mtp2otf v1.6

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 Windows. 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 `mtpro2` package's custom commands are not included. See `mtp2otf-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 `mtpro2/` and `donors/` beside `build.sh`. Supply the font files separately:

- `mtpro2/`: MTPro2 PFB/TFM files and `mtpro2.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. Ordinary mathematical characters include the source italic correction in their advances, so their spacing does not depend on Word adding it between letters. Ordinary pair kerning remains separate. Script placement and the width of a completed formula still depend on the equation engine; see Appendix C.

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 `mtp2otf-guide.pdf` are for LuaLaTeX.

Large operator selection depends on the equation's growth setting. In the Microsoft 365 Word examples checked on 25 October 2026, ordinary sums, products and integrals remained at display size even with tall operands or limits. Enabling n-ary growth selected the XXL designs for the three matrix examples. This does not establish access to every size or character variant in Word.

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 `mtp2otf-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.
v1.5 Added four original ready-made sizes to 27 large-operator series, including their existing character variants. Each series now contains six finite designs. Ordinary text/display selection is unchanged. Reorganized character variants and stylistic sets, separated independent symbol choices, and added feature labels and numbered-choice names.
v1.6 Included italic correction in ordinary mathematical advances and added compensating right-lower MATH kerns. Preserved source pair kerning, accent attachment and dedicated operator metrics.

A Stylistic sets and 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	F/L
	Bold forms	U+1D49B	F
cv02	Alternate Beta	U+03B2; U+1D6FD	F/L
	Bold forms	U+1D6C3; U+1D737	F
cv03	Alternate Delta	U+03B4; U+1D6FF	F/L
	Bold forms	U+1D6C5; U+1D739	F
cv04	Alternate Hbar	U+210F	F/L
cv05	Alternate Conjunction	U+2227; U+22C0	F/L
cv06	Straight Braces	U+007B; U+007D	F/L
cv07	Extended Parentheses	U+0028; U+0029	F/L
cv08	Extended Curly Braces	U+007B; U+007D	F/L
cv09	Horizontal Brace Forms: 1 = Plain 2 = Curly	U+23DE; U+23DF	F/L
cv10	Proper Inclusion Forms	U+228A; U+228B; U+2ACB; U+2ACC	F
cv11	Unequal Comparison Forms	U+2268; U+2269	F
cv12	Negated Comparison Forms: 1 = Slanted equality 2 = Double equality	U+2270; U+2271	F
cv13	Negated Inclusion Forms: 1 = Alternate inclusion 2 = Double equality	U+2288; U+2289	F
cv14	Not-Similar Form	U+2241	F
cv15	Proportional-To Form	U+221D	F
cv16	Smile and Frown Forms	U+2322; U+2323	F
cv17	Empty-Set Form	U+2205	F
cv18	Yen Form	U+00A5	F
cv19	Registered Form	U+00AE	F
ss01	Curly Script	Regular Script alphabet	F
ss02	Script Alternates	Regular and bold Script alternatives; existing Curly alternatives	F
ss03	Curly Script Alternates	Regular Script alphabet	F
ss04	Fraktur Alternates	Fraktur alternate inputs listed below	F
ss05	Heavy Blackboard	Upright Blackboard Latin and digits	F
ss06	Holey Blackboard	Upright Blackboard Latin and digits; U+213C--213F; U+2145--2149	F
ss07	Slanted Large Operators	U+220F--2211	F/L
ss08	Short Bar Relations	U+2223--2226	F
ss09	Thick Tilde Relations	U+223C; U+2248	F
ss10	Shaded Card Suits	U+2660; U+2663	F/L

ss05 and ss06 together select the heavier Holey upright Blackboard design. ss05 leaves the Greek and standard italic Blackboard characters unchanged. ss06 selects their corresponding Holey designs.

ss01 and ss02 can be combined. ss03 selects the Curly alternate series without requiring ss01; it overrides the regular Script choices of ss01 and ss02. ss04 remains independent.

For cv09, cv12 and cv13, choose one numbered value per tag. Value 0 disables that feature; it does not cancel another active feature. Negated comparisons (cv12) and inclusions (cv13) can use different choices. cv14 affects only U+2241.

ss03 alone and ss01,ss02 agree for regular Script. Only the latter also enables the existing bold Script alternatives. The Curly inputs G/M/N/Q/Y have existing alternatives under ss02.

Full provides 19 character variants and 10 stylistic sets. Lite provides cv01--cv09, ss07 and ss10, with the same meanings. All selectors are off by default. Numbering does not determine lookup order. New features will use unused numbers; published meanings will remain fixed.

Feature labels are stored in OpenType FeatureParams and the name table. Character variants also contain descriptions, samples and, where relevant, ordered parameter labels. Their descriptive Unicode lists are separate from the actual GSUB glyph coverage. Applications decide whether to expose these names; they are not additional TeX commands.

The substitution report records the public feature tags and lookup order. Glyph names are independent of public feature tags.

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 uses fixed U+0304; \overbar uses fixed U+0305, and \wideoverbar requests the stretching U+0305 accent.

Unicode	Input	Selection	V/H	Var.	Asm.
U+0028	\lparen	default	V	5	yes
U+0028	\lparen	cv07=1	V	24	yes
U+0029	\rparen	default	V	5	yes
U+0029	\rparen	cv07=1	V	24	yes
U+002F	\mathslash	default	V	5	no
U+005B	\lbrack	default	V	5	yes
U+005C	\backslash	default	V	5	no
U+005D	\rbrack	default	V	5	yes
U+007B	\lbrace	default	V	5	yes
U+007B	\lbrace	cv08=1	V	24	yes
U+007B	\lbrace	cv06=1	V	5	yes
U+007C	\vert	default	V	8	yes
U+007D	\rbrace	default	V	5	yes
U+007D	\rbrace	cv08=1	V	24	yes
U+007D	\rbrace	cv06=1	V	5	yes

Unicode	Input	Selection	V/H	Var.	Asm.
U+0302	\widehat {...}	default	H	37	no
U+0303	\widetilde {...}	default	H	37	no
U+0305	\wideoverbar {...}	default	H	3	yes
U+030C	\widecheck {...}	default	H	35	no
U+2016	\Vert	default	V	8	yes
U+20D6	\overleftarrow {...}	default	H	2	yes
U+20D7	\overrightarrow {...}	default	H	2	yes
U+2190	\leftarrow	default	H	2	yes
U+2191	\uparrow	default	V	1	yes
U+2192	\rightarrow	default	H	2	yes
U+2193	\downarrow	default	V	1	yes
U+2194	\leftrightarrow	default	H	2	yes
U+2195	\updownarrow	default	V	1	yes
U+21A6	\mapsto	default	H	2	no
U+21D0	\Leftarrow	default	H	2	yes
U+21D1	\Uparrow	default	V	1	yes
U+21D2	\Rightarrow	default	H	2	yes
U+21D3	\Downarrow	default	V	1	yes
U+21D4	\Leftrightarrow	default	H	2	yes
U+21D5	\Updownarrow	default	V	1	yes
U+220F	\prod	default	V	6	no
U+220F	\prod	ss07=1	V	6	no
U+2210	\coprod	default	V	6	no
U+2210	\coprod	ss07=1	V	6	no
U+2211	\sum	default	V	6	no
U+2211	\sum	ss07=1	V	6	no
U+221A	\sqrt {...}	default	V	5	yes
U+2223	\mid	default	V	8	yes
U+2225	\parallel	default	V	8	yes
U+222B	\int	default	V	6	no
U+222C	\iint	default	V	6	no
U+222D	\iiint	default	V	6	no
U+222E	\oint	default	V	6	no
U+222F	\oiint	default	V	6	no
U+2230	\oiiint	default	V	6	no
U+2231	\intclockwise	default	V	6	no
U+2232	\varointclockwise	default	V	6	no
U+2233	\ointctrclockwise	default	V	6	no
U+22C0	\bigwedge	default	V	6	no
U+22C0	\bigwedge	cv05=1	V	6	no
U+22C1	\bigvee	default	V	6	no
U+22C2	\bigcap	default	V	6	no
U+22C3	\bigcup	default	V	6	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

Unicode	Input	Selection	V/H	Var.	Asm.
U+23B1	\rmoustache	default	V	1	yes
U+23DC	\overparen {...}	default	H	24	yes
U+23DE	\overbrace {...}	default	H	0	yes
U+23DE	\overbrace {...}	cv09=1	H	0	yes
U+23DE	\overbrace {...}	cv09=2	H	35	yes
U+23DF	\underbrace {...}	default	H	0	yes
U+23DF	\underbrace {...}	cv09=1	H	0	yes
U+23DF	\underbrace {...}	cv09=2	H	35	yes
U+27E8	\langle	default	V	8	no
U+27E9	\rangle	default	V	8	no
U+27EE	\lgroup	default	V	1	yes
U+27EF	\rgroup	default	V	1	yes
U+2A00	\bigodot	default	V	6	no
U+2A01	\bigoplus	default	V	6	no
U+2A02	\bigotimes	default	V	6	no
U+2A03	\bigcupdot	default	V	2	no
U+2A04	\biguplus	default	V	6	no
U+2A06	\bigsqcup	default	V	6	no
U+2A0D	\intbar	default	V	6	no
U+2A0F	\fint	default	V	6	no

The engine compares its size request with MATH AdvanceMeasurement. This is a selection dimension and need not equal the glyph's ink extent. The 27 extended operator series have text/display and four additional ready-made designs (xl, XL, XXL, XXXL), with no repeatable assembly. Normal text/display selection and all MATH constants remain unchanged. With ordinary `unicode-math` input, the height of a following expression does not enlarge an operator. A larger design can be selected by an explicit LuaTeX size request; see `mtp2otf-guide.pdf`. Word controls these requests through its equation settings. Bigcupdot (U+2A03) retains its two designs. Default parentheses, braces and radicals use the ordinary source sequence; `cv07`/`cv08` select larger delimiter series.

B.1 Angle brackets and vertical bars

The small sizes of angle brackets and vertical bars need a common size sequence. In the original conversion, the angle brackets had five finite sizes, while single and double bars had only their base size followed by an assembly. Comparing automatic delimiters with Latin Modern Math showed that a bar could remain visibly shorter than the adjacent angle brackets around the same expression. Latin Modern provided a reference for closer small-size steps, rather than outlines to copy.

The source and output use different growth mechanisms. Traditional TeX uses the TFM next-larger chain and extensible recipes; part placement is based on TFM box dimensions. OpenType MATH supplies size measurements, part advances and connector lengths, and lets the layout engine vary connector overlap within those limits. Thus, translating the source outlines does not by itself reproduce the Type 1 assembly heights. For the source bar extension parts, the TFM height plus depth is about 600 font units, whereas the outline spans 636. The source's repeated placement therefore gives about 36 units of outline overlap. This is not the same quantity as MATH's minimum connector overlap, which is 20 in the final font. With only a few parts, a difference in placement or size selection is a larger fraction of the total height and is more noticeable than in a tall assembly.

To give small delimiters more consistent choices, v1.5 adds three intermediate angle sizes and seven finite sizes to each bar series. The resulting eight stages are shared by the left and right angle brackets, single bars and double bars in both editions:

Ink height	AdvanceMeasurement	Angle design
931	932	source
1100	1101	interpolated
1200	1201	source
1450	1451	interpolated
1800	1801	source
2100	2101	interpolated
2400	2401	source
3000	3001	source

Values are in font units at 1000 units per em. The new 1100, 1450 and 2100 stages follow the corresponding Latin Modern angle sizes, while retaining all five source angle designs. The bars follow this adopted angle sequence, not Latin Modern's separate bar sequence. AdvanceMeasurement is the ceiling of the ink height plus one unit; it is the selection measurement, not the horizontal advance width.

New angle outlines interpolate corresponding vertices and advance widths between adjacent source designs after aligning their vertical centers. This retains the source designs at their original sizes and avoids simply stretching one angle vertically. New bar outlines use the source extension part's horizontal coordinates and advance: each stroke is 45 units wide, with a 103-unit inner gap in the double bar. The original base bars are retained.

Each added bar matches both the lower and upper ink bounds of the corresponding angle, not just its total height. LuaTeX rounds height and depth separately in script and scriptscript styles; matching only their sum can select different stages at a size boundary. The relation inputs U+2223 and U+2225 share the finite size lists of U+007C and U+2016 respectively. The implementation is `build_angle_bar_sizes` in `tools/font_geometry.py`.

This addition leaves the existing bar assembly records intact. Bars can still grow beyond the largest finite size, whereas angle brackets have no assembly and stop at the 3000-unit design. The shared sequence improves small-size selection; it does not guarantee equal heights for every expression, application or size request.

C Metrics and MATH constants

C.1 Glyph metrics

The output uses 1000 units per em. For ordinary mathematical characters, let W and I be the separately rounded integer source width and italic correction before the final advance conversion. The final values are

$$\text{advance} = W + I, \qquad \text{MATH IC} = 0, \qquad \text{BottomRight MathKern} = -I.$$

The right-lower kern is constant at every height; the other three corners are unset. Glyphs with zero correction retain their metrics without a redundant zero-kern record. The sum uses the existing integers, rather than rounding the raw TFM sum again.

This policy covers the mathematical alphabets, including Script, Fraktur and Blackboard, their optical sizes and alternate designs. It also covers ordinary nabla, bold nabla and partial differential, and the AMS yen alternative where the edition supplies them. Each physical glyph uses its own source correction. The correction is not inferred from whether the outline looks italic or from its Unicode category alone.

The source TeX spacing for an ordinary character includes its italic correction. Storing that effective width as the advance makes it available to engines that do not add the correction between ordinary letters. The superscript reference $W + I$ is retained; the subscript reference returns to W when the engine applies the right-lower kern. Dedicated integrals, sums and products retain their separate metrics and italic corrections, including the corrections used to position limits. Their existing conversion may already include source italic correction in the advance. Delimiters, assemblies, accents and primes retain their own construction and placement rules.

Ordinary GPOS pair adjustments store the TFM pair kern without an added italic-correction term. Skewchar pairs are excluded from ordinary kerning. Top-accent attachment for source math alphabets is based on

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

For untransformed source alphabets, this calculation uses the exact TFM fixed-point values and is rounded once to output units. Its absolute coordinate is retained when the advance is changed; it is not recomputed from the new advance or zero MATH IC. Combining accents and synthesized forms use the corresponding coordinate transformations and geometric attachment positions. Primes retain their dedicated conversion. Text, script and scriptscript designs use the source optical sizes.

Engine-dependent limits. LuaHBTeX 1.24 can omit MathKern when a subscript starts with a fraction, radical, explicit box or unsupported structure, or uses a font without MATH support. An explicit style at the start or an empty subscript can also change the path used by the engine. Such a subscript may move right by the original I; a single visible letter does not guarantee that the kern is used. Converting an ordinary character into an operator with `\mathop` also cannot recover its former limit correction. These cases do not change the dedicated operator data or call for an additional pair-kerning adjustment.

Preserving the subscript origin does not always preserve the completed box width. An engine that includes the base advance in the maximum of component widths changes that term from W to $W + I$. For a subscript of width S and trailing script space s, the width is preserved when $S + s \geq I$. A narrow subscript on a large-correction glyph can therefore increase the box width; this is not restricted to Script alphabets. For example, a 10 pt Script T with a dot subscript gives a 0.365 pt increase under this maximum-width rule, although the subscript origin is unchanged. This is a reference calculation, not a Word measurement. Similarly, retaining accent attachment coordinates does not guarantee that an engine selects the same width of stretching accent.

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 `mt2syt.tfm`; `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
AxisHeight	s(22) MT-TFM
FractionNumeratorShiftUp	s(9) MT-TFM
FractionNumeratorDisplayStyle ShiftUp	s(8) MT-TFM
FractionDenominatorShiftDown	s(12) MT-TFM
FractionDenominatorDisplayStyle ShiftDown	s(11) MT-TFM
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
SuperscriptBottomMaxWith Subscript	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

Field	Expression and source class
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
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	20 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

Field	Expression and source class
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.