta OT-SUG+MT
FractionDenomDisplayStyleGapMin	3 * theta OT-SUG+MT
FractionRuleThickness	theta MT-TFM
StackTopShiftUp	s(10) MT-TFM
StackTopDisplayStyleShiftUp	s(8) MT-TFM
StackBottomShiftDown	s(12) MT-TFM
StackBottomDisplayStyleShiftDown	s(11) MT-TFM
StackGapMin	3 * theta OT-SUG+MT
StackDisplayStyleGapMin	7 * theta OT-SUG+MT
SuperscriptShiftUp	s(13) 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(11) OT-MIG+MT
UpperLimitGapMin	e(9) OT-MIG+MT
LowerLimitGapMin	e(10) MT-TFM
LowerLimitBaselineDropMin	e(12) MT-TFM
StretchStackTopShiftUp	e(11) OT-MIG+MT
StretchStackGapAboveMin	e(9) 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

Field	Expression and source class
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
RadicalKernBeforeDegree	round(Fraction(5,18) * upm) OT-SUG
RadicalKernAfterDegree	-round(Fraction(10,18) * upm) OT-SUG
RadicalDegreeBottomRaisePercent	60 OT-SUG
ScriptPercentScaleDown	_pct(script_ratio) MT-STY
ScriptScriptPercentScaleDown	_pct(scriptscript_ratio) MT-STY
MinConnectorOverlap	100 OT-MIG+MT-PFB
AccentBaseHeight	xh MT-TFM
FlattenedAccentBaseHeight	round(cap_height * upm) MT-TFM
DelimitedSubFormulaMinHeight	int(display_operator_min_height) MT-MIG+MT
DisplayOperatorMinHeight	int(display_operator_min_height) MT-TFM+OT-MIG
MathLeading	150 OT-MIG
SkewedFractionHorizontalGap	350 OT-MIG
SkewedFractionVerticalGap	102 OT-MIG
SpaceAfterScript	round(Fraction(1,20) * upm) TeX
SubscriptBaselineDropMin	round(script_sy[19] * float(script_ratio) * upm) MT-TFM+OT-MIG
SuperscriptBaselineDropMax	round(script_sy[18] * float(script_ratio) * upm) MT-TFM+OT-MIG

D Roman donor coverage

The minimal donor contract contains 317 inputs: 255 from Regular and 62 from Bold, as defined in `tools/donor_scope.json`. Mathematical bold upright Latin and digits use MTPro2 glyphs in the final font; the Bold donor inputs remain part of the build's input requirements. The ranges below list the Regular selection; donor and output Unicode values are identical. Ranges include both endpoints.

Mathematical syntax can remap input characters. For a literal Lua math input, use `\Umathchar 0 \symoperators "00E9\relax`, replacing the Unicode value as needed. Upright Latin uses `\symup`; bold Latin and digits use `\symbfup`, which selects the MTPro2 designs.

U+0022--U+0027	U+00D1--U+00D6	U+0154--U+0165
U+002A	U+00D9--U+00DD	U+0168--U+017E
U+002D	U+00E0--U+00E5	U+01FA--U+01FF
U+0040--U+005A	U+00E7--U+00EF	U+0218--U+021B
U+005E--U+007A	U+00F1--U+00F6	U+0237
U+007E	U+00F9--U+00FD	U+02C6
U+00A3	U+00FF--U+010F	U+02DC
U+00A5	U+0112--U+0125	U+1E80--U+1E85
U+00AE	U+0128--U+0131	U+1EF2--U+1EF3
U+00B0	U+0134--U+0137	U+2044
U+00B7	U+0139--U+013E	U+20AC
U+00C0--U+00C5	U+0143--U+0148	
U+00C7--U+00CF	U+014C--U+0151	

D.1 Extended coverage and eligibility

Extended retains actually encoded Regular characters in U+0020–036F, U+1E00–1EFF, U+2000–218F, U+2500–266F, U+301A–301B and U+FB00–FB06, subject to MTPro2 ownership and edition restrictions. The following ranges are excluded: U+0370–03FF, U+0400–052F, U+2100–214F, U+2190–21FF, U+2200–22FF, U+2300–23FF, U+25A0–25FF, U+27C0–27EF, U+2980–2AFF and U+1D400–1D7FF. Exceptions U+2105, U+2116, U+211E, U+2120, U+2121, U+2122 and U+212E remain subject to the same ownership and edition restrictions. Primes U+2032–2037/U+2057, PUA U+E000–F8FF, U+FEFF/U+FFFD and existing output names are reserved. Imported glyphs must have usable outlines and metrics. The Bold donor requirements are unchanged by extended coverage.

The additional coverage depends on the donor pair.