

mtp2otf

User guide and font 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 large symbols, 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; Section B.6 relates them to the available inputs.

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 Word, ordinary character spacing can be tighter than in the original Type 1 package.

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 and private-use examples in this manual 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. OpenType feature selection is described in Section 4.

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

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

3.3 Samples

`doc/word.pdf` and `doc/lualatex.pdf` show matching Full-edition examples. Their editable sources are `doc/word.docx` and `doc/lualatex.tex`.

Open the Word source in Word and export its PDF there. To typeset the LuaLaTeX source, build Full and run `lualatex lualatex.tex` from `doc/`. The source uses `mathtools`, `unicode-math`, and Nimbus Roman Regular, Bold, Italic and BoldItalic.

4 Alphabets and symbol selection

4.1 Mathematical alphabets

Use the mathematical Unicode characters or the corresponding `unicode-math` commands. The following commands select the principal Latin alphabets:

Alphabet	Command	Edition
Upright	<code>\symup {ABCxyz}</code>	Full, Lite
Italic	<code>\symit {ABCxyz}</code>	Full, Lite
Bold upright	<code>\symbfup {ABCxyz}</code>	Full, Lite
Bold italic	<code>\symbfir {ABCxyz}</code>	Full
Script	<code>\symscr {ABCxyz}</code>	Full
Bold Script	<code>\symbfscr {ABCxyz}</code>	Full
Fraktur	<code>\symfrak {ABCxyz}</code>	Full
Bold Fraktur	<code>\symbffrak {ABCxyz}</code>	Full
Blackboard bold	<code>\symbb {ABCxyz}</code>	Full

Mathematical sans and monospace alphabets are not included. Blackboard bold uses the original mtpbb design by default.

Roman donors supply upright Latin and mathematical bold upright Latin and digits. Mathematical italic, bold italic and non-Latin-shaped Greek come from MTPro2. Character variants provide alternative designs, including original MTPro2 bold and other Blackboard styles.

4.2 Alternative designs

Character variants (CVs) select alternatives to the default glyphs. Appendix A lists their tags and input characters. In LuaLaTeX, select a feature through `RawFeature`:

```
\setmathfont{MTPro2Math.otf}[RawFeature={+cv01}]
```

This selects swash z wherever its input character occurs. For a local choice, define a named math version in the preamble and select it outside the formula:

```
\setmathfont{MTPro2Math.otf}
\setmathfont{MTPro2Math.otf}[version=swash,
                             RawFeature={+cv01}]
\begingroup\mathversion{swash}
\[ z \]
\endgroup
```

A group restores the previous version. Numbered `RawFeature` choices start at 1; for example, `cv15=27` selects a particular arc design. Enable only one of the competing variants for a given input.

The `ssty` feature supplies optical script designs to the math engine. In Full, `salt` selects Curly alternatives from Script inputs: add `RawFeature={+salt}` to `\setmathfont` and use `\symscr {ABC}`. A named math version limits the selection to a group.

4.3 Growing and fixed symbols

Delimiters, radicals and wide accents select sizes through OpenType MATH. For example:

```
\[ \left(\frac{a+b}{c+d}\right)
\quad \sqrt{x^2+y^2}
\quad \widehat{ABC} \]
```

The font supplies prebuilt designs and, for some symbols, assemblies that extend them with repeated parts. The engine selects a size from the formula's dimensions and math style. Large operators use a finite series of prebuilt designs.

Fixed CV choices select individual designs for custom constructions. Their measurement, centering and spacing are the responsibility of the calling macro. Accent glyphs require accent syntax to attach them to a nucleus. Appendix B gives the automatic inputs, fixed choices and private-use entries.

5 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>

A Character variants

The table lists CV tags, their standard Unicode inputs and edition availability. Tags with multiple choices use the numbered values in Appendix B. Selection syntax is given in Section 4. AMS denotes MTPro2's own AMS symbol sources.

Feature	Meaning and standard Unicode targets	Edition
cv01	swash z U+1D467	both
cv02	Full empty-set alternate U+2205	Full
cv03	mtpbbd (mt2bb / b / n) Blackboard inputs	Full
cv04	mtpbbi (mt2bb / m / it) Blackboard inputs	Full
cv05	mtphrb (Holey Roman Bold family mt2hrb / m / n) Blackboard inputs	Full
cv06	mtphrd (Holey Roman Bold family mt2hrb / b / n) Blackboard inputs	Full
cv07	mtphbi (Holey Roman Bold family mt2hrb / m / it) Blackboard inputs	Full
cv08	straight braces U+007B, U+007D	both
cv09	MTPro2 AMS negated-relation alternates U+2241, U+2288–2289	Full
cv10	variant subsetneq/supsetneq U+228A–228B	Full
cv11	hbar alternate U+210F	both
cv12	slanted large operators U+220F–2211	both
cv13	fixed overbrace designs U+23DE	both
cv14	fixed underbrace designs U+23DF	both
cv15	fixed overparen designs U+23DC	both
cv16	fixed delimiter designs U+0028–0029, U+005C, U+007B, U+007D, U+27E8–27E9	both
cv17	fixed radical designs U+221A	both
cv18	fixed accent designs U+0302–0303, U+030C	both
cv19	fixed brace designs U+007B, U+007D	both
cv20	plain stretchable brace construction U+23DE–23DF	both
cv21	original curly brace variant/assembly construction U+23DE–23DF	both
cv89	original MTPro2 bold Latin and digits U+1D400–1D433, U+1D7CE–1D7D7	both
cv89	Latin-shaped Greek aliases of those bold glyphs U+1D6A8–1D6A9, U+1D6AC–1D6AE, U+1D6B0–1D6B1, U+1D6B3–1D6B4, U+1D6B6, U+1D6B8, U+1D6BB, U+1D6BE, U+1D6D0	Full
cv90	Script/Fraktur source alternates and Fraktur reuse, text design Script/Fraktur inputs	Full
cv91	Script/Fraktur source alternates and Fraktur reuse, script design Script/Fraktur inputs	Full

Feature	Meaning and standard Unicode targets	Edition
cv92	Script/Fraktur source alternates and Fraktur reuse, scriptscript design Script/Fraktur inputs	Full
cv93	Curly source alternates, text design U+2133, U+1D4A2, U+1D4A9, U+1D4AC, U+1D4B4	Full
cv94	Curly source alternates, script design U+2133, U+1D4A2, U+1D4A9, U+1D4AC, U+1D4B4	Full
cv95	Curly source alternates, scriptscript design U+2133, U+1D4A2, U+1D4A9, U+1D4AC, U+1D4B4	Full
cv96	original MTPPro2 AMS yen/registered alternates U+00A5, U+00AE	Full

Blackboard inputs: U+2102, U+210D, U+2115, U+2119–211A, U+211D, U+2124, U+1D538–1D539, U+1D53B–1D53E, U+1D540–1D544, U+1D546, U+1D54A–1D550, U+1D552–1D56B, U+1D7D8–1D7E1.

Script/Fraktur inputs: U+2110–2112, U+211C, U+1D49E, U+1D4A2, U+1D4AC, U+1D4AE, U+1D4B4–1D4B5, U+1D4C7, U+1D4CF, U+1D4D2, U+1D4D6, U+1D4D8, U+1D4DB, U+1D4E0, U+1D4E2, U+1D4E8–1D4E9, U+1D4FB, U+1D503, U+1D51C, U+1D535–1D536, U+1D584, U+1D59D–1D59E.

A.1 Additional source letters and fixed accents

The following private-use characters select source letter and fixed-accent designs in both editions. Letter inputs use an explicit math character; fixed accents use an accent primitive:

```
\[ \Umathchar 0 \symoperators "E240\relax \]
\[ \Umathaccent fixed 0 \symoperators "E24C {x} \]
```

Input	Design
U+E240	Italic beta variant
U+E241, U+E242	Italic and upright delta variants
U+E243, U+E244	Italic and upright barred d
U+E24E, U+E24A	Fixed bar designs: wbar and wwbar
U+E24F, U+E250, U+E251	Fixed wide hat, tilde and check
U+E24B, U+E24C	Raised dot and double-dot accents
U+E254, U+E255	Raised triple-dot and quadruple-dot accents

B Large-symbol inputs

B.1 Automatic size selection

The table lists the input, direction of growth (V: vertical; H: horizontal), number of prebuilt variants, and availability of assembly. The input examples use LuaLaTeX. Pair `\left` and `\right` around a formula; a dot gives an invisible delimiter. Use `\sqrt[n]` for a radical with a degree.

Meaning	Unicode	Input	V/H	Var.	Asm.	Control	Edition
left parenthesis	U+0028	<code>\left (</code>	V	26	yes	body height	both
right parenthesis	U+0029	<code>\right)</code>	V	26	yes	body height	both

Meaning	Unicode	Input	V/H	Var.	Asm.	Control	Edition
solidus	U+002F	<code>\left/</code>	V	26	no	body height	both
left square bracket	U+005B	<code>\left\lbrack</code>	V	5	yes	body height	both
reverse solidus	U+005C	<code>\left\backslash</code>	V	26	no	body height	both
right square bracket	U+005D	<code>\right\rbrack</code>	V	5	yes	body height	both
left curly bracket	U+007B	<code>\left\{</code>	V	26	yes	body height	both
vertical line	U+007C	<code>\left\vert</code>	V	2	yes	body height	both
right curly bracket	U+007D	<code>\right\}</code>	V	26	yes	body height	both
combining circumflex accent	U+0302	<code>\widehat</code>	H	40	no	body width	both
combining tilde	U+0303	<code>\widetilde</code>	H	40	no	body width	both
combining caron	U+030C	<code>\widecheck</code>	H	40	no	body width	both
double vertical line	U+2016	<code>\left\Vert</code>	V	2	yes	body height	both
n-ary product	U+220F	<code>\prod</code>	V	6	no	operator size	both
n-ary coproduct	U+2210	<code>\coprod</code>	V	6	no	operator size	both
n-ary summation	U+2211	<code>\sum</code>	V	6	no	operator size	both
square root	U+221A	<code>\sqrt</code>	V	26	yes	body height	both
integral	U+222B	<code>\int</code>	V	6	no	operator size	both
double integral	U+222C	<code>\iint</code>	V	6	no	operator size	both
triple integral	U+222D	<code>\iiint</code>	V	6	no	operator size	both
contour integral	U+222E	<code>\oint</code>	V	6	no	operator size	both
surface integral	U+222F	<code>\oiint</code>	V	6	no	operator size	both
volume integral	U+2230	<code>\oiint</code>	V	6	no	operator size	both
clockwise integral	U+2231	<code>\intclockwise</code>	V	6	no	operator size	both
clockwise contour integral	U+2232	<code>\varointclockwise</code>	V	6	no	operator size	both
anticlockwise contour integral	U+2233	<code>\ointctrclockwise</code>	V	6	no	operator size	both
n-ary logical and	U+22C0	<code>\bigwedge</code>	V	6	no	operator size	both
n-ary logical or	U+22C1	<code>\bigvee</code>	V	6	no	operator size	both
n-ary intersection	U+22C2	<code>\bigcap</code>	V	6	no	operator size	both
n-ary union	U+22C3	<code>\bigcup</code>	V	6	no	operator size	both
left ceiling	U+2308	<code>\left\lceil</code>	V	5	yes	body height	both
right ceiling	U+2309	<code>\right\rceil</code>	V	5	yes	body height	both
left floor	U+230A	<code>\left\lfloor</code>	V	5	yes	body height	both
right floor	U+230B	<code>\right\rfloor</code>	V	5	yes	body height	both
top parenthesis	U+23DC	<code>\overparen</code>	H	26	yes	body width	both
top curly bracket	U+23DE	<code>\overbrace</code>	H	0	yes	body width	both
bottom curly bracket	U+23DF	<code>\underbrace</code>	H	0	yes	body width	both
mathematical left angle bracket	U+27E8	<code>\left\langle</code>	V	26	no	body height	both
mathematical right angle bracket	U+27E9	<code>\right\rangle</code>	V	26	no	body height	both
n-ary circled dot operator	U+2A00	<code>\bigodot</code>	V	6	no	operator size	both
n-ary circled plus operator	U+2A01	<code>\bigoplus</code>	V	6	no	operator size	both
n-ary circled times operator	U+2A02	<code>\bigotimes</code>	V	6	no	operator size	both
n-ary union operator with dot	U+2A03	<code>\bigcupdot</code>	V	6	no	operator size	both
n-ary union operator with plus	U+2A04	<code>\biguplus</code>	V	6	no	operator size	both
n-ary square union operator	U+2A06	<code>\bigsqcup</code>	V	6	no	operator size	both
finite part integral	U+2A0D	<code>\intbar</code>	V	6	no	operator size	both
integral average with slash	U+2A0F	<code>\fint</code>	V	6	no	operator size	both
cap product	U+E280	<code>\Umathchar 1</code> <code>\symoperators "E280</code>	V	6	no	operator size	both
slanted sum	U+E281	<code>\Umathchar 1</code> <code>\symoperators "E281</code>	V	6	no	operator size	both
slanted product	U+E282	<code>\Umathchar 1</code> <code>\symoperators "E282</code>	V	6	no	operator size	both

Meaning	Unicode	Input	V/H	Var.	Asm.	Control	Edition
slanted coproduct	U+E283	<code>\Umathchar 1</code> <code>\symoperators "E283</code>	V	6	no	operator size	both
variant large land	U+E284	<code>\Umathchar 1</code> <code>\symoperators "E284</code>	V	6	no	operator size	both
large asterisk	U+E285	<code>\Umathchar 1</code> <code>\symoperators "E285</code>	V	6	no	operator size	both
finite wide overbar	U+E286	accent input; see B.5	H	3	no	body width	both
assembled wide vector	U+E287	accent input; see B.5	H	0	yes	body width	both

B.2 Size requests

The engine selects variants using `MATH AdvanceMeasurement`, formula dimensions, style, delimiter factor and shortfall. `AdvanceMeasurement` is a selection dimension, not an ink measurement. Operators have finite size series. Lua can request a larger operator with `\Umathoperatorsize` :

```
\begingroup
\[ \Umathoperatorsize\displaystyle=72pt
  \sum_{n=0}^N a_n \]
\endgroup
```

The request selects an available operator design; its ink height may differ from 72pt.

B.3 Fixed character-variant choices

The table lists 121 choices in their parameter order. `cv15=27` selects the alternate arc design. `cv13–cv16` and `cv18–cv19` select glyphs without growing constructions. Each `cv17` choice selects a single terminal radical design without assembly. Use the `Input` and `RawFeature` columns to select a design; the last column names the stored glyph.

For a fixed design, select the feature and use an explicit math class. This example displays the alternate arc as a standalone class-0 symbol:

```
\setmathfont{MTPro2Math.otf}[version=fixedarc,
                             RawFeature={cv15=27}]
\begingroup\mathversion{fixedarc}
\[ \Umathchar 0 \symoperators "23DC \]
\endgroup
```

Fixed accents require accent syntax; use the ordinary radical for automatic `\sqrt` growth.

Input	RawFeature	Glyph identifier
U+23DE	cv13=1	uni23DE.curly.exe176.fixed
U+23DE	cv13=2	uni23DE.curly.exe177.fixed
U+23DE	cv13=3	uni23DE.curly.exe178.fixed
U+23DE	cv13=4	uni23DE.curly.exe179.fixed
U+23DE	cv13=5	uni23DE.curly.exe180.fixed
U+23DE	cv13=6	uni23DE.curly.exe181.fixed
U+23DE	cv13=7	uni23DE.curly.exe182.fixed
U+23DE	cv13=8	uni23DE.curly.exe183.fixed
U+23DE	cv13=9	uni23DE.curly.exe184.fixed
U+23DE	cv13=10	uni23DE.curly.exe185.fixed
U+23DE	cv13=11	uni23DE.curly.exe186.fixed

Input	RawFeature	Glyph identifier
U+23DE	cv13=12	uni23DE.curly.exe187.fixed
U+23DE	cv13=13	uni23DE.curly.exe188.fixed
U+23DE	cv13=14	uni23DE.curly.exe189.fixed
U+23DE	cv13=15	uni23DE.curly.exe190.fixed
U+23DE	cv13=16	uni23DE.curly.exe191.fixed
U+23DE	cv13=17	uni23DE.curly.exe192.fixed
U+23DE	cv13=18	uni23DE.curly.exf160.fixed
U+23DE	cv13=19	uni23DE.curly.exf161.fixed
U+23DE	cv13=20	uni23DE.curly.exf162.fixed
U+23DE	cv13=21	uni23DE.curly.exf163.fixed
U+23DE	cv13=22	uni23DE.curly.exf164.fixed
U+23DE	cv13=23	uni23DE.curly.exf165.fixed
U+23DE	cv13=24	uni23DE.curly.exf166.fixed
U+23DE	cv13=25	uni23DE.curly.exf167.fixed
U+23DE	cv13=26	uni23DE.curly.exf168.fixed
U+23DE	cv13=27	uni23DE.curly.exf169.fixed
U+23DE	cv13=28	uni23DE.curly.exf170.fixed
U+23DE	cv13=29	uni23DE.curly.exf171.fixed
U+23DE	cv13=30	uni23DE.curly.exg160.fixed
U+23DE	cv13=31	uni23DE.curly.exg161.fixed
U+23DE	cv13=32	uni23DE.curly.exg162.fixed
U+23DE	cv13=33	uni23DE.curly.exg163.fixed
U+23DE	cv13=34	uni23DE.curly.exg164.fixed
U+23DE	cv13=35	uni23DE.curly.exg165.fixed
U+23DF	cv14=1	uni23DF.curly.exe144.fixed
U+23DF	cv14=2	uni23DF.curly.exe145.fixed
U+23DF	cv14=3	uni23DF.curly.exe146.fixed
U+23DF	cv14=4	uni23DF.curly.exe147.fixed
U+23DF	cv14=5	uni23DF.curly.exe148.fixed
U+23DF	cv14=6	uni23DF.curly.exe149.fixed
U+23DF	cv14=7	uni23DF.curly.exe150.fixed
U+23DF	cv14=8	uni23DF.curly.exe151.fixed
U+23DF	cv14=9	uni23DF.curly.exe152.fixed
U+23DF	cv14=10	uni23DF.curly.exe153.fixed
U+23DF	cv14=11	uni23DF.curly.exe154.fixed
U+23DF	cv14=12	uni23DF.curly.exe155.fixed
U+23DF	cv14=13	uni23DF.curly.exe156.fixed
U+23DF	cv14=14	uni23DF.curly.exe157.fixed
U+23DF	cv14=15	uni23DF.curly.exe158.fixed
U+23DF	cv14=16	uni23DF.curly.exe159.fixed
U+23DF	cv14=17	uni23DF.curly.exe160.fixed
U+23DF	cv14=18	uni23DF.curly.exf144.fixed
U+23DF	cv14=19	uni23DF.curly.exf145.fixed
U+23DF	cv14=20	uni23DF.curly.exf146.fixed
U+23DF	cv14=21	uni23DF.curly.exf147.fixed
U+23DF	cv14=22	uni23DF.curly.exf148.fixed
U+23DF	cv14=23	uni23DF.curly.exf149.fixed
U+23DF	cv14=24	uni23DF.curly.exf150.fixed
U+23DF	cv14=25	uni23DF.curly.exf151.fixed
U+23DF	cv14=26	uni23DF.curly.exf152.fixed
U+23DF	cv14=27	uni23DF.curly.exf153.fixed
U+23DF	cv14=28	uni23DF.curly.exf154.fixed

Input	RawFeature	Glyph identifier
U+23DF	cv14=29	uni23DF.curly.exf155.fixed
U+23DF	cv14=30	uni23DF.curly.exg144.fixed
U+23DF	cv14=31	uni23DF.curly.exg145.fixed
U+23DF	cv14=32	uni23DF.curly.exg146.fixed
U+23DF	cv14=33	uni23DF.curly.exg147.fixed
U+23DF	cv14=34	uni23DF.curly.exg148.fixed
U+23DF	cv14=35	uni23DF.curly.exg149.fixed
U+23DC	cv15=1	uni23DC.fixed
U+23DC	cv15=2	exa.191.fixed
U+23DC	cv15=3	exa.192.fixed
U+23DC	cv15=4	exa.193.fixed
U+23DC	cv15=5	exa.194.fixed
U+23DC	cv15=6	uni23DC.exe208.fixed
U+23DC	cv15=7	uni23DC.exe209.fixed
U+23DC	cv15=8	uni23DC.exe210.fixed
U+23DC	cv15=9	uni23DC.exe211.fixed
U+23DC	cv15=10	uni23DC.exe212.fixed
U+23DC	cv15=11	uni23DC.exe213.fixed
U+23DC	cv15=12	uni23DC.exe214.fixed
U+23DC	cv15=13	uni23DC.exf177.fixed
U+23DC	cv15=14	uni23DC.exf178.fixed
U+23DC	cv15=15	uni23DC.exf179.fixed
U+23DC	cv15=16	uni23DC.exf180.fixed
U+23DC	cv15=17	uni23DC.exf181.fixed
U+23DC	cv15=18	uni23DC.exf182.fixed
U+23DC	cv15=19	uni23DC.exf183.fixed
U+23DC	cv15=20	uni23DC.exg177.fixed
U+23DC	cv15=21	uni23DC.exg178.fixed
U+23DC	cv15=22	uni23DC.exg179.fixed
U+23DC	cv15=23	uni23DC.exg180.fixed
U+23DC	cv15=24	uni23DC.exg181.fixed
U+23DC	cv15=25	uni23DC.exg182.fixed
U+23DC	cv15=26	uni23DC.exg183.fixed
U+23DC	cv15=27	exa.arc195.fixed
U+0028	cv16=1	uni0028.exg0.fixed
U+0029	cv16=1	uni0029.exg1.fixed
U+007B	cv16=1	uni007B.exf8.fixed
U+007B	cv16=2	uni007B.exg8.fixed
U+007D	cv16=1	uni007D.exf9.fixed
U+007D	cv16=2	uni007D.exg9.fixed
U+27E8	cv16=1	uni27E8.exg10.fixed
U+27E9	cv16=1	uni27E9.exg11.fixed
U+005C	cv16=1	uni005C.exg15.fixed
U+221A	cv17=1	uni221A.exf112.fixed
U+221A	cv17=2	uni221A.exg112.fixed
U+0302	cv18=1	uni0302.exe57.fixed
U+0302	cv18=2	uni0302.exf59.fixed
U+0303	cv18=1	exa.103.fixed
U+030C	cv18=1	uni030C.exe89.fixed
U+030C	cv18=2	uni030C.exf91.fixed
U+007B	cv19=1	exa.morphed182.fixed
U+007B	cv19=2	exa.morphed184.fixed

Input	RawFeature	Glyph identifier
U+007B	cv19=3	exa.morphed186.fixed
U+007B	cv19=4	exa.morphed188.fixed
U+007D	cv19=1	exa.morphed183.fixed
U+007D	cv19=2	exa.morphed185.fixed
U+007D	cv19=3	exa.morphed187.fixed
U+007D	cv19=4	exa.morphed189.fixed

B.4 Alternative growing constructions

cv08 selects straight vertical braces; cv12 selects slanted large operators. For horizontal braces, cv20 selects plain assemblies and cv21 selects Curly variants with assembly.

Input	RawFeature	Construction root	Variants	Assembly
U+007B	+cv08	braceleft.straight	5	yes
U+007D	+cv08	braceright.straight	5	yes
U+2211	+cv12	slsum	6	no
U+220F	+cv12	slprod	6	no
U+2210	+cv12	slcoprod	6	no
U+23DE	+cv20	brace.plain.over	0	yes
U+23DF	+cv20	brace.plain.under	0	yes
U+23DE	+cv21	uni23DE.curly.exe176	35	yes
U+23DF	+cv21	uni23DF.curly.exe144	35	yes

For example, select the curly construction in the preamble and use ordinary overbrace and underbrace commands:

```
\setmathfont{MTPro2Math.otf}[RawFeature={+cv21}]

\[ \overbrace{a+b+c+d}^{\scriptstyle n} \quad \underbrace{a+b+c+d}_{\scriptstyle n} \]
```

B.5 Private-use operators and accents

These private-use entries provide explicit access to large operators and extensible accents. U+E280–E285 have class 1 and six vertical variants; U+E286 and U+E287 use accent primitives. For the operators, choose `\limits` or `\nolimits` according to the intended macro.

```
% A cap-product operator, with an explicit size request:
\[ \Umathoperators\displaystyle=72pt
    \Umathchar 1 \symoperators "E280\limits_{i=1}^{\scriptstyle n}a_i \]
```

Input	Meaning	Class / structure	Concrete input
U+E280	cap product	1 (operator)	<code>\Umathchar 1 \symoperators "E280</code>
U+E281	slanted sum	1 (operator)	<code>\Umathchar 1 \symoperators "E281</code>
U+E282	slanted product	1 (operator)	<code>\Umathchar 1 \symoperators "E282</code>
U+E283	slanted coproduct	1 (operator)	<code>\Umathchar 1 \symoperators "E283</code>
U+E284	variant large land	1 (operator)	<code>\Umathchar 1 \symoperators "E284</code>
U+E285	large asterisk	1 (operator)	<code>\Umathchar 1 \symoperators "E285</code>
U+E286	finite wide overbar	accent primitive	<code>\Umathaccent 0 \symoperators "E286 {a+b+c}</code>
U+E287	assembled wide vector	accent primitive	<code>\Umathaccent 0 \symoperators "E287 {a+b+c}</code>

B.6 Correspondence with Type 1 commands

The font provides glyphs and MATH data; the original package’s custom measurements and placement remain macro responsibilities. The following table gives the corresponding OpenType inputs.

Original name	OpenType input	Usage
<code>\PARENS / \LEFTRIGHT</code>	<code>\left / \right</code>	Choose delimiter inputs; the engine measures body height and centers using the math axis. Preserve custom side-space policy in a macro.
<code>\SQRT / \ROOT ... \OF</code>	<code>\sqrt / \sqrt[n]</code>	MATH handles radical growth and ordinary degree placement; original manual offsets remain a macro policy.
<code>\XL / \XXL / \XXXL</code>	<code>\Umathoperatorsize</code>	Request size for standard/PUA or cv12 operator families; source names are not exact size presets. Class and limits are explicit.
<code>\overbrace / \underbrace</code>	<code>cv21 / cv13 / cv14</code>	cv21 grows over/under a nucleus. cv13/cv14 select fixed designs whose placement and width policy are supplied by the macro.
<code>\arc / \Arc / \widearc</code>	<code>\overparen / cv15</code>	Automatic width selection or fixed parameter. cv15=27 preserves the alternate arc design.
<code>\widehat / \widecheck / \widetilde</code>	<code>automatic / cv18</code>	Automatic width selection or fixed choices; accent syntax supplies nucleus attachment.
<code>\straightbraces / \morphedbraces</code>	<code>cv08 / cv19</code>	cv08 selects growing straight braces; cv19 selects fixed morphed designs.

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

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

Coordinate transformations are accounted for in combining accents and synthesized forms. 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/math_constants.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 `mt2synt.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 the floor of the average of the separately rounded height-plus-depth values of extension TFM slots 0x50 and 0x58, in output units. The delimited-subformula threshold uses the same value.

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
FractionNumeratorDisplayStyleShiftUp	s (8) MT-TFM
FractionDenominatorShiftDown	s (12) MT-TFM
FractionDenominatorDisplayStyleShiftDown	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 (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

Field	Expression and source class
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
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	477 OT-MIG
FlattenedAccentBaseHeight	656 OT-MIG
DelimitedSubFormulaMinHeight	int(display_operator_min_height) MT-MIG+MT
DisplayOperatorMinHeight	int(display_operator_min_height) MT-TFM
MathLeading	154 OT-MIG

Field	Expression and source class
SkewedFractionHorizontalGap	350 OT-MIG
SkewedFractionVerticalGap	102 OT-MIG
SpaceAfterScript	$\text{round}(\text{Fraction}(1,20) * \text{upm})$ TeX
SubscriptBaselineDropMin	$\text{round}(\text{script_sy}[19] * \text{float}(\text{script_ratio}) * \text{upm})$ MT-TFM+OT-MIG
SuperscriptBaselineDropMax	$\text{round}(\text{script_sy}[18] * \text{float}(\text{script_ratio}) * \text{upm})$ MT-TFM+OT-MIG

D Roman donor coverage

Minimal coverage selects 317 encoded inputs: 255 from Regular and 62 from Bold. The table gives output and donor Unicode ranges. Each range includes both endpoints. The selection is defined in `tools/donor_scope.json`.

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`.

Output Unicode	Donor face	Donor Unicode
U+0022--U+0027	Regular	U+0022--U+0027
U+002A	Regular	U+002A
U+002D	Regular	U+002D
U+0040--U+005A	Regular	U+0040--U+005A
U+005E--U+007A	Regular	U+005E--U+007A
U+007E	Regular	U+007E
U+00A3	Regular	U+00A3
U+00A5	Regular	U+00A5
U+00AE	Regular	U+00AE
U+00B0	Regular	U+00B0
U+00B7	Regular	U+00B7
U+00C0--U+00C5	Regular	U+00C0--U+00C5
U+00C7--U+00CF	Regular	U+00C7--U+00CF
U+00D1--U+00D6	Regular	U+00D1--U+00D6
U+00D9--U+00DD	Regular	U+00D9--U+00DD
U+00E0--U+00E5	Regular	U+00E0--U+00E5
U+00E7--U+00EF	Regular	U+00E7--U+00EF
U+00F1--U+00F6	Regular	U+00F1--U+00F6
U+00F9--U+00FD	Regular	U+00F9--U+00FD
U+00FF--U+010F	Regular	U+00FF--U+010F
U+0112--U+0125	Regular	U+0112--U+0125
U+0128--U+0131	Regular	U+0128--U+0131
U+0134--U+0137	Regular	U+0134--U+0137
U+0139--U+013E	Regular	U+0139--U+013E
U+0143--U+0148	Regular	U+0143--U+0148

Output Unicode	Donor face	Donor Unicode
U+014C--U+0151	Regular	U+014C--U+0151
U+0154--U+0165	Regular	U+0154--U+0165
U+0168--U+017E	Regular	U+0168--U+017E
U+01FA--U+01FF	Regular	U+01FA--U+01FF
U+0218--U+021B	Regular	U+0218--U+021B
U+0237	Regular	U+0237
U+02C6	Regular	U+02C6
U+02DC	Regular	U+02DC
U+1E80--U+1E85	Regular	U+1E80--U+1E85
U+1EF2--U+1EF3	Regular	U+1EF2--U+1EF3
U+2044	Regular	U+2044
U+20AC	Regular	U+20AC
U+1D400--U+1D419	Bold	U+0041--U+005A
U+1D41A--U+1D433	Bold	U+0061--U+007A
U+1D7CE--U+1D7D7	Bold	U+0030--U+0039

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. Bold coverage remains unchanged.

The additional coverage depends on the donor pair.