

# Experimental Unicode mathematical typesetting: The unicode-math package

Will Robertson, Philipp Stephani and Khaled Hosny  
will.robertson@latex-project.org

2017/10/02      v0.8g

## Abstract

This document describes the unicode-math package, which is intended as an implementation of Unicode maths for  $\LaTeX$  using the  $X_{\text{E}}\text{TeX}$  and  $\text{LuaTeX}$  typesetting engines. With this package, changing maths fonts is as easy as changing text fonts — and there are more and more maths fonts appearing now. Maths input can also be simplified with Unicode since literal glyphs may be entered instead of control sequences in your document source.

The package provides support for both  $X_{\text{E}}\text{TeX}$  and  $\text{LuaTeX}$ . The different engines provide differing levels of support for Unicode maths. Please let us know of any troubles.

Alongside this documentation file, you should be able to find a minimal example demonstrating the use of the package, ‘unimath-example.ltx’. It also comes with a separate document, ‘unimath-symbols.pdf’, containing a complete listing of mathematical symbols defined by unicode-math, including comparisons between different fonts.

Finally, while the STIX fonts may be used with this package, accessing their alphabets in their ‘private user area’ is not yet supported. (Of these additional alphabets there is a separate caligraphic design distinct to the script design already included.) Better support for the STIX fonts is planned for an upcoming revision of the package after any problems have been ironed out with the initial version.

# Part I

## User documentation

### Table of Contents

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Acknowledgements</b>	<b>3</b>
<b>3</b>	<b>Getting started</b>	<b>3</b>
3.1	New commands	3
3.2	Package options	5
<b>4</b>	<b>Unicode maths font setup</b>	<b>5</b>
4.1	Using multiple fonts	6
4.2	Script and scriptscript fonts/features	8
4.3	Maths ‘versions’	8
4.4	Legacy maths ‘alphabet’ commands	8
<b>5</b>	<b>Maths input</b>	<b>10</b>
5.1	Math ‘style’	10
5.2	Bold style	11
5.3	Sans serif style	12
5.4	All (the rest) of the mathematical styles	13
5.5	Miscellanea	14
<b>6</b>	<b>Advanced</b>	<b>20</b>
6.1	Warning messages	20
6.2	Programmer’s interface	20
<b>A</b>	<b>stix table data extraction</b>	<b>21</b>
<b>B</b>	<b>Documenting maths support in the NFSS</b>	<b>21</b>
<b>C</b>	<b>Legacy T<sub>E</sub>X font dimensions</b>	<b>23</b>
<b>D</b>	<b>X<sub>Y</sub>T<sub>E</sub>X math font dimensions</b>	<b>23</b>

---

## 1 Introduction

This document describes the unicode-math package, which is an *experimental* implementation of a macro to Unicode glyph encoding for mathematical characters.

Users who desire to specify maths alphabets only (Greek and Latin letters, and Arabic numerals) may wish to use Andrew Moschou’s mathspec package instead. (X<sub>Y</sub>TeX-only at time of writing.)

## 2 Acknowledgements

Many thanks to: Microsoft for developing the mathematics extension to OpenType as part of Microsoft Office 2007; Jonathan Kew for implementing Unicode math support in X<sub>Y</sub>TeX; Taco Hoekwater for implementing Unicode math support in LuaTeX; Barbara Beeton for her prodigious effort compiling the definitive list of Unicode math glyphs and their L<sup>A</sup>TeX names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions; Philipp Stephani for extending the package to support LuaTeX. Ross Moore and Chris Rowley have provided moral and technical support from the very early days with great insight into the issues we face trying to extend and use TeX in the future. Apostolos Syropoulos, Joel Salomon, Khaled Hosny, and Mariusz Wodzicki have been fantastic beta testers.

## 3 Getting started

Load unicode-math as a regular L<sup>A</sup>TeX package. It should be loaded after any other maths or font-related package in case it needs to overwrite their definitions. Here’s an example using the filename syntax to load the TeX Gyre Pagella Math font: (this works for both X<sub>Y</sub>L<sup>A</sup>TeX and LuaL<sup>A</sup>TeX)

```
\usepackage{amsmath} % if desired
\usepackage{unicode-math}
\setmathfont{texgyrepagella-math.otf}
```

Once the package is loaded, traditional TFM-based maths fonts are no longer supported; you can only switch to a different OpenType maths font using the `\setmathfont` command. If you do not load an OpenType maths font before `\begin{document}`, Latin Modern Math (see above) will be loaded automatically.

### 3.1 New commands

unicode-math provides a number of commands (such as `\symbfsf`) to select specific ‘symbol alphabets’ within the unicode maths font, with usage, e.g., `\symbfsf{g}` → **g**. The full listing is shown in Table 1. For backwards compatibility, many of these are also defined with ‘familiar’ synonyms such as `\mathbfsf`. However, where possible the ‘sym’ prefix commands should be preferred, as certain synonyms may become deprecated in time.

Table 1: New unicode-math commands.

unicode-math command	Synonym
<code>\symup</code>	
<code>\symit</code>	
<code>\symbf</code>	
<code>\symsf</code>	
<code>\symtt</code>	
<code>\symnormal</code>	<code>\mathnormal</code>
<code>\symliteral</code>	
<code>\symbfup</code>	<code>\mathbfup</code>
<code>\symbfit</code>	<code>\mathbfit</code>
<code>\symsfup</code>	<code>\mathsfup</code>
<code>\symsfit</code>	<code>\mathsfit</code>
<code>\symbfsfup</code>	<code>\mathbfsfup</code>
<code>\symbfsfit</code>	<code>\mathbfsfit</code>
<code>\symbfsf</code>	<code>\mathbfsf</code>
<code>\symbb</code>	<code>\mathbb</code>
<code>\symbbit</code>	<code>\mathbbit</code>
<code>\symscr</code>	<code>\mathscr</code>
<code>\symbfscr</code>	<code>\mathbfscr</code>
<code>\symcal</code>	<code>\mathcal</code>
<code>\symbfcal</code>	<code>\mathbfcal</code>
<code>\symfrak</code>	<code>\mathfrak</code>
<code>\symbffrak</code>	<code>\mathbffrak</code>

While most alphabet commands are provided with the `\math...` prefix synonyms, there are five ‘legacy’ font alphabets that intentionally behave somewhat different. These are `\mathup`, `\mathit`, `\mathbf`, `\mathsf`, and `\mathtt`. (N.B.: `\mathrm` is defined as a synonym for `\mathup`, but the latter is preferred as it is a script-agnostic term.)

These commands have ‘overloaded’ meanings in traditional L<sup>A</sup>T<sub>E</sub>X, and it’s important to consider the subtle differences between, e.g., the new `\symbf` and `\mathbf`. The `\symbf` command switches to single-letter mathematical symbols (generally within the same OpenType font). The `\mathbf` command switches to a text font that is set up to behave correctly in mathematics, and should be used for multi-letter identifiers. These could be denoted ‘text math alphabets’; further details are discussed in section §4.4. Additional similar ‘text math alphabet’ commands can be defined using the `\setmathfontface` command discussed in section §4.4. To control the behaviour of the default text math alphabet commands to behave in a backwards-compatible mode, see the package options described in section §4.4.2.

### 3.2 Package options

Package options may be set when the package is loaded or at any later stage with the `\unimathsetup` command. Therefore, the following two examples are equivalent:

```
\usepackage[math-style=TeX]{unicode-math}
% OR
\usepackage{unicode-math}
\unimathsetup{math-style=TeX}
```

Note, however, that some package options affect how maths is initialised and changing an option such as `math-style` will not take effect until a new maths font is set up.

Package options may *also* be used when declaring new maths fonts, passed via options to the `\setmathfont` command. Therefore, the following two examples are equivalent:

```
\unimathsetup{math-style=TeX}
\setmathfont{Cambria Math}
% OR
\setmathfont{Cambria Math}[math-style=TeX]
```

A summary list of package options is shown in table 2. See following sections for more information.

## 4 Unicode maths font setup

In the ideal case, a single Unicode font will contain all maths glyphs we need. The file `unicode-math-table.tex` (based on Barbara Beeton’s `stix` table) provides

Table 2: Package options.

Option	Description	See...
<code>math-style</code>	Style of letters	section §5.1
<code>bold-style</code>	Style of bold letters	section §5.2
<code>sans-style</code>	Style of sans serif letters	section §5.3
<code>nabla</code>	Style of the nabla symbol	section §5.5.1
<code>partial</code>	Style of the partial symbol	section §5.5.2
<code>colon</code>	Behaviour of <code>\colon</code>	section §5.5.5
<code>slash-delimiter</code>	Glyph to use for ‘stretchy’ slash	section §5.5.6

Table 3: Maths font options.

Option	Description	See...
<code>range</code>	Style of letters	section §4.1
<code>script-font</code>	Font to use for sub- and super-scripts	section §4.2
<code>script-features</code>	Font features for sub- and super-scripts	section §4.2
<code>sscript-font</code>	Font to use for nested sub- and super-scripts	section §4.2
<code>sscript-features</code>	Font features for nested sub- and super-scripts	section §4.2

the mapping between Unicode maths glyphs and macro names (all 3298 — or however many — of them!). A single command

$$\setmathfont{\langle font name \rangle}[\langle font features \rangle]$$

implements this for every every symbol and alphabetic variant. That means  $x$  to  $x$ ,  $\xi$  to  $\xi$ ,  $\leq$  to  $\leq$ , etc.,  $\mathscr{H}$  to  $\mathscr{H}$  and so on, all for Unicode glyphs within a single font.

This package deals well with Unicode characters for maths input. This includes using literal Greek letters in formulae, resolving to upright or italic depending on preference.

Font features specific to unicode-math are shown in table 3. Package options (see table 2) may also be used. Other fontspec features are also valid.

#### 4.1 Using multiple fonts

There will probably be few cases where a single Unicode maths font suffices (simply due to glyph coverage). The `stix` font comes to mind as a possible exception. It will therefore be necessary to delegate specific Unicode ranges of glyphs to separate fonts:

$$\setmathfont{\langle font name \rangle}[range=\langle unicode range \rangle, \langle font features \rangle]$$

where  $\langle unicode range \rangle$  is a comma-separated list of Unicode slot numbers and ranges such as `{ "27D0-"27EB, "27FF, "295B-"297F }`. Note that TeX’s syntax for accessing the slot number of a character, such as `\+`, will also work here.

You may also use the macro for accessing the glyph, such as `\int`, or whole collection of symbols with the same math type, such as `\mathopen`, or complete

math styles such as `\symbb`. (Only numerical slots, however, can be used in ranged declarations.)

#### 4.1.1 Control over alphabet ranges

As discussed earlier, Unicode mathematics consists of a number of ‘alphabet styles’ within a single font. In `unicode-math`, these ranges are indicated with the following (hopefully self-explanatory) labels:

```
up, it, tt, bfup, bfit, bb, bbit, scr, bfscr, cal, bfcalf,
frak, bffrak, sfup, sfit, bfsfup, bfsfit, bfsf
```

Fonts can be selected for specified ranges only using the following syntax, in which case all other maths font setup remains untouched:

- `[range=bb]` to use the font for ‘bb’ letters only.
- `[range=bfsfit/{greek,Greek}]` for Greek lowercase and uppercase only (also with `latin`, `Latin`, `num` as possible options for Latin lower-/upper-case and numbers, resp.).
- `[range=up->sfup]` to map to different output styles.

Note that ‘meta-styles’ such as ‘bf’ and ‘sf’ are not included here since they are context dependent. Use `[range=bfup]` and `[range=bfit]` to effect changes to the particular ranges selected by ‘bf’ (and similarly for ‘sf’).

If a particular math style is not defined in the font, we fall back onto the lower-base plane (i.e., ‘upright’) glyphs. Therefore, to use an `ASCII`-encoded fractur font, for example, write

```
\setmathfont{SomeFrakturFont}[range=frak]
```

and because the math plane fractur glyphs will be missing, `unicode-math` will know to use the `ASCII` ones instead. If necessary this behaviour can be forced with `[range=frak->up]`, since the ‘up’ range corresponds to `ASCII` letters.

Users of the impressive Minion Math fonts (commercial) may use remapping to access the bold glyphs using:

```
\setmathfont{MinionMath-Regular.otf}
\setmathfont{MinionMath-Bold.otf}[range={bfup->up,bfit->it}]
```

To set up the complete range of optical sizes for these fonts, a font declaration such as the following may be used: (adjust may be desired according to the font size of the document)

```
\setmathfont{Minion Math}[
SizeFeatures = {
  {Size = -6.01, Font = MinionMath-Tiny},
  {Size = 6.01-8.41, Font = MinionMath-Capt},
  {Size = 8.41-13.01, Font = MinionMath-Regular},
  {Size = 13.01-19.91, Font = MinionMath-Subh},
```

```

{Size = 19.91-,      Font = MinionMath-Disp}
}]

\setmathfont{Minion Math}[range = {bfup->up,bfit->it},
SizeFeatures = {
{Size =      -6.01,  Font = MinionMath-BoldTiny},
{Size =   6.01-8.41, Font = MinionMath-BoldCapt},
{Size =   8.41-13.01, Font = MinionMath-Bold},
{Size =  13.01-19.91, Font = MinionMath-BoldSubh},
{Size =  19.91-,     Font = MinionMath-BoldDisp}
}]

```

**v0.8:** Note that in previous versions of unicode-math, these features were labelled `[range=\mathbb]` and so on. This old syntax is still supported for backwards compatibility, but is now discouraged.

## 4.2 Script and scriptscript fonts/features

Cambria Math uses OpenType font features to activate smaller optical sizes for scriptsize and scriptscriptsize symbols (the  $B$  and  $C$ , respectively, in  $A_{B_C}$ ). Other typefaces (such as Minion Math) may use entirely separate font files.

The features `script-font` and `sscript-font` allow alternate fonts to be selected for the script and scriptscript sizes, and `script-features` and `sscript-features` to apply different OpenType features to them.

By default `script-features` is defined as `Style=MathScript` and `sscript-features` is `Style=MathScriptScript`. These correspond to the two levels of OpenType's `ssty` feature tag. If the (s)script-features options are specified manually, you must additionally specify the `Style` options as above.

## 4.3 Maths ‘versions’

L<sup>A</sup>T<sub>E</sub>X uses a concept known as ‘maths versions’ to switch math fonts mid-document. This is useful because it is more efficient than loading a complete maths font from scratch every time—especially with thousands of glyphs in the case of Unicode maths! The canonical example for maths versions is to select a ‘bold’ maths font which might be suitable for section headings, say. (Not everyone agrees with this typesetting choice, though; be careful.)

To select a new maths font in a particular version, use the syntax

```
\setmathfont{<font name>}[version=<version name>,<font features>]
```

and to switch between maths versions mid-document use the standard L<sup>A</sup>T<sub>E</sub>X command `\mathversion{<version name>}`.

## 4.4 Legacy maths ‘alphabet’ commands

L<sup>A</sup>T<sub>E</sub>X traditionally uses `\DeclareMathAlphabet` and `\SetMathAlphabet` to define document commands such as `\mathit`, `\mathbf`, and so on. While these commands



can still be used, `unicode-math` defines a wrapper command to assist with the creation of new such maths alphabet commands. This command is known as `\setmathface` in symmetry with `fontspec`'s `\newfontface` command; it takes syntax:

```
\setmathfontface<command>{\font name}[\font features]
\setmathfontface<command>{\font name}[version=<version name>,\font features]
```

For example, if you want to define a new legacy maths alphabet font `\mathittt`:

```
\setmathfontface\mathittt{texgyrecursor-italic.otf}
...
$\mathittt{foo} = \mathittt{a} + \mathittt{b}$
```

#### 4.4.1 Default ‘text math’ fonts

The five ‘text math’ fonts, discussed above, are: `\mathrm`, `\mathbf`, `\mathit`, `\mathsf`, and `\mathtt`. These commands are also defined with their original definition under synonyms `\mathtextrm`, `\mathtextbf`, and so on.

When selecting document fonts using `fontspec` commands such as `\setmainfont`, `unicode-math` inserts some additional code into `fontspec` that keeps the current default fonts ‘in sync’ with their corresponding `\mathrm` commands, etc.

For example, in standard  $\text{\LaTeX}$ , `\mathsf` doesn’t change even if the main document font is changed using `\renewcommand\sfddefault{...}`. With `unicode-math` loaded, after writing `\setsansfont{Helvetica}`, `\mathsf` will now be set in Helvetica.

If the `\mathsf` font is set explicitly at any time in the preamble, this ‘auto-following’ does not occur. The legacy math font switches can be defined either with commands defined by `fontspec` (`\setmathrm`, `\setmathsf`, etc.) or using the more general `\setmathfontface\mathsf` interface defined by `unicode-math`.

#### 4.4.2 Replacing ‘text math’ fonts by symbols

For certain types of documents that use legacy input syntax (say you’re typesetting a new version of a book written in the 1990s), it would be preferable to use `\symbf` rather than `\mathbf` en masse. For example, if bold maths is used only for vectors and matrices, a dedicated symbol font will produce better spacing and will better match the main math font.

Alternatively, you may have used an old version of `unicode-math` (pre-v0.8), when the `\symXYZ` commands were not defined and `\mathbf` behaved like `\symbf` does now. A series of package options (table 4) are provided to facilitate switching the definition of `\mathXYZ` for the five legacy text math font definitions.

#### 4.4.3 Operator font

$\text{\LaTeX}$  defines an internal command `\operator@font` for typesetting elements such as `\sin` and `\cos`. This font is selected from the legacy operators NFSS ‘MathAlphabet’, which is no longer relevant in the context of `unicode-math`. By default, the

Table 4: Maths text font configuration options. Note that `\mathup` and `\mathrm` are aliases of each other and cannot be configured separately.

Defaults (from ‘text’ font)	From ‘maths symbols’
<code>\mathrm=text</code>	<code>\mathrm=sym</code>
<code>\mathup=text*</code>	<code>\mathup=sym*</code>
<code>\mathit=text</code>	<code>\mathit=sym</code>
<code>\mathsf=text</code>	<code>\mathsf=sym</code>
<code>\mathbf=text</code>	<code>\mathbf=sym</code>
<code>\mathtt=text</code>	<code>\mathtt=sym</code>

`\operator@font` command is defined to switch to the `\mathrm` font. You may now change these using the command:

```
\setoperatorfont\mathit
```

Or, to select a unicode-math range:

```
\setoperatorfont\symscr
```

For example, after the latter above, `\sin x` will produce ‘*sin x*’.

## 5 Maths input

X<sub>Y</sub>TeX’s Unicode support allows maths input through two methods. Like classical T<sub>E</sub>X, macros such as `\alpha`, `\sum`, `\pm`, `\leq`, and so on, provide verbose access to the entire repertoire of characters defined by Unicode. The literal characters themselves may be used instead, for more readable input files.

### 5.1 Math ‘style’

Classically, T<sub>E</sub>X uses italic lowercase Greek letters and *upright* uppercase Greek letters for variables in mathematics. This is contrary to the iso standards of using italic forms for both upper- and lowercase. Furthermore, in various historical contexts, often associated with French typesetting, it was common to use upright uppercase *Latin* letters as well as upright upper- and lowercase Greek, but italic lowercase latin. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

The unicode-math package accommodates these possibilities with the option `math-style` that takes one of four (case sensitive) arguments: `TeX`, `ISO`, `french`, or `upright`.<sup>1</sup> The `math-style` options’ effects are shown in brief in table 5.

The philosophy behind the interface to the mathematical symbols lies in L<sup>A</sup>T<sub>E</sub>X’s attempt of separating content and formatting. Because input source text may come from a variety of places, the upright and ‘mathematical’ italic Latin and

<sup>1</sup>Interface inspired by Walter Schmidt’s `lucimatx` package.

Table 5: Effects of the `math-style` package option.

Package option	Example	
	Latin	Greek
<code>math-style=ISO</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=TeX</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=french</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=upright</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$

Greek alphabets are *unified* from the point of view of having a specified meaning in the source text. That is, to get a mathematical ‘ $x$ ’, either the ASCII (‘keyboard’) letter `x` may be typed, or the actual Unicode character may be used. Similarly for Greek letters. The upright or italic forms are then chosen based on the `math-style` package option.

If glyphs are desired that do not map as per the package option (for example, an upright ‘ $g$ ’ is desired but typing `$g$` yields ‘ $g$ ’), *markup* is required to specify this; to follow from the example: `\symup{g}`. Maths style commands such as `\symup` are detailed later.

*‘Literal’ interface* Some may not like this convention of normalising their input. For them, an upright `x` is an upright ‘ $x$ ’ and that’s that. (This will be the case when obtaining source text from copy/pasting PDF or Microsoft Word documents, for example.) For these users, the `literal` option to `math-style` will effect this behaviour. The `\symliteral{<syms>}` command can also be used, regardless of package setting, to force the style to match the literal input characters. This is a ‘mirror’ to `\symnormal{<syms>}` (also alias `\mathnormal`) which ‘resets’ the character mapping in its argument to that originally set up through package options.

## 5.2 Bold style

Similar as in the previous section, ISO standards differ somewhat to  $\TeX$ ’s conventions (and classical typesetting) for ‘boldness’ in mathematics. In the past, it has been customary to use bold *upright* letters to denote things like vectors and matrices. For example,  $\mathbf{M} = (M_x, M_y, M_z)$ . Presumably, this was due to the relatively scarcity of bold italic fonts in the pre-digital typesetting era. It has been suggested by some that *italic* bold symbols should be used nowadays instead, but this practise is certainly not widespread.

Bold Greek letters have simply been bold variant glyphs of their regular weight, as in  $\boldsymbol{\zeta} = (\zeta_r, \zeta_\phi, \zeta_\theta)$ . Confusingly, the syntax in  $\LaTeX$  traditionally has been different for obtaining ‘normal’ bold symbols in Latin and Greek: `\mathbf` in the former (‘ $\mathbf{M}$ ’), and `\bm` (or `\boldsymbol`, deprecated) in the latter (‘ $\boldsymbol{\zeta}$ ’).

In unicode-math, the `\symbf` command works directly with both Greek and Latin maths characters and depending on package option either switches to upright for Latin letters (`bold-style=TeX`) as well or keeps them italic (`bold-`

Table 6: Effects of the bold-style package option.

Package option	Example	
	Latin	Greek
<code>bold-style=ISO</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$
<code>bold-style=TeX</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Xi})$
<code>bold-style=upright</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\alpha, \beta, \Gamma, \Xi)$

`style=ISO`). To match the package options for non-bold characters, with option `bold-style=upright` all bold characters are upright, and `bold-style=literal` does not change the upright/italic shape of the letter. The bold-style options' effects are shown in brief in table 6.

Upright and italic bold mathematical letters input as direct Unicode characters are normalised with the same rules. For example, with `bold-style=TeX`, a literal bold italic latin character will be typeset upright.

Note that `bold-style` is independent of `math-style`, although if the former is not specified then matching defaults are chosen based on the latter.

### 5.3 Sans serif style

Unicode contains upright and italic, medium and bold mathematical style characters. These may be explicitly selected with the `\mathsfup`, `\mathsfif`, `\mathbfsup`, and `\mathbfsif` commands discussed in section §5.4.

How should the generic `\mathsf` behave? Unlike bold, sans serif is used much more sparingly in mathematics. I've seen recommendations to typeset tensors in sans serif italic or sans serif italic bold (e.g., examples in the `isomath` and `mattens` packages). But L<sup>A</sup>T<sub>E</sub>X's `\mathsf` is *upright* sans serif.

Therefore I reluctantly add the package options `[sans-style=upright]` and `[sans-style=italic]` to control the behaviour of `\mathsf`. The upright style sets up the command to use upright sans serif, including Greek; the italic style switches to using italic in both Latin and Greek. In other words, this option simply changes the meaning of `\mathsf` to either `\mathsfup` or `\mathsfif`, respectively. Please let me know if more granular control is necessary here.

There is also a `[sans-style=literal]` setting, set automatically with `[math-style=literal]`, which retains the uprightness of the input characters used when selecting the sans serif output.

#### 5.3.1 What about bold sans serif?

While you might want your bold upright and your sans serif italic, I don't believe you'd also want your bold sans serif upright (or all vice versa, if that's even conceivable). Therefore, bold sans serif follows from the setting for sans serif; it is completely independent of the setting for bold.

In other words, `\mathbfsf` is either `\mathbfsup` or `\mathbfsif` based on

Table 7: Mathematical styles defined in Unicode. Black dots indicate an style exists in the font specified; blue dots indicate shapes that should always be taken from the upright font even in the italic style. See main text for description of `\mathbbi`.

Font				Alphabet		
Style	Shape	Series	Switch	Latin	Greek	Numerals
Serif	Upright	Normal	<code>\mathup</code>	•	•	•
		Bold	<code>\mathbfup</code>	•	•	•
	Italic	Normal	<code>\mathit</code>	•	•	•
		Bold	<code>\mathbfit</code>	•	•	•
Sans serif	Upright	Normal	<code>\mathsfup</code>	•		•
	Italic	Normal	<code>\mathsfit</code>	•		•
	Upright	Bold	<code>\mathbfsfup</code>	•	•	•
	Italic	Bold	<code>\mathbfsfit</code>	•	•	•
Typewriter	Upright	Normal	<code>\mathtt</code>	•		•
Double-struck	Upright	Normal	<code>\mathbb</code>	•		•
	Italic	Normal	<code>\mathbbi</code>	•		
Script	Upright	Normal	<code>\mathscr</code>	•		
		Bold	<code>\mathbfscr</code>	•		
Fraktur	Upright	Normal	<code>\mathfrak</code>	•		
		Bold	<code>\mathbffrac</code>	•		

[`sans-style=upright`] or [`sans-style=italic`], respectively. And [`sans-style = literal`] causes `\mathbfsf` to retain the same italic or upright shape as the input, and turns it bold sans serif.

N.B.: there is no medium-weight sans serif Greek range in Unicode. Therefore, `\symsf{\alpha}` does not make sense (it produces ‘ $\alpha$ ’), while `\symsf{\alpha}` gives ‘ $\alpha$ ’ or ‘ $\alpha$ ’ according to the `sans-style`.

## 5.4 All (the rest) of the mathematical styles

Unicode contains separate codepoints for most if not all variations of style shape one may wish to use in mathematical notation. The complete list is shown in table 7. Some of these have been covered in the previous sections.

The math font switching commands do not nest; therefore if you want sans serif bold, you must write `\symsf{\dots}` rather than `\symsf{\symsf{\dots}}`. This may change in the future.

### 5.4.1 Double-struck

The double-struck style (also known as ‘blackboard bold’) consists of upright Latin letters  $\{a-z, A-Z\}$ , numerals  $0-9$ , summation symbol  $\Sigma$ , and four Greek letters only:  $\{\gamma, \pi, \Gamma, \Pi\}$ .

While `\symsf{\sum}` does produce a double-struck summation symbol, its

limits aren't properly aligned. Therefore, either the literal character or the control sequence `\Bbbsum` are recommended instead.

There are also five Latin *italic* double-struck letters:  $\mathbb{D}\mathbb{d}\mathbb{E}\mathbb{I}\mathbb{J}$ . These can be accessed (if not with their literal characters or control sequences) with the `\mathbb{bit}` style switch, but note that only those five letters will give the expected output.

#### 5.4.2 Caligraphic vs. Script variants

The Unicode maths encoding contains a style for 'Script' letters, and while by default `\mathcal` and `\mathscr` are synonyms, there are some situations when a separate 'Caligraphic' style is needed as well.

If a font contains alternate glyphs for a separate caligraphic style, they can be selected explicitly as shown below. This feature is currently only supported by the XITS Math font, where the caligraphic letters are accessed with the same glyph slots as the script letters but with the first stylistic set feature (ss01) applied.

```
\setmathfont{xits-math.otf}[range={cal,bfcal},StylisticSet=1]
```

An example is shown below.

The Script style (`\mathscr`) in XITS Math is:  $\mathscr{A}\mathscr{B}\mathscr{C}\mathscr{X}\mathscr{Y}\mathscr{Z}$

The Caligraphic style (`\mathcal`) in XITS Math is:  $\mathcal{A}\mathcal{B}\mathcal{C}\mathcal{X}\mathcal{Y}\mathcal{Z}$

### 5.5 Miscellanea

#### 5.5.1 Nabla

The symbol  $\nabla$  comes in the six forms shown in table 8. We want an individual option to specify whether we want upright or italic nabla by default (when either upright or italic nabla is used in the source).  $\TeX$  classically uses an upright nabla, and iso standards agree with this convention. The package options `nabla=upright` and `nabla=italic` switch between the two choices, and `nabla=literal` respects the shape of the input character. This is then inherited through `\sympb`; `\symit` and `\symup` can be used to force one way or the other.

`nabla=italic` is the default. `nabla=literal` is activated automatically after `math-style=literal`.

#### 5.5.2 Partial

The same applies to the symbols  $\partial$  partial differential and  $\partial$  math italic partial differential.

At time of writing, both the Cambria Math and STIX fonts display these two glyphs in the same italic style, but this is hopefully a bug that will be corrected in the future — the 'plain' partial differential should really have an upright shape.

Use the `partial=upright` or `partial=italic` package options to specify which one you would like, or `partial=literal` to have the same character used in the output as was used for the input. The default is (always, unless someone requests

Table 8: The various forms of nabla.

Description		Glyph
Upright	Serif	$\nabla$
	Bold serif	<b><math>\nabla</math></b>
	Bold sans	<b><math>\nabla</math></b>
Italic	Serif	$\nabla$
	Bold serif	<b><math>\nabla</math></b>
	Bold sans	<b><math>\nabla</math></b>

Table 9: The partial differential.

Description		Glyph
Regular	Upright	$\partial$
	Italic	$\partial$
Bold	Upright	<b><math>\partial</math></b>
	Italic	<b><math>\partial</math></b>
Sans bold	Upright	<b><math>\partial</math></b>
	Italic	<b><math>\partial</math></b>

and argues otherwise) `partial=italic`.<sup>2</sup> `partial=literal` is activated following `math-style=literal`.

See table 9 for the variations on the partial differential symbol.

### 5.5.3 Primes

Primes ( $x'$ ) may be input in several ways. You may use any combination the ASCII straight quote (') or the Unicode prime U+2032 ('); when multiple primes occur next to each other, they chain together to form double, triple, or quadruple primes if the font contains pre-drawn glyphs. The individual prime glyphs are accessed, as usual, with the `\prime` command, and the double-, triple-, and quadruple-prime glyphs are available with `\dprime`, `\trprime`, and `\qprime`, respectively.

If the font does not contain the pre-drawn glyphs or more than four primes are used, the single prime glyph is used multiple times with a negative kern to get the spacing right. There is no user interface to adjust this negative kern yet (because I haven't decided what it should look like); if you need to, write something like this:

```
\ExplSyntaxOn
\muskip_gset:Nn \g_@@_primekern_muskip { -\thinmuskip/2 }
\ExplSyntaxOff
```

Backwards or reverse primes behave in exactly the same way; use the ASCII backtick (`) or the Unicode reverse prime U+2035 ('). The command to access the backprime is `\backprime`, and multiple backwards primes can be accessed with `\backdprime`, `\backtrprime`, and `\backqprime`.

In all cases above, no error checking is performed if you attempt to access a multi-prime glyph in a font that doesn't contain one. For this reason, it may be safer to write `x''''` instead of `x\qprime` in general.

If you ever need to enter the straight quote ' or the backtick ` in maths mode, these glyphs can be accessed with `\mathstraightquote` and `\mathbacktick`.

<sup>2</sup>A good argument would revolve around some international standards body recommending upright over italic. I just don't have the time right now to look it up.

A	0	1	2	3	4	5	6	7	8	9	+	-	=	(	)	i	n	n	h	j	r	w	y	Z
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Figure 1: The Unicode superscripts supported as input characters. These are the literal glyphs from Charis SIL, not the output seen when used for maths input. The ‘A’ and ‘Z’ are to provide context for the size and location of the superscript glyphs.

A	0	1	2	3	4	5	6	7	8	9	+	-	=	(	)	a	e	i	o	r	u	v	x	β	γ	ρ	φ	χ	Z
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Figure 2: The Unicode subscripts supported as input characters. See note from figure 1.

#### 5.5.4 Unicode subscripts and superscripts

You may, if you wish, use Unicode subscripts and superscripts in your source document. For basic expressions, the use of these characters can make the input more readable. Adjacent sub- or super-scripts will be concatenated into a single expression.

The range of subscripts and superscripts supported by this package are shown in figures 1 and 2. Please request more if you think it is appropriate.

#### 5.5.5 Colon

The colon is one of the few confusing characters of Unicode maths. In  $\text{T}_{\text{E}}\text{X}$ , `:` is defined as a colon with relation spacing: ‘ $a : b$ ’. While `\colon` is defined as a colon with punctuation spacing: ‘ $a : b$ ’.

In Unicode,  $\text{U}+\text{003A}$  colon is defined as a punctuation symbol, while  $\text{U}+\text{2236}$  ratio is the colon-like symbol used in mathematics to denote ratios and other things.

This breaks the usual straightforward mapping from control sequence to Unicode input character to (the same) Unicode glyph.

To preserve input compatibility, we remap the `ASCII` input character ‘`:`’ to  $\text{U}+\text{2236}$ . Typing a literal  $\text{U}+\text{2236}$  char will result in the same output. If `amsmath` is loaded, then the definition of `\colon` is inherited from there (it looks like a punctuation colon with additional space around it). Otherwise, `\colon` is made to output a colon with `\mathpunct` spacing.

The package option `colon=literal` forces `ASCII` input ‘`:`’ to be printed as `\mathcolon` instead.

#### 5.5.6 Slashes and backslashes

There are several slash-like symbols defined in Unicode. The complete list is shown in table 10.

In regular  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  we can write `\left\slash...\right\backslash` and so on and obtain extensible delimiter-like symbols. Not all of the Unicode slashes are suitable for this (and do not have the font support to do it).



Table 10: Slashes and backslashes.

Slot	Name	Glyph	Command
U+002F	SOLIDUS	/	\slash
U+2044	FRACTION SLASH	/	\fracslash
U+2215	DIVISION SLASH	/	\divslash
U+29F8	BIG SOLIDUS	/	\xsol
U+005C	REVERSE SOLIDUS	\	\backslash
U+2216	SET MINUS	\	\smallsetminus
U+29F5	REVERSE SOLIDUS OPERATOR	\	\setminus
U+29F9	BIG REVERSE SOLIDUS	\	\xbsol

*Slash* Of U+2044 fraction slash, TR25 says that it is:

...used to build up simple fractions in running text...however parsers of mathematical texts should be prepared to handle fraction slash when it is received from other sources.

U+2215 division slash should be used when division is represented without a built-up fraction;  $\pi \approx 22/7$ , for example.

U+29F8 big solidus is a ‘big operator’ (like  $\sum$ ).

*Backslash* The U+005C reverse solidus character \backslash is used for denoting double cosets:  $A \backslash B$ . (So I’m led to believe.) It may be used as a ‘stretchy’ delimiter if supported by the font.

MathML uses U+2216 set minus like this:  $A \setminus B$ .<sup>3</sup> The  $\LaTeX$  command name \smallsetminus is used for backwards compatibility.

Presumably, U+29F5 reverse solidus operator is intended to be used in a similar way, but it could also (perhaps?) be used to represent ‘inverse division’:  $\pi \approx 7 \setminus 22$ .<sup>4</sup> The  $\LaTeX$  name for this character is \setminus.

Finally, U+29F9 big reverse solidus is a ‘big operator’ (like  $\sum$ ).

*How to use all of these things* Unfortunately, font support for the above characters/glyphs is rather inconsistent. In Cambria Math, the only slash that grows (say when writing

$$\left[ \begin{array}{cc} a & b \\ c & d \end{array} \right] / \left[ \begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right] )$$

is the FRACTION SLASH, which we just established above is sort of only supposed to be used in text.

Of the above characters, the following are allowed to be used after \left, \middle, and \right:

<sup>3</sup>§4.4.5.11 <http://www.w3.org/TR/MathML3/>

<sup>4</sup>This is valid syntax in the Octave and Matlab programming languages, in which it means matrix inverse pre-multiplication. I.e.,  $A \setminus B \equiv A^{-1}B$ .

- `\fracdash;`
- `\slash;` and,
- `\backslash` (the only reverse slash).

However, we assume that there is only *one* stretchy slash in the font; this is assumed by default to be U+002F solidus. Writing `\left/` or `\left\slash` or `\left\fracdash` will all result in the same stretchy delimiter being used.

The delimiter used can be changed with the `slash-delimiter` package option. Allowed values are `ascii`, `frac`, and `div`, corresponding to the respective Unicode slots.

For example: as mentioned above, Cambria Math’s stretchy slash is U+2044 fraction slash. When using Cambria Math, then `unicode-math` should be loaded with the `slash-delimiter=frac` option. (This should be a font option rather than a package option, but it will change soon.)

#### 5.5.7 Growing and non-growing accents

There are a few accents for which T<sub>E</sub>X has both non-growing and growing versions. Among these are `\hat` and `\tilde`; the corresponding growing versions are called `\widehat` and `\widetilde`, respectively.

Older versions of X<sub>Y</sub>T<sub>E</sub>X and LuaT<sub>E</sub>X did not support this distinction, however, and *all* accents there were growing automatically. (I.e., `\hat` and `\widehat` are equivalent.) As of LuaT<sub>E</sub>X v0.65 and X<sub>Y</sub>T<sub>E</sub>X v0.9998, these wide/non-wide commands will again behave in their expected manner.

#### 5.5.8 Pre-drawn fraction characters

Pre-drawn fractions U+00BC–U+00BE, U+2150–U+215E are not suitable for use in mathematics output. However, they can be useful as input characters to abbreviate common fractions.

$\frac{1}{4}$   $\frac{1}{2}$   $\frac{3}{4}$   $\frac{0}{3}$   $\frac{1}{7}$   $\frac{1}{9}$   $\frac{1}{10}$   $\frac{1}{3}$   $\frac{2}{3}$   $\frac{1}{5}$   $\frac{2}{5}$   $\frac{3}{5}$   $\frac{4}{5}$   $\frac{1}{6}$   $\frac{5}{6}$   $\frac{1}{8}$   $\frac{3}{8}$   $\frac{5}{8}$   $\frac{7}{8}$

For example, instead of writing `\tfrac{12}{x}`, you may consider it more readable to have `\frac{12}{x}` in the source instead.

If the `\tfrac` command exists (i.e., if `amsmath` is loaded or you have specially defined `\tfrac` for this purpose), it will be used to typeset the fractions. If not, regular `\frac` will be used. The command to use (`\tfrac` or `\frac`) can be forced either way with the package option `active-frac=small` or `active-frac=normalsize`, respectively.

#### 5.5.9 Circles

Unicode defines a large number of different types of circles for a variety of mathematical purposes. There are thirteen alone just considering the all white and all black ones, shown in table 11.

Slot	Command	Glyph	Glyph	Command	Slot
U+00B7	<code>\cdotp</code>	·			
U+22C5	<code>\cdot</code>	·			
U+2219	<code>\vysmblkcircle</code>	•	◦	<code>\vysmwhtcircle</code>	U+2218
U+2022	<code>\smbkcircle</code>	•	◦	<code>\smwhtcircle</code>	U+25E6
U+2981	<code>\mdsmbkcircle</code>	●	◦	<code>\mdsmwhtcircle</code>	U+26AC
U+26AB	<code>\mdblkcircle</code>	●	◯	<code>\mdwhtcircle</code>	U+26AA
U+25CF	<code>\mdlgbkcircle</code>	●	◯	<code>\mdlgwhtcircle</code>	U+25CB
U+2B24	<code>\lgblkcircle</code>	●	◯	<code>\lgwhtcircle</code>	U+25EF

Table 11: Filled and hollow Unicode circles.

Slot	Command	Glyph	Class
U+25B5	<code>\vartriangle</code>	△	binary
U+25B3	<code>\bigtriangleup</code>	△	binary
U+25B3	<code>\triangle</code>	△	ordinary
U+2206	<code>\increment</code>	△	ordinary
U+0394	<code>\mathup\Delta</code>	Δ	ordinary

Table 12: Different upwards pointing triangles.

L<sup>A</sup>T<sub>E</sub>X defines considerably fewer: `\circ` and `\bigcirc` for white; `\bullet` for black. This package maps those commands to `\vysmwhtcircle`, `\mdlgwhtcircle`, and `\smbkcircle`, respectively.

#### 5.5.10 Triangles

While there aren’t as many different sizes of triangle as there are circle, there’s some important distinctions to make between a few similar characters. See table 12 for the full summary.

These triangles all have different intended meanings. Note for backwards compatibility with T<sub>E</sub>X, U+25B3 has *two* different mappings in unicode-math. `\bigtriangleup` is intended as a binary operator whereas `\triangle` is intended to be used as a letter-like symbol.

But you’re better off if you’re using the latter form to indicate an increment to use the glyph intended for this purpose, U+2206:  $\Delta x$ .

Finally, given that  $\triangle$  and  $\Delta$  are provided for you already, it is better off to only use upright Greek Delta  $\Delta$  if you’re actually using it as a symbolic entity such as a variable on its own.

## 6 *Advanced*

### 6.1 *Warning messages*

This package can produce a number of informational messages to try and inform the user when something might be going wrong due to package conflicts or something else. As an experimental feature, these can be turned off on an individual basis with the package option `warnings-off` which takes a comma-separated list of warnings to suppress. A warning will give you its name when printed on the console output; e.g.,

```
* unicode-math warning: "mathtools-colon"
*
* ... <warning message> ...
```

This warning could be suppressed by loading the package as follows:

```
\usepackage[warnings-off={mathtools-colon}]{unicode-math}
```

### 6.2 *Programmer's interface*

(Tentative and under construction.) If you are writing some code that needs to know the current maths style (`\mathbf`, `\mathit`, etc.), you can query the variable `\l_@@_mathstyle_t1`. It will contain the maths style without the leading ‘math’ string; for example, `\sympf { \show \l_@@_mathstyle_t1 }` will produce ‘bf’.

## A *STIX table data extraction*

The source for the  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  names for the very large number of mathematical glyphs are provided via Barbara Beeton’s table file for the `stix` project ([ams.org/STIX](http://ams.org/STIX)). A version is located at <http://www.ams.org/STIX/bnb/stix-tbl.asc> but check <http://www.ams.org/STIX/> for more up-to-date info.

This table is converted into a form suitable for reading by  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ . A single file is produced containing all (more than 3298) symbols. Future optimisations might include generating various (possibly overlapping) subsets so not all definitions must be read just to redefine a small range of symbols. Performance for now seems to be acceptable without such measures.

This file is currently developed outside this DTX file. It will be incorporated when the final version is ready. (I know this is not how things are supposed to work!)

## B *Documenting maths support in the NFSS*

In the following,  $\langle NFSS\ decl. \rangle$  stands for something like  $\{\mathrm{T1}\}\{\mathrm{lmr}\}\{\mathrm{m}\}\{\mathrm{n}\}$ .

**Maths symbol fonts** Fonts for symbols:  $\propto, \leq, \rightarrow$

`\DeclareSymbolFont{<name>}\langle NFSS decl. \rangle`

Declares a named maths font such as operators from which symbols are defined with `\DeclareMathSymbol`.

**Maths alphabet fonts** Fonts for  $ABC-xyz, \mathfrak{ABC}-\mathcal{XYZ}$ , etc.

`\DeclareMathAlphabet{<cmd>}\langle NFSS decl. \rangle`

For commands such as `\mathbf`, accessed through maths mode that are unaffected by the current text font, and which are used for alphabetic symbols in the `ASCII` range.

`\DeclareSymbolFontAlphabet{<cmd>}\{<name>\}`

Alternative (and optimisation) for `\DeclareMathAlphabet` if a single font is being used for both alphabetic characters (as above) and symbols.

**Maths ‘versions’** Different maths weights can be defined with the following, switched in text with the `\mathversion{<maths version>}` command.

`\SetSymbolFont{<name>}\{<maths version>\}\langle NFSS decl. \rangle`

`\SetMathAlphabet{<cmd>}\{<maths version>\}\langle NFSS decl. \rangle`

**Maths symbols** Symbol definitions in maths for both characters (=) and macros (`\eqdef`): `\DeclareMathSymbol{<symbol>}\{<type>\}\{<named font>\}\{<slot>\}` This is the macro that actually defines which font each symbol comes from and how they behave.

Delimiters and radicals use wrappers around  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ’s `\delimiter/\radical` primitives, which are re-designed in  $\mathrm{X}_{\mathrm{Y}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ . The syntax used in  $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ’s NFSS is therefore not so relevant here.

**Delimiters** A special class of maths symbol which enlarge themselves in certain contexts.

```
\DeclareMathDelimiter{<symbol>}{<type>}{<sym.font>}{<slot>}{<sym.font>}{<slot>}
```

**Radicals** Similar to delimiters (`\DeclareMathRadical` takes the same syntax) but behave ‘weirdly’.

In those cases, glyph slots in *two* symbol fonts are required; one for the small (‘regular’) case, the other for situations when the glyph is larger. This is not the case in  $\text{\LaTeX}$ .

Accents are not included yet.

*Summary* For symbols, something like:

```
\def\DeclareMathSymbol#1#2#3#4{
  \global\mathchardef#1"\mathchar@type#2
  \expandafter\hexnumber@\csname sym#2\endcsname
  {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}}
```

For characters, something like:

```
\def\DeclareMathSymbol#1#2#3#4{
  \global\mathcode`#1"\mathchar@type#2
  \expandafter\hexnumber@\csname sym#2\endcsname
  {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}}
```

## C Legacy T<sub>E</sub>X font dimensions

Text fonts		Maths font, \fam2		Maths font, \fam3	
$\phi_1$	slant per pt	$\sigma_5$	x height	$\zeta_8$	default rule thickness
$\phi_2$	interword space	$\sigma_6$	quad	$\zeta_9$	big op spacing1
$\phi_3$	interword stretch	$\sigma_8$	num1	$\zeta_{10}$	big op spacing2
$\phi_4$	interword shrink	$\sigma_9$	num2	$\zeta_{11}$	big op spacing3
$\phi_5$	x-height	$\sigma_{10}$	num3	$\zeta_{12}$	big op spacing4
$\phi_6$	quad width	$\sigma_{11}$	denom1	$\zeta_{13}$	big op spacing5
$\phi_7$	extra space	$\sigma_{12}$	denom2		
$\phi_8$	cap height (X <sub>Y</sub> T <sub>E</sub> X only)	$\sigma_{13}$	sup1		
		$\sigma_{14}$	sup2		
		$\sigma_{15}$	sup3		
		$\sigma_{16}$	sub1		
		$\sigma_{17}$	sub2		
		$\sigma_{18}$	sup drop		
		$\sigma_{19}$	sub drop		
		$\sigma_{20}$	delim1		
		$\sigma_{21}$	delim2		
		$\sigma_{22}$	axis height		

## D X<sub>Y</sub>T<sub>E</sub>X math font dimensions

These are the extended \fontdimens available for suitable fonts in X<sub>Y</sub>T<sub>E</sub>X. Note that LuaT<sub>E</sub>X takes an alternative route, and this package will eventually provide a wrapper interface to the two (I hope).

\fontdimen	Dimension name	Description
10	SCRIPTPERCENTSCALEDOWN	Percentage of scaling down for script level 1. Suggested value: 80%.
11	SCRIPTSCRIPTPERCENTSCALEDOWN	Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%.
12	DELIMITEDSUBFORMULAMINHEIGHT	Minimum height required for a delimited expression to be treated as a subformula. Suggested value: normal line height $\times$ 1.5.
13	DISPLAYOPERATORMINHEIGHT	Minimum height of n-ary operators (such as integral and summation) for formulas in display mode.

\fontdimen	Dimension name	Description
14	MATHLEADING	White space to be left between math formulas to ensure proper line spacing. For example, for applications that treat line gap as a part of line ascender, formulas with ink going above (os2.sTypoAscender + os2.sTypoLineGap – MathLeading) or with ink going below os2.sTypoDescender will result in increasing line height.
15	AxisHEIGHT	Axis height of the font.
16	ACCENTBASEHEIGHT	Maximum (ink) height of accent base that does not require raising the accents. Suggested: x-height of the font (os2.sxHeight) plus any possible overshots.
17	FLATTENEDACCENTBASE-HEIGHT	Maximum (ink) height of accent base that does not require flattening the accents. Suggested: cap height of the font (os2.sCapHeight).
18	SUBSCRIPTSHIFTDOWN	The standard shift down applied to subscript elements. Positive for moving in the downward direction. Suggested: os2.ySubscriptYOffset.
19	SUBSCRIPTTOPMAX	Maximum allowed height of the (ink) top of subscripts that does not require moving subscripts further down. Suggested: $\frac{1}{5}$ x-height.
20	SUBSCRIPTBASELINEDROPMIN	Minimum allowed drop of the baseline of subscripts relative to the (ink) bottom of the base. Checked for bases that are treated as a box or extended shape. Positive for subscript baseline dropped below the base bottom.
21	SUPERSCRIPSHIFTUP	Standard shift up applied to superscript elements. Suggested: os2.ySuperscriptYOffset.
22	SUPERSCRIPSHIFTUPCRAMPED	Standard shift of superscripts relative to the base, in cramped style.
23	SUPERSCRIPBOTTOMMIN	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: $\frac{1}{4}$ x-height.



\fontdimen	Dimension name	Description
24	SUPERSCRIPTBASELINEDROP-MAX	Maximum allowed drop of the baseline of superscripts relative to the (ink) top of the base. Checked for bases that are treated as a box or extended shape. Positive for superscript baseline below the base top.
25	SUBSUPERSCRIPGAPMIN	Minimum gap between the superscript and subscript ink. Suggested: 4×default rule thickness.
26	SUPERSCRIPBTOMMAX-WITHSUBSCRIPT	The maximum level to which the (ink) bottom of superscript can be pushed to increase the gap between superscript and subscript, before subscript starts being moved down. Suggested: /5 x-height.
27	SPACEAFTERSCRIPT	Extra white space to be added after each subscript and superscript. Suggested: 0.5pt for a 12 pt font.
28	UPPERLIMITGAPMIN	Minimum gap between the (ink) bottom of the upper limit, and the (ink) top of the base operator.
29	UPPERLIMITBASELINERISEMIN	Minimum distance between baseline of upper limit and (ink) top of the base operator.
30	LOWERLIMITGAPMIN	Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator.
31	LOWERLIMITBASELINEDROP-MIN	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	STACKTOPSHIFTUP	Standard shift up applied to the top element of a stack.
33	STACKTOPDISPLAYSTYLESHIFT-UP	Standard shift up applied to the top element of a stack in display style.
34	STACKBOTTOMSHIFTDOWN	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.
35	STACKBOTTOMDISPLAYSTYLE-SHIFTDOWN	Standard shift down applied to the bottom element of a stack in display style. Positive for moving in the downward direction.
36	STACKGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element. Suggested: 3×default rule thickness.

\fontdimen	Dimension name	Description
37	STACKDISPLAYSTYLEGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element in display style. Suggested: 7×default rule thickness.
38	STRETCHSTACKTOPSHIFTUP	Standard shift up applied to the top element of the stretch stack.
39	STRETCHSTACKBOTTOMSHIFT-DOWN	Standard shift down applied to the bottom element of the stretch stack. Positive for moving in the downward direction.
40	STRETCHSTACKGAPABOVEMIN	Minimum gap between the ink of the stretched element, and the (ink) bottom of the element above. Suggested: UpperLimitGapMin
41	STRETCHSTACKGAPBELOWMIN	Minimum gap between the ink of the stretched element, and the (ink) top of the element below. Suggested: LowerLimitGapMin.
42	FRACTIONNUMERATORSHIFTUP	Standard shift up applied to the numerator.
43	FRACTIONNUMERATOR-DISPLAYSTYLESHIFTUP	Standard shift up applied to the numerator in display style. Suggested: StackTopDisplayStyleShiftUp.
44	FRACTIONDENOMINATORSHIFT-DOWN	Standard shift down applied to the denominator. Positive for moving in the downward direction.
45	FRACTIONDENOMINATOR-DISPLAYSTYLESHIFTDOWN	Standard shift down applied to the denominator in display style. Positive for moving in the downward direction. Suggested: StackBottomDisplayStyleShiftDown.
46	FRACTIONNUMERATORGAP-MIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness
47	FRACTIONNUMDISPLAYSTYLE-GAPMIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
48	FRACTIONRULETHICKNESS	Thickness of the fraction bar. Suggested: default rule thickness.

\fontdimen	Dimension name	Description
49	FRACTIONDENOMINATORGAP-MIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness
50	FRACTIONDENOMDISPLAY-STYLEGAPMIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
51	SKEWEDFRACTION-HORIZONTALGAP	Horizontal distance between the top and bottom elements of a skewed fraction.
52	SKEWEDFRACTIONVERTICAL-GAP	Vertical distance between the ink of the top and bottom elements of a skewed fraction.
53	OVERBARVERTICALGAP	Distance between the overbar and the (ink) top of the base. Suggested: 3×default rule thickness.
54	OVERBARRULETHICKNESS	Thickness of overbar. Suggested: default rule thickness.
55	OVERBAREXTRAASCENDER	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	UNDERBARVERTICALGAP	Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness.
57	UNDERBARRULETHICKNESS	Thickness of underbar. Suggested: default rule thickness.
58	UNDERBAREXTRADESCENDER	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.
59	RADICALVERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: 1¼ default rule thickness.
60	RADICALDISPLAYSTYLE-VERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.
61	RADICALRULETHICKNESS	Thickness of the radical rule. This is the thickness of the rule in designed or constructed radical signs. Suggested: default rule thickness.
62	RADICALEXTRAASCENDER	Extra white space reserved above the radical. Suggested: RadicalRuleThickness.

\fontdimen	Dimension name	Description
63	RADICALKERNBEFOREDEGREE	Extra horizontal kern before the degree of a radical, if such is present. Suggested: 5/18 of em.
64	RADICALKERNAFTERDEGREE	Negative kern after the degree of a radical, if such is present. Suggested: -10/18 of em.
65	RADICALDEGREEBOTTOM- RAISEPERCENT	Height of the bottom of the radical degree, if such is present, in proportion to the ascender of the radical sign. Suggested: 60%.

## Part II

# Package implementation

## Table of Contents

---

<b>E</b>	<b>Bifurcation</b>	<b>30</b>
E.1	Engine differences	30
<b>F</b>	<b>Fundamentals</b>	<b>30</b>
F.1	Setting math chars, math codes, etc.	30
F.2	<code>\setmathalphabet</code>	34
F.3	Hooks into fontspec	35
F.4	The main <code>\setmathfont</code> macro	36
F.5	(Big) operators	45
F.6	Radicals	46
F.7	Maths accents	46
F.8	Common interface for font parameters	46
<b>G</b>	<b>Font features</b>	<b>51</b>
G.1	Math version	51
G.2	Script and scriptscript font options	51
G.3	Range processing	51
G.4	Resolving Greek symbol name control sequences	55
<b>H</b>	<b>Maths alphabets</b>	<b>55</b>
H.1	Hooks into $\LaTeX 2_{\epsilon}$	56
H.2	Setting styles	56
H.3	Defining the math style macros	57
H.4	Definition of alphabets and styles	58
H.5	Defining the math alphabets per style	62
H.6	Mapping ‘naked’ math characters	64
H.7	Mapping chars inside a math style	66
<b>I</b>	<b>A token list to contain the data of the math table</b>	<b>69</b>
<b>J</b>	<b>Definitions of the active math characters</b>	<b>69</b>
<b>K</b>	<b>Preamble</b>	<b>70</b>
K.1	Extras	72
K.2	Alphabet Unicode positions	74
K.3	Package options	74
K.4	Programmers’ interface	79
K.5	Overcoming <code>\@onlypreamble</code>	79
<b>L</b>	<b>Error messages</b>	<b>80</b>

L.1	Alphabet Unicode positions	82
L.2	STIX fonts	88
L.3	Alphabets	92
L.4	Compatibility	109

---

The prefix for unicode-math is um:

```
1 <@=um>
```

We (later on) bifurcate the package based on the engine being used. These separate package files are indicated with the Docstrip flags LU and XE, respectively. Shared code executed before loading the engine-specific code is indicated with the flag preamble.

```
2 <*load>
3 \sys_if_engine luatex:T { \RequirePackage{unicode-math-luatex} }
4 \sys_if_engine xetex:T { \RequirePackage{unicode-math-xetex} }
5 </load>
```

## E Bifurcation

And here the split begins. Most of the code is still shared, but code for Lua $\TeX$  uses the ‘LU’ flag and code for Xe $\TeX$  uses ‘XE’.

```
6 <*package&(XE|LU)>
7 \ExplSyntaxOn
```

### E.1 Engine differences

Xe $\TeX$  before version 0.9999 did not support \U prefix for extended math primitives, and while Lua $\TeX$  had it from the start, prior 0.75.0 the L $\TeX$  format did not provide them without the \luatex prefix. We assume that users of unicode-math are using up-to-date engines however.

```
8 <*LU>
9 \RequirePackage{luaotfload} [2014/05/18]
10 \RequirePackage{lualatex-math}[2011/08/07]
11 </LU>
```

## F Fundamentals

### F.1 Setting math chars, math codes, etc.

```
\@@_set_mathsymbol:nNNn #1 : A L $\TeX$  symbol font, e.g., operators
                        #2 : Symbol macro, e.g., \alpha
                        #3 : Type, e.g., \mathalpha
                        #4 : Slot, e.g., "221E
```

There are a bunch of tests to perform to process the various characters. The following assignments should all be fairly straightforward.

The catcode setting is to work around (strange?) behaviour in LuaTeX in which catcode 11 characters don't have italic correction for maths. We don't adjust ascii chars, however, because certain punctuation should not have their catcodes changed.

```

12 \cs_set:Nn \@@_set_mathsymbol:nNn
13 {
14   \bool_lazy_and:nnT
15   {
16     \int_compare_p:nNn {#4} > {127}
17   }
18   {
19     \int_compare_p:nNn { \char_value_catcode:n {#4} } = {11}
20   }
21   { \char_set_catcode_other:n {#4} }
22 }
23 \tl_case:Nn #3
24 {
25   \mathord { \@@_set_mathcode:nnn {#4} {#3} {#1} }
26   \mathalpha { \@@_set_mathcode:nnn {#4} {#3} {#1} }
27   \mathbin { \@@_set_mathcode:nnn {#4} {#3} {#1} }
28   \mathrel { \@@_set_mathcode:nnn {#4} {#3} {#1} }
29   \mathpunct { \@@_set_mathcode:nnn {#4} {#3} {#1} }
30   \mathop { \@@_set_big_operator:nnn {#1} {#2} {#4} }
31   \mathopen { \@@_set_math_open:nnn {#1} {#2} {#4} }
32   \mathclose { \@@_set_math_close:nnn {#1} {#2} {#4} }
33   \mathfence { \@@_set_math_fence:nnnn {#1} {#2} {#3} {#4} }
34   \mathaccent
35   { \@@_set_math_accent:Nnnn #2 {fixed} {#1} {#4} }
36   \mathbotaccent
37   { \@@_set_math_accent:Nnnn #2 {bottom~ fixed} {#1} {#4} }
38   \mathaccentwide
39   { \@@_set_math_accent:Nnnn #2 {} {#1} {#4} }
40   \mathbotaccentwide
41   { \@@_set_math_accent:Nnnn #2 {bottom} {#1} {#4} }
42   \mathover
43   { \@@_set_math_overunder:Nnnn #2 {} {#1} {#4} }
44   \mathunder
45   { \@@_set_math_overunder:Nnnn #2 {bottom} {#1} {#4} }
46 }
47 }

48 \edef\mathfence{\string\mathfence}
49 \edef\mathover{\string\mathover}
50 \edef\mathunder{\string\mathunder}
51 \edef\mathbotaccent{\string\mathbotaccent}
52 \edef\mathaccentwide{\string\mathaccentwide}
53 \edef\mathbotaccentwide{\string\mathbotaccentwide}

```

`\@@_set_big_operator:nnn` #1 : Symbol font name

#2 : Macro to assign

#3 : Glyph slot

In the examples following, say we're defining for the symbol `\sum` ( $\Sigma$ ). In order for literal Unicode characters to be used in the source and still have the correct limits behaviour, big operators are made math-active. This involves three steps:

- The active math char is defined to expand to the macro `\sum_sym`. (Later, the control sequence `\sum` will be assigned the math char.)
- Declare the plain old `mathchardef` for the control sequence `\sumop`. (This follows the convention of  $\text{\LaTeX}/\text{amsmath}$ .)
- Define `\sum_sym` as `\sumop`, followed by `\nolimits` if necessary.

Whether the `\nolimits` suffix is inserted is controlled by the token list `\l_@@_nolimits_tl`, which contains a list of such characters. This list is checked dynamically to allow it to be updated mid-document.

Examples of expansion, by default, for two big operators:

$(\sum \rightarrow) \Sigma \rightarrow \sum\_sym \rightarrow \sumop\nolimits$   
 $(\int \rightarrow) \int \rightarrow \int\_sym \rightarrow \intop$

```
54 \cs_new:Nn \@@_set_big_operator:nnn
55 {
56   \@@_char_gmake_mathactive:n {#3}
57   \cs_set_protected_nopar:Npx \@@_tmpa: { \exp_not:c { \cs_to_str:N #2 _sym } }
58   \char_gset_active_eq:nN {#3} \@@_tmpa:
59
60   \@@_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}
61
62   \cs_gset:cpx { \cs_to_str:N #2 _sym }
63   {
64     \exp_not:c { \cs_to_str:N #2 op }
65     \exp_not:n { \tl_if_in:NnT \l_@@_nolimits_tl {#2} \nolimits }
66   }
67 }
```

`\@@_set_math_open:nnn` #1 : Symbol font name

#2 : Macro to assign

#3 : Glyph slot

```
68 \cs_new:Nn \@@_set_math_open:nnn
69 {
70   \tl_if_in:NnTF \l_@@_radicals_tl {#2}
71   {
72     \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
73     { \@@_radical:nn {#1} {#3} }
74     \tl_set:cn {\l_@@_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #3}
75   }
76   {
77     \@@_set_delcode:nnn {#1} {#3} {#3}
```



```

78 \@@_set_mathcode:nnn {#3} \mathopen {#1}
79 \cs_gset_protected_nopar:Npx #2
80 { \@@_delimiter:Nnn \mathopen {#1} {#3} }
81 }
82 }

```

$\@@\_set\_math\_close:nnn$  #1 : Symbol font name  
 #2 : Macro to assign  
 #3 : Glyph slot

```

83 \cs_new:Nn \@@_set_math_close:nnn
84 {
85 \@@_set_delcode:nnn {#1} {#3} {#3}
86 \@@_set_mathcode:nnn {#3} \mathclose {#1}
87 \cs_gset_protected_nopar:Npx #2
88 { \@@_delimiter:Nnn \mathclose {#1} {#3} }
89 }

```

$\@@\_set\_math\_fence:nnnn$  #1 : Symbol font name  
 #2 : Macro to assign  
 #3 : Type, *e.g.*,  $\mathalpha$   
 #4 : Glyph slot

```

90 \cs_new:Nn \@@_set_math_fence:nnnn
91 {
92 \@@_set_mathcode:nnn {#4} {#3} {#1}
93 \@@_set_delcode:nnn {#1} {#4} {#4}
94 \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
95 { \@@_delimiter:Nnn \mathopen {#1} {#4} }
96 \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
97 { \@@_delimiter:Nnn \mathclose {#1} {#4} }
98 }

```

$\@@\_set\_math\_accent:Nnnn$  #1 : Accend command  
 #2 : Accent type (string)  
 #3 : Symbol font name  
 #4 : Glyph slot

```

99 \cs_new:Nn \@@_set_math_accent:Nnnn
100 {
101 \cs_gset_protected_nopar:Npx #1
102 { \@@_accent:nnn {#2} {#3} {#4} }
103 }

```

$\@@\_set\_math\_overunder:Nnnn$  #1 : Accend command  
 #2 : Accent type (string)  
 #3 : Symbol font name  
 #4 : Glyph slot

```

104 \cs_new:Nn \@@_set_math_overunder:Nnnn
105 {
106 \cs_gset_protected_nopar:Npx #1 ##1

```

```

107 {
108   \mathop
109   { \@@_accent:nnn {#2} {#3} {#4} {##1} }
110   \limits
111 }
112 }

```

## F.2 \setmathalphabet

\setmathalphabet

```

113 \keys_define:nn {@@_mathface}
114 {
115   version .code:n =
116   { \tl_set:Nn \l_@@_mversion_tl {#1} }
117 }
118
119 \DeclareDocumentCommand \setmathfontface { m O{} m O{} }
120 {
121   \tl_clear:N \l_@@_mversion_tl
122
123   \keys_set_known:nnN {@@_mathface} {#2,#4} \l_@@_keyval_clist
124   \exp_args:Nnx \fontspec_set_family:Nxn \l_@@_tmpa_tl
125   { ItalicFont={}, BoldFont={}, \exp_not:V \l_@@_keyval_clist } {#3}
126
127   \tl_if_empty:NT \l_@@_mversion_tl
128   {
129     \tl_set:Nn \l_@@_mversion_tl {normal}
130     \DeclareMathAlphabet #1 {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\mdde-
131       fault} {\updefault}
132   }
133   \SetMathAlphabet #1 {\l_@@_mversion_tl} {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\md-
134     default} {\updefault}
135
136   % integrate with fontspec's \setmathrm etc:
137   \tl_case:Nn #1
138   {
139     \mathrm { \cs_set_eq:NN \g__fontspec_mathrm_tl \l_@@_tmpa_tl }
140     \mathsf { \cs_set_eq:NN \g__fontspec_mathsf_tl \l_@@_tmpa_tl }
141     \mathtt { \cs_set_eq:NN \g__fontspec_mathtt_tl \l_@@_tmpa_tl }
142   }
143
144   \onlypreamble \setmathfontface

```

Note that L<sup>A</sup>T<sub>E</sub>X's SetMathAlphabet simply doesn't work to "reset" a maths alphabet font after `\begin{document}`, so unlike most of the other maths commands around we still restrict this one to the preamble.

\setoperatorfont TODO: add check?

```

144 \DeclareDocumentCommand \setoperatorfont {m}
145 { \tl_set:Nn \g_@@_operator_mathfont_tl {#1} }
146 \setoperatorfont{\mathrm}

```

### F.3 Hooks into fontspec

Historically, `\mathrm` and so on were completely overwritten by `unicode-math`, and `fontspec`'s methods for setting these fonts in the classical manner were bypassed.

While we could now re-activate the way that `fontspec` does the following, because we can now change maths fonts whenever it's better to define new commands in `unicode-math` to define the `\mathXYZ` fonts.

#### F.3.1 Text font

```

147 \cs_generate_variant:Nn \tl_if_eq:nnT {o}
148 \cs_set:Nn \__fontspec_setmainfont_hook:nn
149 {
150   \tl_if_eq:onT {\g__fontspec_mathrm_tl} {\rmdefault}
151   {
152     (XE) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
153     (LU) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
154     \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
155     \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
156     \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
157   }
158 }
159
160 \cs_set:Nn \__fontspec_setsansfont_hook:nn
161 {
162   \tl_if_eq:onT {\g__fontspec_mathsf_tl} {\sfdefault}
163   {
164     (XE) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
165     (LU) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}
166     \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
167     \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
168   }
169 }
170
171 \cs_set:Nn \__fontspec_setmonofont_hook:nn
172 {
173   \tl_if_eq:onT {\g__fontspec_mathtt_tl} {\ttdefault}
174   {
175     (XE) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {#1} {#2}
176     (LU) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {Renderer=Basic,#1} {#2}
177     \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\mddefault\updefault
178     \SetMathAlphabet\mathtt{bold} \g_fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
179   }
180 }

```

### F.3.2 *Maths font*

If the maths fonts are set explicitly, then the text commands above will not execute their branches to set the maths font alphabets.

```

181 \cs_set:Nn \__fontspec_setmathrm_hook:nn
182 {
183   \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
184   \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
185   \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
186 }
187 \cs_set:Nn \__fontspec_setboldmathrm_hook:nn
188 {
189   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\updefault
190   \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\bfdefault\updefault
191   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\itdefault
192 }
193 \cs_set:Nn \__fontspec_setmathsf_hook:nn
194 {
195   \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
196   \SetMathAlphabet\mathsf{bold}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
197 }
198 \cs_set:Nn \__fontspec_setmathtt_hook:nn
199 {
200   \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\mddefault\updefault
201   \SetMathAlphabet\mathtt{bold}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
202 }

```

### F.4 *The main \setmathfont macro*

Using a range including large character sets such as `\mathrel`, `\mathalpha`, *etc.*, is *very slow*! I hope to improve the performance somehow.

`\setmathfont` [#1]: font features (first optional argument retained for backwards compatibility)  
 #2 : font name  
 [#3]: font features

```

203 \DeclareDocumentCommand \setmathfont { O{} m O{} }
204 {
205   \@@_setmathfont:nn {#1,#3} {#2}
206 }
207 \cs_set:Nn \@@_setmathfont:nn
208 {
209   \tl_set:Nn \l_@@_fontname_tl {#2}
210   \@@_init:

```

Grab the current size information: (is this robust enough? Maybe it should be preceded by `\normalsize`). The macro `\S@<size>` contains the definitions of the sizes used for maths letters, subscripts and subsubscripts in `\tf@size`, `\sf@size`, and `\ssf@size`, respectively.

```

211 \cs_if_exist:cF { S@ \f@size } { \calculate@math@sizes }

```

```
212 \csname S@f@size\endcsname
```

Parse options and tell people what's going on:

```
213 \keys_set_known:nnN {unicode-math} {#1} \l_@@_unknown_keys_clist
214 \bool_if:NT \l_@@_init_bool { \@@_log:n {default-math-font} }
```

Use fontspec to select a font to use. After loading the font, we detect what sizes it recommends for scriptsize and scriptscriptsize, so after setting those values appropriately, we reload the font to take these into account.

```
215
216 <debug> \csname TIC\endcsname
217 \@@_fontspec_select_font:
218 <debug> \csname TOC\endcsname
219 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
220 {
221   \@@_declare_math_sizes:
222   \@@_fontspec_select_font:
223 }
```

Now define \@@\_symfont\_tl as the L<sup>A</sup>T<sub>E</sub>X math font to access everything:

```
224 \cs_if_exist:cF { sym \@@_symfont_tl }
225 {
226   \DeclareSymbolFont{\@@_symfont_tl}
227   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
228 }
229 \SetSymbolFont{\@@_symfont_tl}{\l_@@_mversion_tl}
230 {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
```

Set the bold math version.

```
231 \tl_set:Nn \l_@@_tmpa_tl {normal}
232 \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
233 {
234   \SetSymbolFont{\@@_symfont_tl}{bold}
235   {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
236 }
```

Declare the math sizes (i.e., scaling of superscripts) for the specific values for this font, and set defaults for math fams two and three for legacy compatibility:

```
237 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
238 {
239   \bool_set_true:N \g_@@_mainfont_already_set_bool
240   \@@_setup_legacy_fam_two:
241   \@@_setup_legacy_fam_three:
242 }
```

And now we input every single maths char.

```
243 <debug> \csname TIC\endcsname
244 \@@_input_math_symbol_table:
245 <debug> \csname TOC\endcsname
```

Finally,

- Remap symbols that don't take their natural mathcode

- Activate any symbols that need to be math-active
- Enable wide/narrow accents
- Assign delimiter codes for symbols that need to grow
- Setup the maths alphabets (`\mathbf` etc.)

```

246 \@@_remap_symbols:
247 \@@_setup_mathactives:
248 \@@_setup_delcodes:
249 <debug> \csname TIC\endcsname
250 \@@_setup_alphabets:
251 <debug> \csname TOC\endcsname
252 \@@_setup_negations:

```

Prevent spaces, and that's it:

```

253 \ignorespaces
254 }

```

Backward compatibility alias.

```

255 \cs_set_eq:NN \resetmathfont \setmathfont

```

`\@@_init:`

```

256 \cs_new:Nn \@@_init:
257 {

```

- Initially assume we're using a proper OpenType font with unicode maths.

```

258 \bool_set_true:N \l_@@_ot_math_bool

```

- Erase any conception  $\text{\LaTeX}$  has of previously defined math symbol fonts; this allows `\DeclareSymbolFont` at any point in the document.

```

259 \cs_set_eq:NN \glb@currsiz \scan_stop:

```

- To start with, assume we're defining the font for every math symbol character.

```

260 \bool_set_true:N \l_@@_init_bool
261 \seq_clear:N \l_@@_char_range_seq
262 \clist_clear:N \l_@@_char_nrange_clist
263 \seq_clear:N \l_@@_mathalph_seq
264 \seq_clear:N \l_@@_missing_alph_seq

```

- By default use the 'normal' math version.

```

265 \tl_set:Nn \l_@@_mversion_tl {normal}

```

- Other range initialisations.

```

266 \tl_set:Nn \l_@@_symfont_tl {operators}
267 \cs_set_eq:NN \l_@@_sym:nnn \l_@@_process_symbol_noparse:nnn
268 \cs_set_eq:NN \l_@@_set_mathalphabet_char:nnn \l_@@_mathmap_noparse:nnn
269 \cs_set_eq:NN \l_@@_remap_symbol:nnn \l_@@_remap_symbol_noparse:nnn
270 \cs_set_eq:NN \l_@@_maybe_init_alphabet:n \l_@@_init_alphabet:n
271 \cs_set_eq:NN \l_@@_map_char_single:nn \l_@@_map_char_noparse:nn
272 \cs_set_eq:NN \l_@@_assign_delcode:nn \l_@@_assign_delcode_noparse:nn
273 \cs_set_eq:NN \l_@@_make_mathactive:nNN \l_@@_make_mathactive_noparse:nNN

```

- Define default font features for the script and scriptscript font.

```

274 \tl_set:Nn \l_@@_script_features_tl {Style=MathScript}
275 \tl_set:Nn \l_@@_sscript_features_tl {Style=MathScriptScript}
276 \tl_set_eq:NN \l_@@_script_font_tl \l_@@_fontname_tl
277 \tl_set_eq:NN \l_@@_sscript_font_tl \l_@@_fontname_tl
278 }

```

`\l_@@_declare_math_sizes:` Set the math sizes according to the recommended font parameters:

```

279 \cs_new:Nn \l_@@_declare_math_sizes:
280 {
281   < *LU >
282   \fp_compare:nF { \l_@@_script_style_size:n {ScriptPercentScaleDown} == 0 }
283   {
284     \DeclareMathSizes { \f@size } { \f@size }
285       { \l_@@_script_style_size:n {ScriptPercentScaleDown} }
286       { \l_@@_script_style_size:n {ScriptScriptPercentScaleDown} }
287   }
288   < /LU >
289   < *XE >
290   \dim_compare:nF { \fontdimen 10 \l_@@_font == 0pt }
291   {
292     \DeclareMathSizes { \f@size } { \f@size }
293       { \l_@@_fontdimen_to_scale:nn {10} { \l_@@_font } }
294       { \l_@@_fontdimen_to_scale:nn {11} { \l_@@_font } }
295   }
296   < /XE >
297   }
298   < *LU >

```

`\l_@@_script_style_size:n` Determine script- and scriptscriptstyle sizes using luaotfload:

```

298 \cs_new:Nn \l_@@_script_style_size:n
299 {
300   \fp_eval:n { \directlua{tex.sprint(luaotfload.aux.get_math_dimension("l_@@_font", "#1"))} * \f@size
301   }
302   % \end{macrocode}
303   % \end{macro}

```

```

304 </LU>
305 %
306 %
307 % \begin{macro}{\@@_setup_legacy_fam_two:}
308 % \TeX\ won't load the same font twice at the same scale, so we need to mag-
nify this one by an imperceptable amount.
309 % \begin{macrocode}
310 \cs_new:Nn \@@_setup_legacy_fam_two:
311 {
312   \fontspec_set_family:Nxn \l_@@_family_tl
313   {
314     \l_@@_font_keyval_tl,
315     Scale=1.00001,
316     FontAdjustment =
317     {
318       \fontdimen8\font= \@@_get_fontparam:nn {43} {FractionNumeratorDis-
playStyleShiftUp}\relax
319       \fontdimen9\font= \@@_get_fontparam:nn {42} {FractionNumerator-
ShiftUp}\relax
320       \fontdimen10\font=\@@_get_fontparam:nn {32} {StackTopShiftUp}\relax
321       \fontdimen11\font=\@@_get_fontparam:nn {45} {FractionDenominatorDis-
playStyleShiftDown}\relax
322       \fontdimen12\font=\@@_get_fontparam:nn {44} {FractionDenominatorShift-
Down}\relax
323       \fontdimen13\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
324       \fontdimen14\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
325       \fontdimen15\font=\@@_get_fontparam:nn {22} {SuperscriptShif-
tUpCramped}\relax
326       \fontdimen16\font=\@@_get_fontparam:nn {18} {SubscriptShiftDown}\relax
327       \fontdimen17\font=\@@_get_fontparam:nn {18} {SubscriptShiftDownWith-
Superscript}\relax
328       \fontdimen18\font=\@@_get_fontparam:nn {24} {SuperscriptBaselineDrop-
Max}\relax
329       \fontdimen19\font=\@@_get_fontparam:nn {20} {SubscriptBaselineDrop-
Min}\relax
330       \fontdimen20\font=0pt\relax % delim1 = FractionDelimiterDisplaySize
331       \fontdimen21\font=0pt\relax % delim2 = FractionDelimiterSize
332       \fontdimen22\font=\@@_get_fontparam:nn {15} {AxisHeight}\relax
333     }
334   } {\l_@@_fontname_tl}
335   \SetSymbolFont{symbols}{\l_@@_mversion_tl}
336   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
337
338   \tl_set:Nn \l_@@_tmpa_tl {normal}
339   \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
340   {
341     \SetSymbolFont{symbols}{bold}
342     {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
343   }

```



344 }

\@@\_setup\_legacy\_fam\_three: Similarly, this font is shrunk by an imperceptable amount for T<sub>E</sub>X to load it again.

```

345 \cs_new:Nn \@@_setup_legacy_fam_three:
346 {
347   \fontspec_set_family:Nxn \l_@@_family_tl
348   {
349     \l_@@_font_keyval_tl,
350     Scale=0.99999,
351     FontAdjustment={
352       \fontdimen8\font= \@@_get_fontparam:nn {48} {FractionRuleThick-
ness}\relax
353       \fontdimen9\font= \@@_get_fontparam:nn {28} {UpperLimitGapMin}\relax
354       \fontdimen10\font=\@@_get_fontparam:nn {30} {LowerLimitGapMin}\relax
355       \fontdimen11\font=\@@_get_fontparam:nn {29} {UpperLimitBaselineR-
iseMin}\relax
356       \fontdimen12\font=\@@_get_fontparam:nn {31} {LowerLimitBaselineDrop-
Min}\relax
357       \fontdimen13\font=0pt\relax
358     }
359   } {\l_@@_fontname_tl}
360   \SetSymbolFont{largesymbols}{\l_@@_mversion_tl}
361   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
362
363   \tl_set:Nn \l_@@_tmpa_tl {normal}
364   \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
365   {
366     \SetSymbolFont{largesymbols}{bold}
367     {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
368   }
369 }

370 \cs_new:Nn \@@_get_fontparam:nn
371 (XE) { \the\fontdimen#1\l_@@_font\relax }
372 (LU) { \directlua{fontspec.mathfontdimen("l_@@_font", "#2")}}

```

\@@\_fontspec\_select\_font: Select the font with \fontspec and define \l\_@@\_font from it.

```

373 \cs_new:Nn \@@_fontspec_select_font:
374 {
375   \tl_set:Nx \l_@@_font_keyval_tl {
376     (LU) Renderer = Basic,
377     BoldItalicFont = {}, ItalicFont = {},
378     Script = Math,
379     SizeFeatures =
380     {
381       {
382         Size = \tf@size-
383       } ,
384       {
385         Size = \sf@size-\tf@size ,

```

```

386     Font = \l_@@_script_font_tl ,
387     \l_@@_script_features_tl
388   } ,
389   {
390     Size = -\sf@size ,
391     Font = \l_@@_sscript_font_tl ,
392     \l_@@_sscript_features_tl
393   }
394 } ,
395 \l_@@_unknown_keys_clist
396 }
397 \fontspec_set_fontface:NNxn \l_@@_font \l_@@_family_tl
398 {\l_@@_font_keyval_tl} {\l_@@_fontname_tl}

```

Check whether we're using a real maths font:

```

399 \group_begin:
400   \fontfamily{\l_@@_family_tl}\selectfont
401   \fontspec_if_script:nF {math} {\bool_gset_false:N \l_@@_ot_math_bool}
402 \group_end:
403 }

```

#### F.4.1 Functions for setting up symbols with mathcodes

`\@@_process_symbol_noparse:nnn` If the range font feature has been used, then only a subset of the Unicode glyphs are to be defined. See section §G.3 for the code that enables this.

```

404 \cs_set:Nn \@@_process_symbol_noparse:nnn
405 {
406   \@@_set_mathsymbol:nNNn {\@@_symfont_tl} #2 #3 {#1}
407 }
408 \cs_set:Nn \@@_process_symbol_parse:nnn
409 {
410   \@@_if_char_spec:nNNT {#1} {#2} {#3}
411   {
412     \@@_process_symbol_noparse:nnn {#1} {#2} {#3}
413   }
414 }

```

`\@@_remap_symbols:` This function is used to define the mathcodes for those chars which should be mapped to a different glyph than themselves.

```

\@@_remap_symbol_noparse:nnn
\@@_remap_symbol_parse:nnn
415 \cs_new:Npn \@@_remap_symbols:
416 {
417   \@@_remap_symbol:nnn{'-}{\mathbin}{"02212}% hyphen to minus
418   \@@_remap_symbol:nnn{'*}{\mathbin}{"02217}% text asterisk to "centred as-
      terisk"
419   \bool_if:NF \g_@@_literal_colon_bool
420   {
421     \@@_remap_symbol:nnn{'\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
422   }
423 }

```

Where `\@@_remap_symbol:nnn` is defined to be one of these two, depending on the range setup:

```

424 \cs_new:Nn \@@_remap_symbol_parse:nnn
425 {
426   \@@_if_char_spec:nNNT {#3} {\@nil} {#2}
427   { \@@_remap_symbol_noparse:nnn {#1} {#2} {#3} }
428 }
429 \cs_new:Nn \@@_remap_symbol_noparse:nnn
430 {
431   \clist_map_inline:nn {#1}
432   { \@@_set_mathcode:nnnn {##1} {#2} {\@@_symfont_tl} {#3} }
433 }

```

#### F.4.2 Active math characters

There are more math active chars later in the subscript/superscript section. But they don't need to be able to be typeset directly.

`\@@_setup_mathactives:`

```

434 \cs_new:Npn \@@_setup_mathactives:
435 {
436   \@@_make_mathactive:nNN {"2032} \@@_prime_single_mchar \mathord
437   \@@_make_mathactive:nNN {"2033} \@@_prime_double_mchar \mathord
438   \@@_make_mathactive:nNN {"2034} \@@_prime_triple_mchar \mathord
439   \@@_make_mathactive:nNN {"2057} \@@_prime_quad_mchar \mathord
440   \@@_make_mathactive:nNN {"2035} \@@_backprime_single_mchar \mathord
441   \@@_make_mathactive:nNN {"2036} \@@_backprime_double_mchar \mathord
442   \@@_make_mathactive:nNN {"2037} \@@_backprime_triple_mchar \mathord
443   \@@_make_mathactive:nNN {'\'} \mathstraightquote \mathord
444   \@@_make_mathactive:nNN {'\'} \mathbacktick \mathord
445 }

```

`\@@_make_mathactive:nNN` Makes #1 a mathactive char, and gives cs #2 the meaning of mathchar #1 with class #3. You are responsible for giving active #1 a particular meaning!

```

446 \cs_new:Nn \@@_make_mathactive_parse:nNN
447 {
448   \@@_if_char_spec:nNNT {#1} #2 #3
449   { \@@_make_mathactive_noparse:nNN {#1} #2 #3 }
450 }
451 \cs_new:Nn \@@_make_mathactive_noparse:nNN
452 {
453   \@@_set_mathchar:NNnn #2 #3 {\@@_symfont_tl} {#1}
454   \@@_char_gmake_mathactive:n {#1}
455 }

```

#### F.4.3 Delimiter codes

`\@@_assign_delcode:nn`

```

456 \cs_new:Nn \@@_assign_delcode_noparse:nn

```

```

457 {
458   \@@_set_delcode:nnn \@@_symfont_tl {#1} {#2}
459 }
460 \cs_new:Nn \@@_assign_delcode_parse:nn
461 {
462   \@@_if_char_spec:nNNT {#2} {\@nil} {\@nil}
463   {
464     \@@_assign_delcode_noparse:nn {#1} {#2}
465   }
466 }

```

\@@\_assign\_delcode:n Shorthand.

```

467 \cs_new:Nn \@@_assign_delcode:n { \@@_assign_delcode:nn {#1} {#1} }

```

\@@\_setup\_delcodes: Some symbols that aren't mathopen/mathclose still need to have delimiter codes assigned. The list of vertical arrows may be incomplete. On the other hand, many fonts won't support them all being stretchy. And some of them are probably not meant to stretch, either. But adding them here doesn't hurt.

```

468 \cs_new:Npn \@@_setup_delcodes:
469 {
470   % ensure \left. and \right. work:
471   \@@_set_delcode:nnn \@@_symfont_tl {'\.'} {\c_zero}
472   % this is forcefully done to fix a bug -- indicates a larger problem!
473
474   \@@_assign_delcode:nn {'\'} {\g_@@_slash_delimiter_usv}
475   \@@_assign_delcode:nn {"2044} {\g_@@_slash_delimiter_usv} % fracslash
476   \@@_assign_delcode:nn {"2215} {\g_@@_slash_delimiter_usv} % divslash
477   \@@_assign_delcode:n {"005C} % backslash
478   \@@_assign_delcode:nn {'\<} {"27E8} % angle brackets with ascii notation
479   \@@_assign_delcode:nn {'\>} {"27E9} % angle brackets with ascii notation
480   \@@_assign_delcode:n {"2191} % up arrow
481   \@@_assign_delcode:n {"2193} % down arrow
482   \@@_assign_delcode:n {"2195} % updown arrow
483   \@@_assign_delcode:n {"219F} % up arrow twohead
484   \@@_assign_delcode:n {"21A1} % down arrow twohead
485   \@@_assign_delcode:n {"21A5} % up arrow from bar
486   \@@_assign_delcode:n {"21A7} % down arrow from bar
487   \@@_assign_delcode:n {"21A8} % updown arrow from bar
488   \@@_assign_delcode:n {"21BE} % up harpoon right
489   \@@_assign_delcode:n {"21BF} % up harpoon left
490   \@@_assign_delcode:n {"21C2} % down harpoon right
491   \@@_assign_delcode:n {"21C3} % down harpoon left
492   \@@_assign_delcode:n {"21C5} % arrows up down
493   \@@_assign_delcode:n {"21F5} % arrows down up
494   \@@_assign_delcode:n {"21C8} % arrows up up
495   \@@_assign_delcode:n {"21CA} % arrows down down
496   \@@_assign_delcode:n {"21D1} % double up arrow
497   \@@_assign_delcode:n {"21D3} % double down arrow
498   \@@_assign_delcode:n {"21D5} % double updown arrow

```

```

499 \@@_assign_delcode:n {"21DE} % up arrow double stroke
500 \@@_assign_delcode:n {"21DF} % down arrow double stroke
501 \@@_assign_delcode:n {"21E1} % up arrow dashed
502 \@@_assign_delcode:n {"21E3} % down arrow dashed
503 \@@_assign_delcode:n {"21E7} % up white arrow
504 \@@_assign_delcode:n {"21E9} % down white arrow
505 \@@_assign_delcode:n {"21EA} % up white arrow from bar
506 \@@_assign_delcode:n {"21F3} % updown white arrow
507 }

```

## F.5 (Big) operators

Turns out that  $\text{X}\text{\TeX}$  is clever enough to deal with big operators for us automatically with `\Umathchardef`. Amazing!

However, the limits aren't set automatically; that is, we want to define, a la Plain  $\text{T}\text{E}\text{X}$  *etc.*, `\def\int{\intop\nolimits}`, so there needs to be a transformation from `\int` to `\intop` during the expansion of `\_@@_sym:nnn` in the appropriate contexts.

`\l_@@_nolimits_tl` This macro is a sequence containing those maths operators that require a `\nolimits` suffix. This list is used when processing `unicode-math-table.tex` to define such commands automatically (see the macro `\@@_set_mathsymbol:nNNn`). I've chosen essentially just the operators that look like integrals; hopefully a better mathematician can help me out here. I've a feeling that it's more useful *not* to include the multiple integrals such as `\iiint`, but that might be a matter of preference.

```

508 \tl_new:N \l_@@_nolimits_tl
509 \tl_set:Nn \l_@@_nolimits_tl
510 {
511   \int\iint\iiint\iiint\oint\oiint\oiint
512   \intclockwise\varointclockwise\ointctrclockwise\sumint
513   \intbar\intBar\oint\cirfnint\awint\rrpint
514   \scpolint\ncpolint\pointint\sqint\intlarhk\intx
515   \intcap\intcup\upint\lowint
516 }

```

`\addnolimits` This macro appends material to the macro containing the list of operators that don't take limits.

```

517 \DeclareDocumentCommand \addnolimits {m}
518 {
519   \tl_put_right:Nn \l_@@_nolimits_tl {#1}
520 }

```

`\removenolimits` Can this macro be given a better name? It removes an item from the `nolimits` list.

```

521 \DeclareDocumentCommand \removenolimits {m}
522 {
523   \tl_remove_all:Nn \l_@@_nolimits_tl {#1}
524 }

```

## F.6 Radicals

The radical for square root is organised in `\@@_set_mathsymbol:nNNn`. I think it's the only radical ever. (Actually, there is also `\cuberoot` and `\fourthroot`, but they don't seem to behave as proper radicals.)

Also, what about right-to-left square roots?

`\l_@@_radicals_tl` We organise radicals in the same way as `nolimits`-operators.

```
525 \tl_new:N \l_@@_radicals_tl
526 \tl_set:Nn \l_@@_radicals_tl {\sqrt \longdivision}
```

## F.7 Maths accents

Maths accents should just work *if they are available in the font*.

## F.8 Common interface for font parameters

$\text{\LaTeX}$  and  $\text{\LuaTeX}$  have different interfaces for math font parameters. We use  $\text{\LuaTeX}$ 's interface because it's much better, but rename the primitives to be more  $\text{\LaTeX}$ -like. There are getter and setter commands for each font parameter. The names of the parameters is derived from the  $\text{\LuaTeX}$  names, with underscores inserted between words. For every parameter `\Umath<LuaTeX name>`, we define an expandable getter command `\@@_<LaTeX3 name>:N` and a protected setter command `\@@_set_<LaTeX3 name>:Nn`. The getter command takes one of the style primitives (`\displaystyle` etc.) and expands to the font parameter, which is a *<dimension>*. The setter command takes a style primitive and a dimension expression, which is parsed with `\dim_eval:n`.

Often, the mapping between font dimensions and font parameters is bijective, but there are cases which require special attention:

- Some parameters map to different dimensions in display and non-display styles.
- Likewise, one parameter maps to different dimensions in non-cramped and cramped styles.
- There are a few parameters for which  $\text{\LaTeX}$  doesn't seem to provide `\font-dimens`; in this case the getter and setter commands are left undefined.

*Cramped style tokens*  $\text{\LuaTeX}$  has `\crampeddisplaystyle` etc., but they are loaded as `\luatexcrampeddisplaystyle` etc. by the `luatextra` package.  $\text{\LaTeX}$ , however, doesn't have these primitives, and their syntax cannot really be emulated. Nevertheless, we define these commands as quarks, so they can be used as arguments to the font parameter commands (but nowhere else). Making these commands available is necessary because we need to make a distinction between cramped and non-cramped styles for one font parameter.

`\@@_new_cramped_style:N` #1 : command

Define  $\langle command \rangle$  as a new cramped style switch. For Lua $\TeX$ , simply rename the corresponding primitive if it is not already defined. For Xe $\TeX$ , define  $\langle command \rangle$  as a new quark.

```

527 \cs_new_protected_nopar:Nn \@@_new_cramped_style:N
528 (XE) { \quark_new:N #1 }
529 (LU) {
530 (LU)   \cs_if_exist:NF #1
531 (LU)     { \cs_new_eq:Nc #1 { luatex \cs_to_str:N #1 } }
532 (LU) }

```

`\crampeddisplaystyle` The cramped style commands.

`\crampedtextstyle` 533 `\@@_new_cramped_style:N \crampeddisplaystyle`

`\crampedscriptstyle` 534 `\@@_new_cramped_style:N \crampedtextstyle`

`\crampedscriptscriptstyle` 535 `\@@_new_cramped_style:N \crampedscriptstyle`

536 `\@@_new_cramped_style:N \crampedscriptscriptstyle`

*Font dimension mapping* Font parameters may differ between the styles. Lua $\TeX$  accounts for this by having the parameter primitives take a style token argument. To replicate this behavior in Xe $\TeX$ , we have to map style tokens to specific combinations of font dimension numbers and math fonts (`\textfont` etc.).

`\@@_font_dimen:Nnnnn` #1 : style token

#2 : font dimen for display style

#3 : font dimen for cramped display style

#4 : font dimen for non-display styles

#5 : font dimen for cramped non-display styles

Map math style to Xe $\TeX$  math font dimension.  $\langle style token \rangle$  must be one of the style switches (`\displaystyle`, `\crampeddisplaystyle`, ...). The other parameters are integer constants referring to font dimension numbers. The macro expands to a dimension which contains the appropriate font dimension.

```

537 (*XE)
538 \cs_new_nopar:Npn \@@_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
539   \fontdimen
540   \cs_if_eq:NNTF #1 \displaystyle {
541     #2 \textfont
542   } {
543     \cs_if_eq:NNTF #1 \crampeddisplaystyle {
544       #3 \textfont
545     } {
546       \cs_if_eq:NNTF #1 \textstyle {
547         #4 \textfont
548       } {
549         \cs_if_eq:NNTF #1 \crampedtextstyle {
550           #5 \textfont
551         } {
552           \cs_if_eq:NNTF #1 \scriptstyle {

```

```

553         #4 \scriptfont
554     } {
555         \cs_if_eq:NNTF #1 \crampedscriptstyle {
556             #5 \scriptfont
557         } {
558             \cs_if_eq:NNTF #1 \scriptscriptstyle {
559                 #4 \scriptscriptfont
560             } {

```

Should we check here if the style is invalid?

```

561             #5 \scriptscriptfont
562         }
563     }
564 }
565 }
566 }
567 }
568 }

```

Which family to use?

```

569     \c_two
570 }
571 </XE>

```

*Font parameters* This paragraph contains macros for defining the font parameter interface, as well as the definition for all font parameters known to Lua<sub>T</sub><sub>E</sub>X.

```

\@@_font_param:nnnnn #1 : name
#2 : font dimension for non-cramped display style
#3 : font dimension for cramped display style
#4 : font dimension for non-cramped non-display styles
#5 : font dimension for cramped non-display styles

```

This macro defines getter and setter functions for the font parameter  $\langle name \rangle$ . The Lua<sub>T</sub><sub>E</sub>X font parameter name is produced by removing all underscores and prefixing the result with Umath. The X<sub>Y</sub><sub>T</sub><sub>E</sub>X font dimension numbers must be integer constants.

```

572 \cs_new_protected_nopar:Nn \@@_font_param:nnnnn
573 < *XE>
574 {
575     \@@_font_param_aux:ccnnnn { @@_ #1 :N } { @@_set_ #1 :Nn }
576     { #2 } { #3 } { #4 } { #5 }
577 }
578 </XE>
579 < *LU>
580 {
581     \tl_set:Nn \l_@@_tmpa_tl { #1 }
582     \tl_remove_all:Nn \l_@@_tmpa_tl { _ }
583     \@@_font_param_aux:ccc { @@_ #1 :N } { @@_set_ #1 :Nn }
584     { Umath \l_@@_tmpa_tl }

```



```

585 }
586 </LU>

\@@_font_param:nnn #1 : name
#2 : font dimension for display style
#3 : font dimension for non-display styles
This macro defines getter and setter functions for the font parameter  $\langle name \rangle$ . The
LuaTeX font parameter name is produced by removing all underscores and pre-
fixing the result with Umath. The XeTeX font dimension numbers must be integer
constants.

587 \cs_new_protected_nopar:Nn \@@_font_param:nnn
588 {
589   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #3 } { #3 }
590 }

\@@_font_param:nn #1 : name
#2 : font dimension
This macro defines getter and setter functions for the font parameter  $\langle name \rangle$ . The
LuaTeX font parameter name is produced by removing all underscores and pre-
fixing the result with Umath. The XeTeX font dimension number must be an integer
constant.

591 \cs_new_protected_nopar:Nn \@@_font_param:nn
592 {
593   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #2 } { #2 }
594 }

\@@_font_param:n #1 : name
This macro defines getter and setter functions for the font parameter  $\langle name \rangle$ ,
which is considered unavailable in XeTeX. The LuaTeX font parameter name is
produced by removing all underscores and prefixing the result with Umath.

595 \cs_new_protected_nopar:Nn \@@_font_param:n
596 <XE> { }
597 <LU> { \@@_font_param:nnnnn { #1 } { 0 } { 0 } { 0 } { 0 } }

\@@_font_param_aux:NNnnnn Auxiliary macros for generating font parameter accessor macros.
\@@_font_param_aux:NNN
598 <*XE>
599 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNnnnn
600 {
601   \cs_new_nopar:Npn #1 ##1
602   {
603     \@@_font_dimen:Nnnnn ##1 { #3 } { #4 } { #5 } { #6 }
604   }
605   \cs_new_protected_nopar:Npn #2 ##1 ##2
606   {
607     #1 ##1 \dim_eval:n { ##2 }
608   }
609 }
610 \cs_generate_variant:Nn \@@_font_param_aux:NNnnnn { cc }

```

```

611 </XE>
612 <*LU>
613 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNN
614 {
615   \cs_new_nopar:Npn #1 ##1
616   {
617     #3 ##1
618   }
619   \cs_new_protected_nopar:Npn #2 ##1 ##2
620   {
621     #3 ##1 \dim_eval:n { ##2 }
622   }
623 }
624 \cs_generate_variant:Nn \@@_font_param_aux:NNN { ccc }
625 </LU>

```

Now all font parameters that are listed in the LuaTeX reference follow.

```

626 \@@_font_param:nn { axis } { 15 }
627 \@@_font_param:nn { operator_size } { 13 }
628 \@@_font_param:n { fraction_del_size }
629 \@@_font_param:nnn { fraction_denom_down } { 45 } { 44 }
630 \@@_font_param:nnn { fraction_denom_vgap } { 50 } { 49 }
631 \@@_font_param:nnn { fraction_num_up } { 43 } { 42 }
632 \@@_font_param:nnn { fraction_num_vgap } { 47 } { 46 }
633 \@@_font_param:nn { fraction_rule } { 48 }
634 \@@_font_param:nn { limit_above_bgap } { 29 }
635 \@@_font_param:n { limit_above_kern }
636 \@@_font_param:nn { limit_above_vgap } { 28 }
637 \@@_font_param:nn { limit_below_bgap } { 31 }
638 \@@_font_param:n { limit_below_kern }
639 \@@_font_param:nn { limit_below_vgap } { 30 }
640 \@@_font_param:nn { over_delimiter_vgap } { 41 }
641 \@@_font_param:nn { over_delimiter_bgap } { 38 }
642 \@@_font_param:nn { under_delimiter_vgap } { 40 }
643 \@@_font_param:nn { under_delimiter_bgap } { 39 }
644 \@@_font_param:nn { overbar_kern } { 55 }
645 \@@_font_param:nn { overbar_rule } { 54 }
646 \@@_font_param:nn { overbar_vgap } { 53 }
647 \@@_font_param:n { quad }
648 \@@_font_param:nn { radical_kern } { 62 }
649 \@@_font_param:nn { radical_rule } { 61 }
650 \@@_font_param:nnn { radical_vgap } { 60 } { 59 }
651 \@@_font_param:nn { radical_degree_before } { 63 }
652 \@@_font_param:nn { radical_degree_after } { 64 }
653 \@@_font_param:nn { radical_degree_raise } { 65 }
654 \@@_font_param:nn { space_after_script } { 27 }
655 \@@_font_param:nnn { stack_denom_down } { 35 } { 34 }
656 \@@_font_param:nnn { stack_num_up } { 33 } { 32 }
657 \@@_font_param:nnn { stack_vgap } { 37 } { 36 }

```

```

658 \@@_font_param:nn { sub_shift_down } { 18 }
659 \@@_font_param:nn { sub_shift_drop } { 20 }
660 \@@_font_param:n { subsup_shift_down }
661 \@@_font_param:nn { sub_top_max } { 19 }
662 \@@_font_param:nn { subsup_vgap } { 25 }
663 \@@_font_param:nn { sup_bottom_min } { 23 }
664 \@@_font_param:nn { sup_shift_drop } { 24 }
665 \@@_font_param:nnnn { sup_shift_up } { 21 } { 22 } { 21 } { 22 }
666 \@@_font_param:nn { supsub_bottom_max } { 26 }
667 \@@_font_param:nn { underbar_kern } { 58 }
668 \@@_font_param:nn { underbar_rule } { 57 }
669 \@@_font_param:nn { underbar_vgap } { 56 }
670 \@@_font_param:n { connector_overlap_min }

```

## G Font features

### G.1 Math version

```

671 \keys_define:nn {unicode-math}
672 {
673   version .code:n =
674   {
675     \tl_set:Nn \l_@@_mversion_tl {#1}
676     \DeclareMathVersion {\l_@@_mversion_tl}
677   }
678 }

```

### G.2 Script and scriptscript font options

```

679 \keys_define:nn {unicode-math}
680 {
681   script-features .tl_set:N = \l_@@_script_features_tl ,
682   sscript-features .tl_set:N = \l_@@_sscript_features_tl ,
683   script-font .tl_set:N = \l_@@_script_font_tl ,
684   sscript-font .tl_set:N = \l_@@_sscript_font_tl ,
685 }

```

### G.3 Range processing

```

686 \keys_define:nn {unicode-math}
687 {
688   range .code:n =
689   {
690     \bool_set_false:N \l_@@_init_bool

```

Set processing functions if we're not defining the full Unicode math repertoire. Math symbols are defined with `\_@@_sym:nnn`; see section §F.4.1 for the individual definitions

```

691   \int_incr:N \g_@@_fam_int
692   \tl_set:Nx \@@_symfont_tl {@@_fam\int_use:N\g_@@_fam_int}

```

```

693 \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_parse:nnn
694 \cs_set_eq:NN \@@_set_mathalphabet_char:Nnn \@@_mathmap_parse:Nnn
695 \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_parse:nnn
696 \cs_set_eq:NN \@@_maybe_init_alphabet:n \use_none:n
697 \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_parse:nn
698 \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_parse:nn
699 \cs_set_eq:NN \@@_make_mathactive:nNN \@@_make_mathactive_parse:nNN

```

Proceed by filling up the various ‘range’ seqs according to the user options.

```

700 \seq_clear:N \l_@@_char_range_seq
701 \seq_clear:N \l_@@_mclass_range_seq
702 \seq_clear:N \l_@@_cmd_range_seq
703 \seq_clear:N \l_@@_mathalph_seq
704
705 \clist_map_inline:nn {#1}
706 {
707   \@@_if_mathalph_decl:nTF {##1}
708   {
709     \seq_put_right:Nx \l_@@_mathalph_seq
710     {
711       { \exp_not:V \l_@@_tmpa_tl }
712       { \exp_not:V \l_@@_tmpb_tl }
713       { \exp_not:V \l_@@_tmpc_tl }
714     }
715   }
716 {

```

Four cases: math class matching the known list; single item that is a control sequence—command name; single item that isn’t—edge case, must be 0–9; none of the above—char range.

```

717   \seq_if_in:NnTF \g_@@_mathclasses_seq {##1}
718   { \seq_put_right:Nn \l_@@_mclass_range_seq {##1} }
719   {
720     \bool_lazy_and:nnTF { \tl_if_single_p:n {##1} } { \to-
ken_if_cs_p:N #1 }
721     { \seq_put_right:Nn \l_@@_cmd_range_seq {##1} }
722     { \seq_put_right:Nn \l_@@_char_range_seq {##1} }
723   }
724 }
725 }
726 }
727 }

```

\@@\_if\_mathalph\_decl:nTF Possible forms of input:

```

\mathscr
\mathscr->\mathup
\mathscr/{Latin}
\mathscr/{Latin}->\mathup

```

Outputs:

tmpa: math style (e.g., \mathscr)

tmpb: alphabets (*e.g.*, Latin)

tmpc: remap style (*e.g.*,  $\mathup$ ). Defaults to tmpa.

The remap style can also be  $\mathcal{\rightarrow}\text{stixcal}$ , which I marginally prefer in the general case.

```
728 \prg_new_conditional:Nnn \@@_if_mathalph_decl:n {TF}
729 {
730   \tl_set:Nn \l_@@_tmpa_tl {#1}
731   \tl_clear:N \l_@@_tmpb_tl
732   \tl_clear:N \l_@@_tmpc_tl
733
734   \tl_if_in:NnT \l_@@_tmpa_tl {->}
735   { \exp_after:wN \@@_split_arrow:w \l_@@_tmpa_tl \q_nil }
736
737   \tl_if_in:NnT \l_@@_tmpa_tl {/}
738   { \exp_after:wN \@@_split_slash:w \l_@@_tmpa_tl \q_nil }
739
740   \tl_set:Nx \l_@@_tmpa_tl { \tl_to_str:N \l_@@_tmpa_tl }
741   \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \math }
742   \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \sym }
743   \tl_trim_spaces:N \l_@@_tmpa_tl
744
745   \tl_if_empty:NT \l_@@_tmpc_tl
746   { \tl_set_eq:NN \l_@@_tmpc_tl \l_@@_tmpa_tl }
747
748   \seq_if_in:NVTF \g_@@_named_ranges_seq \l_@@_tmpa_tl
749   { \prg_return_true: } { \prg_return_false: }
750 }
751 \cs_set:Npn \@@_split_arrow:w #1->#2 \q_nil
752 {
753   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
754   \tl_set:Nx \l_@@_tmpc_tl { \tl_trim_spaces:n {#2} }
755 }
756 \cs_set:Npn \@@_split_slash:w #1/#2 \q_nil
757 {
758   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
759   \tl_set:Nx \l_@@_tmpb_tl { \tl_trim_spaces:n {#2} }
760 }
```

Pretty basic comma separated range processing. Donald Arseneau's selectp package has a cleverer technique.

$\@@_if\_char\_spec:n\text{NNT}$  #1 : Unicode character slot  
#2 : control sequence (character macro)  
#3 : control sequence (math class)  
#4 : code to execute

This macro expands to #4 if any of its arguments are contained in  $\l\_@@\_char\_range\_seq$ . This list can contain either character ranges (for checking with #1) or

control sequences. These latter can either be the command name of a specific character, *or* the math type of one (e.g., `\mathbin`).

Character ranges are passed to `\@@_if_char_spec:nNNT`, which accepts input in the form shown in table 14.

Table 14: Ranges accepted by `\@@_if_char_spec:nNNT`.

Input	Range
$x$	$r = x$
$x-$	$r \geq x$
$-y$	$r \leq y$
$x-y$	$x \leq r \leq y$

We have three tests, performed sequentially in order of execution time. Any test finding a match jumps directly to the end.

```

761 \cs_new:Nn \@@_if_char_spec:nNNT
762 {
763   % math class:
764   \seq_if_in:NnT \l_@@_mclass_range_seq {#3}
765   { \use_none_delimit_by_q_nil:w }
766
767   % command name:
768   \seq_if_in:NnT \l_@@_cmd_range_seq {#2}
769   { \use_none_delimit_by_q_nil:w }
770
771   % character slot:
772   \seq_map_inline:Nn \l_@@_char_range_seq
773   {
774     \@@_int_if_slot_in_range:nnT {#1} {##1}
775     { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
776   }
777
778   % the following expands to nil if no match was found:
779   \use_none:nnn
780   \q_nil
781   \use:n
782   {
783     \clist_put_right:Nx \l_@@_char_nrange_clist { \int_eval:n {#1} }
784     #4
785   }
786 }
```

`\@@_int_if_slot_in_range:nnT` A ‘numrange’ is like -2, 5-8, 12, 17- (can be unsorted).

Four cases, four argument types:

```

% input    #2      #3      #4
% "1 "    [ 1] - [qn] - [  ] qs
% "1- "   [ 1] - [  ] - [qn-] qs
```

```

% " -3" [ ] - [ 3] - [qn-] qs
% "1-3" [ 1] - [ 3] - [qn-] qs

787 \cs_new:Nn \@@_int_if_slot_in_range:nnT
788 { \@@_numrange_parse:nwT {#1} #2 - \q_nil - \q_stop {#3} }
789 \cs_set:Npn \@@_numrange_parse:nwT #1 #2 - #3 - #4 \q_stop #5
790 {
791   \tl_if_empty:nTF {#4} { \int_compare:nT {#1=#2} {#5} }
792   {
793     \tl_if_empty:nTF {#3} { \int_compare:nT {#1>=#2} {#5} }
794     {
795       \tl_if_empty:nTF {#2} { \int_compare:nT {#1<=#3} {#5} }
796       {
797         \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
798       } } }
799 }

```

#### G.4 Resolving Greek symbol name control sequences

`\@@_resolve_greek:` This macro defines `\Alpha...``\omega` as their corresponding Unicode (mathematical italic) character. Remember that the mapping to upright or italic happens with the `mathcode` definitions, whereas these macros just stand for the literal Unicode characters.

```

800 \AtBeginDocument{\@@_resolve_greek:}
801 \cs_new:Npn \@@_resolve_greek:
802 {
803   \clist_map_inline:nn
804   {
805     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
806     alpha,beta,gamma,delta,epsilon,zeta,eta,theta,iota,kappa,lambda,
807     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
808     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,phi,chi,psi,omega,
809     varTheta,varsigma,vartheta,varkappa,varrho,varpi,varepsilon,varphi
810   }
811   {
812     \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
813     \tl_set:cx {up ##1} { \exp_not:N \symup \exp_not:c { ##1 } }
814     \tl_set:cx {it ##1} { \exp_not:N \symit \exp_not:c { ##1 } }
815   }
816 }

```

## H Maths alphabets

Defining commands like `\mathrm` is not as simple with Unicode fonts. In traditional  $\TeX$  maths font setups, you simply switch between different ‘families’ (`\fam`), which is analogous to changing from one font to another—a symbol such as ‘a’ will be upright in one font, bold in another, and so on.

In `pkgunicode-math`, a different mechanism is used to switch between styles. For every letter (start with `ascii a-zA-Z` and numbers to keep things simple for now), they are assigned a ‘mathcode’ with `\Umathcode` that maps from input letter to output font glyph slot. This is done with the equivalent of

```
% \Umathcode`\a = 7 1 "1D44E\relax
% \Umathcode`\b = 7 1 "1D44F\relax
% \Umathcode`\c = 7 1 "1D450\relax
% ...
```

When switching from regular letters to, say, `\mathrm`, we now need to execute a new mapping:

```
% \Umathcode`\a = 7 1 ``\a\relax
% \Umathcode`\b = 7 1 ``\b\relax
% \Umathcode`\c = 7 1 ``\c\relax
% ...
```

This is fairly straightforward to perform when we’re defining our own commands such as `\sympf` and so on. However, this means that ‘classical’  $\TeX$  font setups will break, because with the original mapping still in place, the engine will be attempting to insert unicode maths glyphs from a standard font.

## H.1 Hooks into $\LaTeX 2_{\epsilon}$

To overcome this, we patch `\use@mathgroup`. (An alternative is to patch `\extract@alph@from@version`, which constructs the `\mathXYZ` commands, but this method fails if the command has been defined using `\DeclareSymbolFontAlphabet`.) As far as I can tell, this is only used inside of commands such as `\mathXYZ`, so this shouldn’t have any major side-effects.

```
817 \cs_set:Npn \use@mathgroup #1 #2
818 {
819   \mode_if_math:T %<- not sure if this is really necessary since we’ve just checked for mmode and raised
      ror if not!
820   {
821     \math@bgroup
822     \cs_if_eq:cNF {M@f@encoding} #1 {#1}
823     \@@_switchto_literal:
824     \mathgroup #2 \relax
825     \math@egroup
826   }
827 }
```

## H.2 Setting styles

Algorithm for setting alphabet fonts. By default, when `range` is empty, we are in *implicit* mode. If `range` contains the name of the math alphabet, we are in *explicit* mode and do things slightly differently.

Implicit mode:



- Try and set all of the alphabet shapes.
- Check for the first glyph of each alphabet to detect if the font supports each alphabet shape.
- For alphabets that do exist, overwrite whatever’s already there.
- For alphabets that are not supported, *do nothing*. (This includes leaving the old alphabet definition in place.)

Explicit mode:

- Only set the alphabets specified.
- Check for the first glyph of the alphabet to detect if the font contains the alphabet shape in the Unicode math plane.
- For Unicode math alphabets, overwrite whatever’s already there.
- Otherwise, use the ASCII glyph slots instead.

### H.3 Defining the math style macros

We call the different shapes that a math alphabet can be a ‘math style’. Note that different alphabets can exist within the same math style. E.g., we call ‘bold’ the math style `bf` and within it there are upper and lower case Greek and Roman alphabets and Arabic numerals.

`\@@_prepare_mathstyle:n` #1 : math style name (e.g., `it` or `bb`)

Define the high level math alphabet macros (`\mathit`, etc.) in terms of unicode-math definitions. Use `\bgroup`/`\egroup` so s’scripts scan the whole thing.

The flag `\l_@@_mathstyle_tl` is for other applications to query the current math style.

```

828 \cs_new:Nn \@@_prepare_mathstyle:n
829 {
830   \seq_put_right:Nn \g_@@_mathstyles_seq {#1}
831   \@@_init_alphabet:n {#1}
832   \cs_set:cpn {_@@_sym_#1_aux:n}
833   { \use:c {@@_switchto_#1:} \math@egroup }
834   \cs_set_protected:cpx {sym#1}
835   {
836     \exp_not:n
837     {
838       \math@bgroup
839       \mode_if_math:F
840       {
841         \egroup\expandafter
842         \non@alpherr\expandafter{\csname sym#1\endcsname\space}
843       }
844       \tl_set:Nn \l_@@_mathstyle_tl {#1}

```

```

845     }
846     \exp_not:c {_@@_sym_#1_aux:n}
847   }
848 }

```

`\@@_init_alphabet:n` #1 : math alphabet name (e.g., it or bb)

This macro initialises the macros used to set up a math alphabet. First used when the math alphabet macro is first defined, but then used later when redefining a particular maths alphabet.

```

849 \cs_set:Nn \@@_init_alphabet:n
850 {
851   \@@_log:nx {alph-initialise} {#1}
852   \cs_set_eq:cN {\@@_switchto_#1:} \prg_do_nothing:
853 }

```

## H.4 Definition of alphabets and styles

First of all, we break up unicode into ‘named ranges’, such as up, bb, sfup, and so on, which refer to specific blocks of unicode that contain various symbols (usually alphabetical symbols).

```

854 \cs_new:Nn \@@_new_named_range:n
855 {
856   \prop_new:c {g_@@_named_range_#1_prop}
857 }
858 \clist_set:Nn \g_@@_named_ranges_clist
859 {
860   up, it, tt, bfup, bfit, bb , bbit, scr, bfscr, cal, bfcalf,
861   frak, bffrak, sfup, sfit, bfsfup, bfsfit, bfsf
862 }
863 \clist_map_inline:Nn \g_@@_named_ranges_clist
864 { \@@_new_named_range:n {#1} }

```

Each of these styles usually contains one or more ‘alphabets’, which are currently latin, Latin, greek, Greek, num, and misc, although there’s an implicit potential for more. misc is not included in the official list to avoid checking code.

```

865 \clist_new:N \g_@@_alphabets_seq
866 \clist_set:Nn \g_@@_alphabets_seq { latin, Latin, greek, Greek, num }

```

Each alphabet style needs to be configured. This happens in the unicode-math-alphabets.dtx file.

```

867 \cs_new:Nn \@@_new_alphabet_config:nnn
868 {
869   \prop_if_exist:cF {g_@@_named_range_#1_prop}
870   { \@@_warning:nnn {no-named-range} {#1} {#2} }
871
872   \prop_gput:cn {g_@@_named_range_#1_prop} { alpha_t1 }
873   {
874     \prop_item:cn {g_@@_named_range_#1_prop} { alpha_t1 }
875     {#2}

```

```

876     }
877 % Q: do I need to bother removing duplicates?
878
879 \cs_new:cn { @@_config_#1_#2:n } {#3}
880 }
881 \cs_new:Nn \@@_alphabet_config:nnn
882 {
883   \use:c {@@_config_#1_#2:n} {#3}
884 }
885 \prg_new_conditional:Nnn \@@_if_alphabet_exists:nn {T,TF}
886 {
887   \cs_if_exist:cTF {@@_config_#1_#2:n}
888   \prg_return_true: \prg_return_false:
889 }

```

The linking between named ranges and symbol style commands happens here. It's currently not using all of the machinery we're in the process of setting up above. Baby steps.

```

890 \cs_new:Nn \@@_default_mathalph:nnn
891 {
892   \seq_put_right:Nx \g_@@_named_ranges_seq { \tl_to_str:n {#1} }
893   \seq_put_right:Nn \g_@@_default_mathalph_seq {{#1}{#2}{#3}}
894   \prop_gput:cnn { g_@@_named_range_#1_prop } { default-alpha } {#2}
895 }
896 \@@_default_mathalph:nnn {up} {latin, Latin, greek, Greek, num, misc} {up}
897 \@@_default_mathalph:nnn {it} {latin, Latin, greek, Greek, misc} {it}
898 \@@_default_mathalph:nnn {bb} {latin, Latin, num, misc} {bb}
899 \@@_default_mathalph:nnn {bbit} {misc} {bbit}
900 \@@_default_mathalph:nnn {scr} {latin, Latin} {scr}
901 \@@_default_mathalph:nnn {cal} {Latin} {cal}
902 \@@_default_mathalph:nnn {bfcal} {Latin} {bfcal}
903 \@@_default_mathalph:nnn {frak} {latin, Latin} {frak}
904 \@@_default_mathalph:nnn {tt} {latin, Latin, num} {tt}
905 \@@_default_mathalph:nnn {sfup} {latin, Latin, num} {sfup}
906 \@@_default_mathalph:nnn {sfit} {latin, Latin} {sfit}
907 \@@_default_mathalph:nnn {bfup} {latin, Latin, greek, Greek, num, misc} {bfup}
908 \@@_default_mathalph:nnn {bfit} {latin, Latin, greek, Greek, misc} {bfit}
909 \@@_default_mathalph:nnn {bfscr} {latin, Latin} {bfscr}
910 \@@_default_mathalph:nnn {bffrak} {latin, Latin} {bffrak}
911 \@@_default_mathalph:nnn {bfsfup} {latin, Latin, greek, Greek, num, misc} {bfsfup}
912 \@@_default_mathalph:nnn {bfsfit} {latin, Latin, greek, Greek, misc} {bfsfit}

```

#### H.4.1 Define symbol style commands

Finally, all of the 'symbol styles' commands are set up, which are the commands to access each of the named alphabet styles. There is not a one-to-one mapping between symbol style commands and named style ranges!

```

913 \clist_map_inline:nn

```

```

914 {
915   up, it, bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf,
916   tt, bb, bbit, scr, bfscr, cal, bfcalf, frak, bffrak,
917   normal, literal, sf, bf,
918 }
919 { \@@_prepare_mathstyle:n {#1} }

```

#### H.4.2 New names for legacy textmath alphabet selection

In case a package option overwrites, say, `\mathbf` with `\sympf`.

```

920 \clist_map_inline:nn
921 { rm, it, bf, sf, tt }
922 { \cs_set_eq:cc { mathtext #1 } { math #1 } }

```

Perhaps these should actually be defined using a hypothetical unicode-math interface to creating new such styles. To come.

#### H.4.3 Replacing legacy pure-maths alphabets

The following are alphabets which do not have a math/text ambiguity.

```

923 \clist_map_inline:nn
924 {
925   normal, bb , bbit, scr, bfscr, cal, bfcalf, frak, bffrak, tt,
926   bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf
927 }
928 {
929   \cs_set:cpx { math #1 } { \exp_not:c { sym #1 } }
930 }

```

#### H.4.4 New commands for ambiguous alphabets

```

931 \AtBeginDocument{
932 \clist_map_inline:nn
933 { rm, it, bf, sf, tt }
934 {
935   \cs_set_protected:cpx { math #1 }
936   {
937     \exp_not:n { \bool_if:NTF } \exp_not:c { g_@@_ math #1 _text_bool }
938     { \exp_not:c { mathtext #1 } }
939     { \exp_not:c { sym #1 } }
940   }
941 }}

```

*Alias `\mathrm` as legacy name for `\mathup`*

```

942 \cs_set_protected:Npn \mathup { \mathrm }
943 \cs_set_protected:Npn \symrm { \symup }

```

#### H.4.5 Fixing up `\operator@font`

In LaTeX maths, the command `\operator@font` is defined that switches to the operator mathgroup. The classic example is the `\sin` in `sin{x}`; essentially we're using `\mathrm` to typeset the upright symbols, but the syntax is `\operator@font sin`.

It turns out that hooking into `\operator@font` is hard because all other maths font selection in 2e uses `\mathrm{...}` style.

Then reading source2e a little more I stumbled upon: (in the definition of `\select@group`)

```
We surround \select@group with braces so that functions using it can
be used directly after _ or ^. However, if we use oldstyle syntax where
the math alphabet doesn't have arguments (ie if \math@bgroup is not
\bgroup) we need to get rid of the extra group.
```

So there's a trick we can use. Because it's late and I'm tired, I went for the first thing that jumped out at me:

```
% \documentclass{article}
% \DeclareMathAlphabet\mathfoo{OT1}{lmdh}{m}{n}
% \begin{document}
% \makeatletter
% ${\operator@font Mod}\, x$
%
% \def\operator@font{%
%   \let \math@bgroup \relax
%   \def \math@egroup {\let \math@bgroup \@math@bgroup
%                         \let \math@egroup \@math@egroup}%
%   \mathfoo}
% ${\operator@font Mod}\, x$
% \end{document}
```

We define a new math alphabet `\mathfoo` to select the Latin Modern Dunhill font, and then locally redefine `\math@bgroup` to allow `\mathfoo` to be used without an argument temporarily.

Now that I've written this whole thing out, another solution pops to mind:

```
% \documentclass{article}
% \DeclareSymbolFont{foo}{OT1}{lmdh}{m}{n}
% \DeclareSymbolFontAlphabet\mathfoo{foo}
% \begin{document}
% \makeatletter
% ${\operator@font Mod}\, x$
%
% \def\operator@font{\mathgroup\symfoo}
% ${\operator@font Mod}\, x$
% \end{document}
```

I guess that's the better approach!!

Or perhaps I should just use `\fontswitch` to do the first solution with a nicer wrapper. I really should read things more carefully:

`\operator@font`

```
944 \cs_set:Npn \operator@font
945 {
946   \@@_switchto_literal:
947   \fontswitch {} { \g_@@_operator_mathfont_tl }
948 }
```

## H.5 *Defining the math alphabets per style*

`\@@_setup_alphabets:` This function is called within `\setmathfont` to configure the mapping between characters inside math styles.

```
949 \cs_new:Npn \@@_setup_alphabets:
950 {
```

If `range=` has been used to configure styles, those choices will be in `\l_@@_mathalph_seq`.  
If not, set up the styles implicitly:

```
951   \seq_if_empty:NTF \l_@@_mathalph_seq
952   {
953     \@@_log:n {setup-implicit}
954     \seq_set_eq:NN \l_@@_mathalph_seq \g_@@_default_mathalph_seq
955     \bool_set_true:N \l_@@_implicit_alph_bool
956     \@@_maybe_init_alphabet:n {sf}
957     \@@_maybe_init_alphabet:n {bf}
958     \@@_maybe_init_alphabet:n {bfsf}
959   }
```

If `range=` has been used then we're in explicit mode:

```
960   {
961     \@@_log:n {setup-explicit}
962     \bool_set_false:N \l_@@_implicit_alph_bool
963     \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
964     \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
965   }
966
967   % Now perform the mapping:
968   \seq_map_inline:Nn \l_@@_mathalph_seq
969   {
970     \tl_set:No \l_@@_style_tl { \use_i:nnn ##1 }
971     \clist_set:No \l_@@_alphabet_clist { \use_ii:nnn ##1 }
972     \tl_set:No \l_@@_remap_style_tl { \use_iii:nnn ##1 }
973
974     % If no set of alphabets is defined:
975     \clist_if_empty:NT \l_@@_alphabet_clist
976     {
977       \cs_set_eq:NN \@@_maybe_init_alphabet:n \@@_init_alphabet:n
978       \prop_get:cnN { g_@@_named_range_ \l_@@_style_tl _prop }
979       { default-alpha } \l_@@_alphabet_clist
980     }
981 }
```

```

982 \@@_setup_math_alphabet:
983 }
984 \seq_if_empty:NF \l_@@_missing_alph_seq { \@@_log:n { missing-alphabets } }
985 }

```

\@@\_setup\_math\_alphabet:

```

986 \cs_new:Nn \@@_setup_math_alphabet:
987 {

```

First check that at least one of the alphabets for the font shape is defined (this process is fast) ...

```

988 \clist_map_inline:Nn \l_@@_alphabet_clist
989 {
990 \tl_set:Nn \l_@@_alphabet_tl {##1}
991 \@@_if_alphabet_exists:nnTF \l_@@_style_tl \l_@@_alphabet_tl
992 {
993 \str_if_eq_x:nnTF {\l_@@_alphabet_tl} {misc}
994 {
995 \@@_maybe_init_alphabet:n \l_@@_style_tl
996 \clist_map_break:
997 }
998 {
999 \@@_glyph_if_exist:nT { \@@_to_usv:nn {\l_@@_style_tl} {\l_@@_alphabet_tl} }
1000 {
1001 \@@_maybe_init_alphabet:n \l_@@_style_tl
1002 \clist_map_break:
1003 }
1004 }
1005 }
1006 { \msg_warning:nnx {unicode-math} {no-alphabet} { \l_@@_style_tl / \l_@@_alphabet_tl } }
1007 }

```

...and then loop through them defining the individual ranges: (currently this process is slow)

```

1008 (debug) \csname TIC\endcsname
1009 \clist_map_inline:Nn \l_@@_alphabet_clist
1010 {
1011 \tl_set:Nx \l_@@_alphabet_tl { \tl_trim_spaces:n {##1} }
1012 \cs_if_exist:cT {@@_config_ \l_@@_style_tl _ \l_@@_alphabet_tl :n}
1013 {
1014 \exp_args:No \tl_if_eq:nnTF \l_@@_alphabet_tl {misc}
1015 {
1016 \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
1017 \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
1018 }
1019 {
1020 \@@_glyph_if_exist:nTF { \@@_to_usv:nn {\l_@@_remap_style_tl} {\l_@@_alphabet_tl} }
1021 {
1022 \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
1023 \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}

```

```

1024     }
1025     {
1026         \bool_if:NTF \l_@@_implicit_alph_bool
1027         {
1028             \seq_put_right:Nx \l_@@_missing_alph_seq
1029             {
1030                 \@backslashchar sym \l_@@_style_tl \space
1031                 (\tl_use:c{c_@@_math_alphabet_name_ \l_@@_alphabet_tl _tl})
1032             }
1033         }
1034         {
1035             \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {up}
1036         }
1037     }
1038 }
1039 }
1040 }
1041 <debug> \csname TOC\endcsname
1042 }

```

## H.6 Mapping ‘naked’ math characters

Before we show the definitions of the alphabet mappings using the functions `\@@_alphabet_config:nnn \l_@@_style_tl {##1} {...}`, we first want to define some functions to be used inside them to actually perform the character mapping.

### H.6.1 Functions

`\@@_map_char_single:nn` Wrapper for `\@@_map_char_noparse:nn` or `\@@_map_char_parse:nn` depending on the context.

`\@@_map_char_noparse:nn`

```

\@@_map_char_parse:nn
1043 \cs_new:Nn \@@_map_char_noparse:nn
1044 { \@@_set_mathcode:nnnn {#1}{\mathalpha}{\@@_symfont_tl}{#2} }
1045 \cs_new:Nn \@@_map_char_parse:nn
1046 {
1047     \@@_if_char_spec:nNNT {#1} {\@nil} {\mathalpha}
1048     { \@@_map_char_noparse:nn {#1}{#2} }
1049 }

```

`\@@_map_char_single:nnn`

#1 : char name (‘dotlessi’)  
 #2 : from alphabet(s)  
 #3 : to alphabet  
 Logical interface to `\@@_map_char_single:nn`.

```

1050 \cs_new:Nn \@@_map_char_single:nnn
1051 {
1052     \@@_map_char_single:nn { \@@_to_usv:nn {#1}{#3} }
1053     { \@@_to_usv:nn {#2}{#3} }
1054 }

```



```

\@@_map_chars_range:nnnn #1 : Number of chars (26)
                        #2 : From style, one or more (it)
                        #3 : To style (up)
                        #4 : Alphabet name (Latin)
First the function with numbers:
1055 \cs_set:Nn \@@_map_chars_range:nnn
1056 {
1057   \int_step_inline:nnnn {0}{1}{#1-1}
1058   { \@@_map_char_single:nn {#2+##1}{#3+##1} }
1059 }

```

And the wrapper with names:

```

1060 \cs_new:Nn \@@_map_chars_range:nnnn
1061 {
1062   \@@_map_chars_range:nnn {#1} { \@@_to_usv:nn {#2}{#4} }
1063   { \@@_to_usv:nn {#3}{#4} }
1064 }

```

## H.6.2 Functions for ‘normal’ alphabet symbols

```

\@@_set_normal_char:nnn
1065 \cs_set:Nn \@@_set_normal_char:nnn
1066 {
1067   \@@_usv_if_exist:nnT {#3} {#1}
1068   {
1069     \clist_map_inline:nn {#2}
1070     {
1071       \@@_set_mathalphabet_pos:nnnn {normal} {#1} {##1} {#3}
1072       \@@_map_char_single:nnn {##1} {#3} {#1}
1073     }
1074   }
1075 }

1076 \cs_new:Nn \@@_set_normal_Latin:nn
1077 {
1078   \clist_map_inline:nn {#1}
1079   {
1080     \@@_set_mathalphabet_Latin:nnn {normal} {##1} {#2}
1081     \@@_map_chars_range:nnnn {26} {##1} {#2} {Latin}
1082   }
1083 }

1084 \cs_new:Nn \@@_set_normal_latin:nn
1085 {
1086   \clist_map_inline:nn {#1}
1087   {
1088     \@@_set_mathalphabet_latin:nnn {normal} {##1} {#2}
1089     \@@_map_chars_range:nnnn {26} {##1} {#2} {latin}
1090   }
1091 }

```

```

1092 \cs_new:Nn \@@_set_normal_greek:nn
1093 {
1094   \clist_map_inline:nn {#1}
1095   {
1096     \@@_set_mathalphabet_greek:nnn {normal} {##1} {#2}
1097     \@@_map_chars_range:nnnn {25} {##1} {#2} {greek}
1098     \@@_map_char_single:nnn {##1} {#2} {epsilon}
1099     \@@_map_char_single:nnn {##1} {#2} {vartheta}
1100     \@@_map_char_single:nnn {##1} {#2} {varkappa}
1101     \@@_map_char_single:nnn {##1} {#2} {phi}
1102     \@@_map_char_single:nnn {##1} {#2} {varrho}
1103     \@@_map_char_single:nnn {##1} {#2} {varpi}
1104     \@@_set_mathalphabet_pos:nnnn {normal} {epsilon} {##1} {#2}
1105     \@@_set_mathalphabet_pos:nnnn {normal} {vartheta} {##1} {#2}
1106     \@@_set_mathalphabet_pos:nnnn {normal} {varkappa} {##1} {#2}
1107     \@@_set_mathalphabet_pos:nnnn {normal} {phi} {##1} {#2}
1108     \@@_set_mathalphabet_pos:nnnn {normal} {varrho} {##1} {#2}
1109     \@@_set_mathalphabet_pos:nnnn {normal} {varpi} {##1} {#2}
1110   }
1111 }

1112 \cs_new:Nn \@@_set_normal_Greek:nn
1113 {
1114   \clist_map_inline:nn {#1}
1115   {
1116     \@@_set_mathalphabet_Greek:nnn {normal} {##1} {#2}
1117     \@@_map_chars_range:nnnn {25} {##1} {#2} {Greek}
1118     \@@_map_char_single:nnn {##1} {#2} {varTheta}
1119     \@@_set_mathalphabet_pos:nnnn {normal} {varTheta} {##1} {#2}
1120   }
1121 }

1122 \cs_new:Nn \@@_set_normal_numbers:nn
1123 {
1124   \@@_set_mathalphabet_numbers:nnn {normal} {#1} {#2}
1125   \@@_map_chars_range:nnnn {10} {#1} {#2} {num}
1126 }

```

## H.7 Mapping chars inside a math style

### H.7.1 Functions for setting up the maths alphabets

`\@@_set_mathalphabet_char:Nnn` This is a wrapper for either `\@@_mathmap_noparse:nnn` or `\@@_mathmap_parse:Nnn`, depending on the context.

`\@@_mathmap_noparse:nnn` #1 : Maths alphabet, *e.g.*, ‘bb’  
 #2 : Input slot(s), *e.g.*, the slot for ‘A’ (comma separated)  
 #3 : Output slot, *e.g.*, the slot for ‘A’  
 Adds `\@@_set_mathcode:nnnn` declarations to the specified maths alphabet’s definition.

```

1127 \cs_new:Nn \@@_mathmap_noparse:nnn
1128 {
1129   \clist_map_inline:nn {#2}
1130   {
1131     \tl_put_right:cx {@@_switchto_#1:}
1132     {
1133       \@@_set_mathcode:nnnn {##1} {\mathalpha} {\@@_symfont_tl} {#3}
1134     }
1135   }
1136 }

```

`\@@_mathmap_parse:nnn` #1 : Maths alphabet, *e.g.*, ‘bb’  
 #2 : Input slot(s), *e.g.*, the slot for ‘A’ (comma separated)  
 #3 : Output slot, *e.g.*, the slot for ‘A’  
 When `\@@_if_char_spec:nNT` is executed, it populates the `\l_@@_char_nrange_clist` macro with slot numbers corresponding to the specified range. This range is used to conditionally add `\@@_set_mathcode:nnnn` declarations to the maths alphabet definition.

```

1137 \cs_new:Nn \@@_mathmap_parse:nnn
1138 {
1139   \clist_if_in:NnT \l_@@_char_nrange_clist {#3}
1140   {
1141     \@@_mathmap_noparse:nnn {#1}{#2}{#3}
1142   }
1143 }

```

`\@@_set_mathalphabet_char:nnnn` #1 : math style command  
 #2 : input math alphabet name  
 #3 : output math alphabet name  
 #4 : char name to map

```

1144 \cs_new:Nn \@@_set_mathalphabet_char:nnnn
1145 {
1146   \@@_set_mathalphabet_char:nnn {#1} { \@@_to_usv:nn {#2} {#4} }
1147   { \@@_to_usv:nn {#3} {#4} }
1148 }

```

`\@@_set_mathalph_range:nnnn` #1 : Number of iterations  
 #2 : Maths alphabet  
 #3 : Starting input char (single)  
 #4 : Starting output char  
 Loops through character ranges setting `\mathcode`. First the version that uses numbers:

```

1149 \cs_new:Nn \@@_set_mathalph_range:nnnn
1150 {
1151   \int_step_inline:nnnn {0} {1} {#1-1}
1152   { \@@_set_mathalphabet_char:nnn {#2} { ##1 + #3 } { ##1 + #4 } }
1153 }

```

Then the wrapper version that uses names:

```

1154 \cs_new:Nn \@@_set_mathalph_range:nnnnn
1155 {
1156   \@@_set_mathalph_range:nnnn {#1} {#2} { \@@_to_usv:nn {#3} {#5} }
1157                                   { \@@_to_usv:nn {#4} {#5} }
1158 }

```

### *H.7.2 Individual mapping functions for different alphabets*

```

1159 \cs_new:Nn \@@_set_mathalphabet_pos:nnnn
1160 {
1161   \@@_usv_if_exist:nnT {#4} {#2}
1162   {
1163     \clist_map_inline:nn {#3}
1164       { \@@_set_mathalphabet_char:nnnn {#1} {##1} {#4} {#2} }
1165   }
1166 }

1167 \cs_new:Nn \@@_set_mathalphabet_numbers:nnn
1168 {
1169   \clist_map_inline:nn {#2}
1170     { \@@_set_mathalph_range:nnnnn {10} {#1} {##1} {#3} {num} }
1171 }

1172 \cs_new:Nn \@@_set_mathalphabet_Latin:nnn
1173 {
1174   \clist_map_inline:nn {#2}
1175     { \@@_set_mathalph_range:nnnnn {26} {#1} {##1} {#3} {Latin} }
1176 }

1177 \cs_new:Nn \@@_set_mathalphabet_latin:nnn
1178 {
1179   \clist_map_inline:nn {#2}
1180     {
1181       \@@_set_mathalph_range:nnnnn {26} {#1} {##1} {#3} {latin}
1182       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {h}
1183     }
1184 }

1185 \cs_new:Nn \@@_set_mathalphabet_Greek:nnn
1186 {
1187   \clist_map_inline:nn {#2}
1188     {
1189       \@@_set_mathalph_range:nnnnn {25} {#1} {##1} {#3} {Greek}
1190       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varTheta}
1191     }
1192 }

1193 \cs_new:Nn \@@_set_mathalphabet_greek:nnn
1194 {
1195   \clist_map_inline:nn {#2}
1196     {

```

```

1197 \@@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {greek}
1198 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {epsilon}
1199 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {vartheta}
1200 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varkappa}
1201 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {phi}
1202 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varrho}
1203 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varpi}
1204 }
1205 }

```

## I *A token list to contain the data of the math table*

Instead of `\input`-ing the unicode math table every time we want to re-read its data, we save it within a macro. This has two advantages: 1. it should be slightly faster, at the expense of memory; 2. we don't need to worry about catcodes later, since they're frozen at this point.

In time, the case statement inside `set_mathsymbol` will be moved in here to avoid re-running it every time.

```

1206 \cs_new:Npn \@@_symbol_setup:
1207 {
1208   \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4
1209   {
1210     \exp_not:n { \_@@_sym:nnn {##1} {##2} {##3} }
1211   }
1212 }
1213 \tl_set_from_file:x:Nnn \g_@@_mathtable_tl {\@@_symbol_setup:} {unicode-math-
table.tex}

```

`\@@_input_math_symbol_table:` This function simply expands to the token list containing all the data.

```

1214 \cs_new:Nn \@@_input_math_symbol_table: {\g_@@_mathtable_tl}

```

## J *Definitions of the active math characters*

Now give `\_@@_sym:nnn` a definition in terms of `\@@_cs_set_eq_active_char:Nw` and we're good to go.

Ensure catcodes are appropriate; make sure `#` is an 'other' so that we don't get confused with `\mathoctothorpe`.

```

1215 \AtBeginDocument{\@@_define_math_chars:}
1216 \cs_new:Nn \@@_define_math_chars:
1217 {
1218   \group_begin:
1219   \cs_set:Npn \_@@_sym:nnn ##1##2##3
1220   {
1221     \tl_if_in:nnT
1222     { \mathord \mathalpha \mathbin \mathrel \mathpunct \mathop \mathfence }
1223     {##3}
1224   {

```

```

1225     \exp_last_unbraced:NNx \cs_gset_eq:NN ##2 { \Ucharcat ##1 ~ 12 ~ }
1226   }
1227 }
1228 \@@_input_math_symbol_table:
1229 \group_end:
1230 }
1231 </package&(XE|LU)>

```

## K Preamble

The prefix for unicode-math is um:

```
1232 <@@=um>
```

The shared part of the code starts here before the split above.

```
1233 <*preamble&!XE&!LU>
```

Bail early if using pdf $\TeX$ .

```

1234 \ifdefined\XeTeXversion
1235   \ifdim\number\XeTeXversion\XeTeXrevision in<0.9998in%
1236     \PackageError{unicode-math}{%
1237       Cannot run with this version of XeTeX!\MessageBreak
1238       You need XeTeX 0.9998 or newer.%
1239     }\@ehd
1240   \fi
1241 \else\ifdefined\luatexversion
1242   \ifnum\luatexversion<64%
1243     \PackageError{unicode-math}{%
1244       Cannot run with this version of LuaTeX!\MessageBreak
1245       You need LuaTeX 0.64 or newer.%
1246     }\@ehd
1247   \fi
1248 \else
1249   \PackageError{unicode-math}{%
1250     Cannot be run with pdfLaTeX!\MessageBreak
1251     Use XeLaTeX or LuaLaTeX instead.%
1252   }\@ehd
1253 \fi\fi

```

## Packages

```

1254 \RequirePackage{expl3}[2015/03/01]
1255 \RequirePackage{ucharcat}
1256 \RequirePackage{xparse}
1257 \RequirePackage{l3keys2e}
1258 \RequirePackage{fontspec}[2015/03/14]
1259 \RequirePackage{fix-cm} % avoid some warnings
1260 \RequirePackage{filehook}
1261 \ExplSyntaxOn

```

Variants needed from expl3:

```
1262 \cs_set_protected_nopar:Npn \exp_last_unbraced:NNx { \::N \::x_unbraced \::: }
```

For fontspec:

```
1263 \cs_generate_variant:Nn \fontspec_set_family:Nnn {Nx}
```

```
1264 \cs_generate_variant:Nn \fontspec_set_fontface:NNnn {NNx}
```

### *Conditionals*

```
1265 \bool_new:N \l_@@_ot_math_bool
```

```
1266 \bool_new:N \l_@@_init_bool
```

```
1267 \bool_new:N \l_@@_implicit_alph_bool
```

```
1268 \bool_new:N \g_@@_mainfont_already_set_bool
```

For math-style:

```
1269 \bool_new:N \g_@@_literal_bool
```

```
1270 \bool_new:N \g_@@_upLatin_bool
```

```
1271 \bool_new:N \g_@@_uplatin_bool
```

```
1272 \bool_new:N \g_@@_upGreek_bool
```

```
1273 \bool_new:N \g_@@_upgreek_bool
```

For bold-style:

```
1274 \bool_new:N \g_@@_bfliteral_bool
```

```
1275 \bool_new:N \g_@@_bfupLatin_bool
```

```
1276 \bool_new:N \g_@@_bfuplatin_bool
```

```
1277 \bool_new:N \g_@@_bfupGreek_bool
```

```
1278 \bool_new:N \g_@@_bfupgreek_bool
```

For sans-style:

```
1279 \bool_new:N \g_@@_upsans_bool
```

```
1280 \bool_new:N \g_@@_sfliteral_bool
```

For assorted package options:

```
1281 \bool_new:N \g_@@_upNabla_bool
```

```
1282 \bool_new:N \g_@@_uppartial_bool
```

```
1283 \bool_new:N \g_@@_literal_Nabla_bool
```

```
1284 \bool_new:N \g_@@_literal_partial_bool
```

```
1285 \bool_new:N \l_@@_smallfrac_bool
```

```
1286 \bool_new:N \g_@@_literal_colon_bool
```

```
1287 \bool_new:N \g_@@_mathrm_text_bool
```

```
1288 \bool_new:N \g_@@_mathit_text_bool
```

```
1289 \bool_new:N \g_@@_mathbf_text_bool
```

```
1290 \bool_new:N \g_@@_mathsf_text_bool
```

```
1291 \bool_new:N \g_@@_mathtt_text_bool
```

### *Variables*

```
1292 \int_new:N \g_@@_fam_int
```

For displaying in warning messages, etc.:

```
1293 \tl_const:Nn \c_@@_math_alphabet_name_latin_tl {Latin,~lowercase}
```

```
1294 \tl_const:Nn \c_@@_math_alphabet_name_Latin_tl {Latin,~uppercase}
```

```
1295 \tl_const:Nn \c_@@_math_alphabet_name_greek_tl {Greek,~lowercase}
```

```

1296 \tl_const:Nn \c_@@_math_alphabet_name_Greek_tl {Greek,~uppercase}
1297 \tl_const:Nn \c_@@_math_alphabet_name_num_tl {Numerals}
1298 \tl_const:Nn \c_@@_math_alphabet_name_misc_tl {Misc.}
1299 \tl_new:N \l_@@_mathstyle_tl

Used to store the font switch for the \operator@font.

1300 \tl_new:N \g_@@_operator_mathfont_tl

Variables:
1301 \seq_new:N \l_@@_missing_alph_seq
1302 \seq_new:N \l_@@_mathalph_seq
1303 \seq_new:N \l_@@_char_range_seq
1304 \seq_new:N \l_@@_mclass_range_seq
1305 \seq_new:N \l_@@_cmd_range_seq

```

`\g_@@_mathclasses_seq` Every math class.

```

1306 \seq_new:N \g_@@_mathclasses_seq
1307 \seq_set_from_clist:Nn \g_@@_mathclasses_seq
1308 {
1309   \mathord,\mathalpha,\mathbin,\mathrel,\mathpunct,
1310   \mathop,
1311   \mathopen,\mathclose,
1312   \mathfence,\mathover,\mathunder,
1313   \mathaccent,\mathbotaccent,\mathaccentwide,\mathbotaccentwide
1314 }

```

`\g_@@_default_mathalph_seq` This sequence stores the alphabets in each math style.

```

1315 \seq_new:N \g_@@_default_mathalph_seq

```

`\g_@@_mathstyles_seq` This is every ‘named range’ and every ‘math style’ known to unicode-math. A named range is such as “bfit” and “sfit”, which are also math styles (with `\symbfit` and `\symsfit`). ‘Mathstyles’ are a superset of named ranges and also include commands such as `\symbf` and `\symsf`.

N.B. for parsing purposes ‘named ranges’ are defined as strings!

```

1316 \seq_new:N \g_@@_named_ranges_seq
1317 \seq_new:N \g_@@_mathstyles_seq

1318 \muskip_new:N \g_@@_primekern_muskip
1319 \muskip_gset:Nn \g_@@_primekern_muskip { -\thinmuskip/2 }% arbitrary
1320 \int_new:N \l_@@_primecount_int
1321 \prop_new:N \g_@@_supers_prop
1322 \prop_new:N \g_@@_subs_prop
1323 \tl_new:N \l_not_token_name_tl

```

## K.1 Extras

What might end up being provided by the kernel.



`\@@_glyph_if_exist:nTF` : TODO: Generalise for arbitrary fonts! `\l_@@_font` is not always the one used for a specific glyph!!

```

1324 \prg_new_conditional:Nnn \@@_glyph_if_exist:n {p,TF,T,F}
1325 {
1326   \etex_iffontchar:D \l_@@_font #1 \scan_stop:
1327   \prg_return_true:
1328   \else:
1329   \prg_return_false:
1330   \fi:
1331 }

```

`\@@_set_mathcode:nnnn` These are all wrappers for the primitive commands that take numerical input only.

```

\@@_set_mathcode:nnn 1332 \cs_set:Npn \@@_set_mathcode:nnnn #1#2#3#4 {
\@@_set_mathchar:NNnn 1333   \Umathcode \int_eval:n {#1} =
\@@_set_mathchar:cNnn 1334   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
\@@_set_delcode:nnn 1335 }
\@@_radical:nn 1336 \cs_set:Npn \@@_set_mathcode:nnn #1#2#3 {
\@@_delimiter:Nnn 1337   \Umathcode \int_eval:n {#1} =
\@@_accent:nnn 1338   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
\@@_accent_keyword: 1339 }
1340 \cs_set:Npn \@@_set_mathchar:NNnn #1#2#3#4 {
1341   \Umathchardef #1 =
1342   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
1343 }
1344 \cs_new:Nn \@@_set_delcode:nnn {
1345   \Udelcode#2 = \csname sym#1\endcsname #3 \scan_stop:
1346 }
1347 \cs_new:Nn \@@_radical:nn {
1348   \Uradical \csname sym#1\endcsname #2 \scan_stop:
1349 }
1350 \cs_new:Nn \@@_delimiter:NNn {
1351   \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
1352 }
1353 \cs_new:Nn \@@_accent:nnn {
1354   \Umathaccent #1~ \mathchar@type\mathaccent \use:c { sym #2 } #3 \scan_stop:
1355 }
1356 \cs_generate_variant:Nn \@@_set_mathchar:NNnn {c}

```

`\@@_char_gmake_mathactive:N`

```

\@@_char_gmake_mathactive:n 1357 \cs_new:Nn \@@_char_gmake_mathactive:N
1358 {
1359   \global\mathcode `#1 = "8000 \scan_stop:
1360 }
1361 \cs_new:Nn \@@_char_gmake_mathactive:n
1362 {
1363   \global\mathcode #1 = "8000 \scan_stop:
1364 }

```

## K.2 Alphabet Unicode positions

Before we begin, let's define the positions of the various Unicode alphabets so that our code is a little more readable.<sup>5</sup>

Rather than 'readable', in the end, this makes the code more extensible.

```

1365 \cs_new:Nn \usv_set:nnn
1366 { \tl_set:cn { g_@@_#1_#2_usv } {#3} }
1367 \cs_new:Nn \@@_to_usv:nn
1368 { \use:c { g_@@_#1_#2_usv } }
1369 \prg_new_conditional:Nnn \@@_usv_if_exist:nn {T,F,TF}
1370 {
1371   \cs_if_exist:cTF { g_@@_#1_#2_usv }
1372   \prg_return_true: \prg_return_false:
1373 }

```

## K.3 Package options

`\unimathsetup` This macro can be used in lieu of or later to override options declared when the package is loaded.

```

1374 \DeclareDocumentCommand \unimathsetup {m}
1375 { \keys_set:nn {unicode-math} {#1} }

```

`\@@_keys_choices:nn` To simplify the creation of option keys, let's iterate in pairs rather than worry about equals signs and commas.

```

1376 \cs_new:Nn \@@_keys_choices:nn
1377 {
1378   \cs_set:Npn \@@_keys_choices_fn:nn { \@@_keys_choices_aux:nnn {#1} }
1379   \use:x
1380   {
1381     \exp_not:N \keys_define:nn {unicode-math}
1382     {
1383       #1 .choice: ,
1384       \@@_tl_map_dbl:nN {#2} \@@_keys_choices_fn:nn
1385     }
1386   }
1387 }
1388 \cs_new:Nn \@@_keys_choices_aux:nnn { #1 / #2 .code:n = { \exp_not:n {#3} } , }
1389
1390 \cs_new:Nn \@@_tl_map_dbl:nN
1391 {
1392   \__@@_tl_map_dbl:Nnn #2 #1 \q_recursion_tail {}{} \q_recursion_stop
1393 }
1394 \cs_new:Nn \__@@_tl_map_dbl:Nnn
1395 {
1396   \quark_if_recursion_tail_stop:n {#2}
1397   \quark_if_recursion_tail_stop:n {#3}
1398   #1 {#2} {#3}

```

---

<sup>5</sup>'u.s.v.' stands for 'Unicode scalar value'.

```

1399 \_@@_tl_map_dbl:Nnn #1
1400 }

```

### *Compatibility*

```

1401 \@@_keys_choices:nn {mathup}
1402 {
1403   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
1404   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
1405 }
1406 \@@_keys_choices:nn {mathrm}
1407 {
1408   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
1409   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
1410 }
1411 \@@_keys_choices:nn {mathit}
1412 {
1413   {sym} { \bool_set_false:N \g_@@_mathit_text_bool }
1414   {text} { \bool_set_true:N \g_@@_mathit_text_bool }
1415 }
1416 \@@_keys_choices:nn {mathbf}
1417 {
1418   {sym} { \bool_set_false:N \g_@@_mathbf_text_bool }
1419   {text} { \bool_set_true:N \g_@@_mathbf_text_bool }
1420 }
1421 \@@_keys_choices:nn {mathsf}
1422 {
1423   {sym} { \bool_set_false:N \g_@@_mathsf_text_bool }
1424   {text} { \bool_set_true:N \g_@@_mathsf_text_bool }
1425 }
1426 \@@_keys_choices:nn {mathtt}
1427 {
1428   {sym} { \bool_set_false:N \g_@@_mathtt_text_bool }
1429   {text} { \bool_set_true:N \g_@@_mathtt_text_bool }
1430 }

```

### *math-style*

```

1431 \@@_keys_choices:nn {normal-style}
1432 {
1433   {ISO} {
1434     \bool_set_false:N \g_@@_literal_bool
1435     \bool_set_false:N \g_@@_upGreek_bool
1436     \bool_set_false:N \g_@@_upgreek_bool
1437     \bool_set_false:N \g_@@_upLatin_bool
1438     \bool_set_false:N \g_@@_uplatin_bool
1439   }
1440   {TeX} {
1441     \bool_set_false:N \g_@@_literal_bool
1442     \bool_set_true:N \g_@@_upGreek_bool

```

```

1443         \bool_set_false:N \g_@@_upgreek_bool
1444         \bool_set_false:N \g_@@_upLatin_bool
1445         \bool_set_false:N \g_@@_uplatin_bool
1446     }
1447     {french} {
1448         \bool_set_false:N \g_@@_literal_bool
1449         \bool_set_true:N \g_@@_upGreek_bool
1450         \bool_set_true:N \g_@@_upgreek_bool
1451         \bool_set_true:N \g_@@_upLatin_bool
1452         \bool_set_false:N \g_@@_uplatin_bool
1453     }
1454     {upright} {
1455         \bool_set_false:N \g_@@_literal_bool
1456         \bool_set_true:N \g_@@_upGreek_bool
1457         \bool_set_true:N \g_@@_upgreek_bool
1458         \bool_set_true:N \g_@@_upLatin_bool
1459         \bool_set_true:N \g_@@_uplatin_bool
1460     }
1461     {literal} {
1462         \bool_set_true:N \g_@@_literal_bool
1463     }
1464 }
1465 \@@_keys_choices:nn {math-style}
1466 {
1467     {ISO} {
1468         \unimathsetup { nabra=upright, partial=italic,
1469             normal-style=ISO, bold-style=ISO, sans-style=italic }
1470     }
1471     {TeX} {
1472         \unimathsetup { nabra=upright, partial=italic,
1473             normal-style=TeX, bold-style=TeX, sans-style=upright }
1474     }
1475     {french} {
1476         \unimathsetup { nabra=upright, partial=upright,
1477             normal-style=french, bold-style=upright, sans-style=upright }
1478     }
1479     {upright} {
1480         \unimathsetup { nabra=upright, partial=upright,
1481             normal-style=upright, bold-style=upright, sans-style=upright }
1482     }
1483     {literal} {
1484         \unimathsetup { colon=literal, nabra=literal, partial=literal,
1485             normal-style=literal, bold-style=literal, sans-style=literal }
1486     }
1487 }

```

*bold-style*

```

1488 \@@_keys_choices:nn {bold-style}

```

```

1489 {
1490     {ISO} {
1491         \bool_set_false:N \g_@@_bfliteral_bool
1492         \bool_set_false:N \g_@@_bfupGreek_bool
1493         \bool_set_false:N \g_@@_bfupgreek_bool
1494         \bool_set_false:N \g_@@_bfupLatin_bool
1495         \bool_set_false:N \g_@@_bfuplatin_bool
1496     }
1497     {TeX} {
1498         \bool_set_false:N \g_@@_bfliteral_bool
1499         \bool_set_true:N \g_@@_bfupGreek_bool
1500         \bool_set_false:N \g_@@_bfupgreek_bool
1501         \bool_set_true:N \g_@@_bfupLatin_bool
1502         \bool_set_true:N \g_@@_bfuplatin_bool
1503     }
1504     {upright} {
1505         \bool_set_false:N \g_@@_bfliteral_bool
1506         \bool_set_true:N \g_@@_bfupGreek_bool
1507         \bool_set_true:N \g_@@_bfupgreek_bool
1508         \bool_set_true:N \g_@@_bfupLatin_bool
1509         \bool_set_true:N \g_@@_bfuplatin_bool
1510     }
1511     {literal} {
1512         \bool_set_true:N \g_@@_bfliteral_bool
1513     }
1514 }

```

### *sans-style*

```

1515 \@@_keys_choices:nn {sans-style}
1516 {
1517     {italic} { \bool_set_false:N \g_@@_upsans_bool }
1518     {upright} { \bool_set_true:N \g_@@_upsans_bool }
1519     {literal} { \bool_set_true:N \g_@@_sfliteral_bool }
1520 }

```

### *Nabla and partial*

```

1521 \@@_keys_choices:nn {nabla}
1522 {
1523     {upright} {
1524         \bool_set_false:N \g_@@_literal_Nabla_bool
1525         \bool_set_true:N \g_@@_upNabla_bool
1526     }
1527     {italic} {
1528         \bool_set_false:N \g_@@_literal_Nabla_bool
1529         \bool_set_false:N \g_@@_upNabla_bool
1530     }
1531     {literal} { \bool_set_true:N \g_@@_literal_Nabla_bool }
1532 }

```

```

1533 \@@_keys_choices:nn {partial}
1534 {
1535   {upright} {
1536     \bool_set_false:N \g_@@_literal_partial_bool
1537     \bool_set_true:N \g_@@_uppartial_bool
1538   }
1539   {italic} {
1540     \bool_set_false:N \g_@@_literal_partial_bool
1541     \bool_set_false:N \g_@@_uppartial_bool
1542   }
1543   {literal} { \bool_set_true:N \g_@@_literal_partial_bool }
1544 }

```

### *Colon style*

```

1545 \@@_keys_choices:nn {colon}
1546 {
1547   {literal} { \bool_set_true:N \g_@@_literal_colon_bool }
1548   {TeX} { \bool_set_false:N \g_@@_literal_colon_bool }
1549 }

```

### *Slash delimiter style*

```

1550 \@@_keys_choices:nn {slash-delimiter}
1551 {
1552   {ascii} { \tl_set:Nn \g_@@_slash_delimiter_usv {"002F} }
1553   {frac} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2044} }
1554   {div} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2215} }
1555 }

```

### *Active fraction style*

```

1556 \@@_keys_choices:nn {active-frac}
1557 {
1558   {small}
1559   {
1560     \cs_if_exist:NTF \tfrac
1561     { \bool_set_true:N \l_@@_smallfrac_bool }
1562     {
1563       \@@_warning:n {no-tfrac}
1564       \bool_set_false:N \l_@@_smallfrac_bool
1565     }
1566     \use:c {@@_setup_active_frac:}
1567   }
1568
1569   {normalsize}
1570   {
1571     \bool_set_false:N \l_@@_smallfrac_bool
1572     \use:c {@@_setup_active_frac:}
1573   }
1574 }

```

### *Debug/tracing*

```
1575 \keys_define:nn {unicode-math}
1576 {
1577   warnings-off .code:n =
1578   {
1579     \clist_map_inline:nn {#1}
1580     { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
1581   }
1582 }

1583 \@@_keys_choices:nn {trace}
1584 {
1585   {on}    {} % default
1586   {debug} { \msg_redirect_module:nnn { unicode-math } { log } { warning } }
1587   {off}   { \msg_redirect_module:nnn { unicode-math } { log } { none } }
1588 }

1589 \unimathsetup {math-style=TeX}
1590 \unimathsetup {slash-delimiter=ascii}
1591 \unimathsetup {trace=off}
1592 \unimathsetup {mathrm=text,mathit=text,mathbf=text,mathsf=text,mathtt=text}
1593 \cs_if_exist:NT \tfrac { \unimathsetup {active-frac=small} }
1594 \ProcessKeysOptions {unicode-math}
```

### *K.4 Programmers' interface*

`\unimath_get_mathstyle:` This command expands to the currently math style.

```
1595 \cs_new:Nn \unimath_get_mathstyle:
1596 {
1597   \tl_use:N \l_@@_mathstyle_tl
1598 }
```

### *K.5 Overcoming \onlypreamble*

The requirement of only setting up the maths fonts in the preamble is now removed. The following list might be overly ambitious.

```
1599 \tl_map_inline:nn
1600 {
1601   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
1602   \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
1603   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
1604   \version@list\version@elt\alpha@list\alpha@elt
1605   \restore@mathversion\init@restore@version\dorestore@version\process@table
1606   \new@mathversion\DeclareSymbolFont\group@list\group@elt
1607   \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
1608   \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
1609   \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
1610   \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
1611   \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
```

```

1612 \set@@mathdelimater\DeclareMathRadical\mathchar@type
1613 \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
1614 }
1615 {
1616 \tl_remove_once:Nn \@preamblecmds {\do#1}
1617 }
      End of preamble code.
1618 </preamble&!XE&!LU>

```

## L *Error messages*

These are defined at the beginning of the package, but we leave their definition until now in the source to keep them out of the way.

```

1619 <*msg>

      Wrapper functions:
1620 \cs_new:Npn \@@_error:n    { \msg_error:nn {unicode-math} }
1621 \cs_new:Npn \@@_warning:n  { \msg_warning:nn {unicode-math} }
1622 \cs_new:Npn \@@_warning:nnn { \msg_warning:nnxx {unicode-math} }
1623 \cs_new:Npn \@@_log:n      { \msg_log:nn {unicode-math} }
1624 \cs_new:Npn \@@_log:nx     { \msg_log:nnx {unicode-math} }

1625 \msg_new:nnn {unicode-math} {no-tfrac}
1626 {
1627   Small~ fraction~ command~ \protect\tfrac~ not~ defined.\\
1628   Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
1629 }
1630 \msg_new:nnn {unicode-math} {default-math-font}
1631 {
1632   Defining~ the~ default~ maths~ font~ as~ '\l_@@_fontname_tl'.
1633 }
1634 \msg_new:nnn {unicode-math} {setup-implicit}
1635 {
1636   Setup~ alphabets:~ implicit~ mode.
1637 }
1638 \msg_new:nnn {unicode-math} {setup-explicit}
1639 {
1640   Setup~ alphabets:~ explicit~ mode.
1641 }
1642 \msg_new:nnn {unicode-math} {alph-initialise}
1643 {
1644   Initialising~ \@backslashchar math#1.
1645 }
1646 \msg_new:nnn {unicode-math} {setup-alph}
1647 {
1648   Setup~ alphabet:~ #1.
1649 }
1650 \msg_new:nnn {unicode-math} {no-alphabet}

```



```

1651 {
1652   I~ am~ trying~ to~ set~ up~ alphabet~"#1"~ but~ there~ are~ no~ configura-
        tion~ settings~ for~ it.~
1653   (See~ source~ file~ "unicode-math-alphabets.dtx"~ to~ debug.)
1654 }
1655 \msg_new:nnn { unicode-math } { no-named-range }
1656 {
1657   I~ am~ trying~ to~ define~ new~ alphabet~ "#2"~ in~ range~ "#1",~ but~ range~ "#1"~ hasn't~ been~ de-
        fined~ yet.
1658 }
1659 \msg_new:nnn { unicode-math } { missing-alphabets }
1660 {
1661   Missing~math~alphabets~in~font~ "\fontname\l_@@_font" \ \ \
1662   \seq_map_function:NN \l_@@_missing_alph_seq \@@_print_indent:n
1663 }
1664 \cs_new:Nn \@@_print_indent:n { \space\space\space\space #1 \ \ }
1665 \msg_new:nnn { unicode-math } { macro-expected }
1666 {
1667   I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
1668 }
1669 \msg_new:nnn { unicode-math } { wrong-meaning }
1670 {
1671   I've~ expected~ #1~ to~ have~ the~ meaning~ #3,~ but~ it~ has~ the~ mean-
        ing~ #2.
1672 }
1673 \msg_new:nnn { unicode-math } { patch-macro }
1674 {
1675   I'm~ going~ to~ patch~ macro~ #1.
1676 }
1677 \msg_new:nnn { unicode-math } { mathtools-overbracket } {
1678   Using~ \token_to_str:N \overbracket~ and~
1679   \token_to_str:N \underbracket~ from~
1680   'mathtools'~ package.\ \
1681   \ \
1682   Use~ \token_to_str:N \Uoverbracket~ and~
1683   \token_to_str:N \Uunderbracket~ for~
1684   original~ 'unicode-math'~ definition.
1685 }
1686 \msg_new:nnn { unicode-math } { mathtools-colon } {
1687   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
1688   the~ 'mathtools'~ package: \ \ \
1689   \ \ \ \ \token_to_str:N \dblcolon,~
1690   \token_to_str:N \coloneqq,~
1691   \token_to_str:N \Coloneqq,~
1692   \token_to_str:N \eqqcolon. \ \ \
1693   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
1694   commands,~ using~ them~ will~ lead~ to~ inconsistencies.
1695 }
1696 \msg_new:nnn { unicode-math } { colonequals } {

```

```

1697 I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
1698 the~ 'colonequals'~ package: \\ \\
1699 \\ \\ \\ \token_to_str:N \ratio,~
1700         \token_to_str:N \coloncolon,~
1701         \token_to_str:N \minuscolon, \\
1702 \\ \\ \\ \token_to_str:N \colonequals,~
1703         \token_to_str:N \equalscolon,~
1704         \token_to_str:N \coloncolonequals. \\ \\
1705 Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
1706 commands,~ using~ them~ will~ lead~ to~ inconsistencies.~
1707 Furthermore,~ changing~ \token_to_str:N \colonsep \c_space_tl
1708 or~ \token_to_str:N \doublecolonsep \c_space_tl won't~ have~
1709 any~ effect~ on~ the~ re-defined~ commands.
1710 }
1711 </msg>

```

## L.1 Alphabet Unicode positions

Before we begin, let's define the positions of the various Unicode alphabets so that our code is a little more readable.<sup>6</sup>

```

1712 <*usv>

```

### Alphabets

```

1713 \usv_set:nnn {normal} {num} {48}
1714 \usv_set:nnn {normal} {Latin}{1D434}
1715 \usv_set:nnn {normal} {latin}{1D44E}
1716 \usv_set:nnn {normal} {Greek}{1D6E2}
1717 \usv_set:nnn {normal} {greek}{1D6FC}
1718 \usv_set:nnn {normal}{varTheta} {"1D6F3}
1719 \usv_set:nnn {normal}{epsilon}{1D716}
1720 \usv_set:nnn {normal}{vartheta} {"1D717}
1721 \usv_set:nnn {normal}{varkappa} {"1D718}
1722 \usv_set:nnn {normal}{phi} {"1D719}
1723 \usv_set:nnn {normal}{varrho} {"1D71A}
1724 \usv_set:nnn {normal}{varpi} {"1D71B}
1725 \usv_set:nnn {normal} {Nabla}{1D6FB}
1726 \usv_set:nnn {normal} {partial}{1D715}
1727
1728 \usv_set:nnn {up} {num} {48}
1729 \usv_set:nnn {up} {Latin}{65}
1730 \usv_set:nnn {up} {latin}{97}
1731 \usv_set:nnn {up} {Greek}{391}
1732 \usv_set:nnn {up} {greek}{3B1}
1733 \usv_set:nnn {it} {Latin}{1D434}
1734 \usv_set:nnn {it} {latin}{1D44E}
1735 \usv_set:nnn {it} {Greek}{1D6E2}

```

---

<sup>6</sup>'u.s.v.' stands for 'Unicode scalar value'.

1736 \usv\_set:nnn {it} {greek}{ "1D6FC}  
 1737 \usv\_set:nnn {bb} {num} { "1D7D8}  
 1738 \usv\_set:nnn {bb} {Latin}{ "1D538}  
 1739 \usv\_set:nnn {bb} {latin}{ "1D552}  
 1740 \usv\_set:nnn {scr} {Latin}{ "1D49C}  
 1741 \usv\_set:nnn {cal} {Latin}{ "1D49C}  
 1742 \usv\_set:nnn {scr} {latin}{ "1D4B6}  
 1743 \usv\_set:nnn {frak}{Latin}{ "1D504}  
 1744 \usv\_set:nnn {frak}{latin}{ "1D51E}  
 1745 \usv\_set:nnn {sf} {num} { "1D7E2}  
 1746 \usv\_set:nnn {sfup}{num} { "1D7E2}  
 1747 \usv\_set:nnn {sfit}{num} { "1D7E2}  
 1748 \usv\_set:nnn {sfup}{Latin}{ "1D5A0}  
 1749 \usv\_set:nnn {sf} {Latin}{ "1D5A0}  
 1750 \usv\_set:nnn {sfup}{latin}{ "1D5BA}  
 1751 \usv\_set:nnn {sf} {latin}{ "1D5BA}  
 1752 \usv\_set:nnn {sfit}{Latin}{ "1D608}  
 1753 \usv\_set:nnn {sfit}{latin}{ "1D622}  
 1754 \usv\_set:nnn {tt} {num} { "1D7F6}  
 1755 \usv\_set:nnn {tt} {Latin}{ "1D670}  
 1756 \usv\_set:nnn {tt} {latin}{ "1D68A}

#### **Bold:**

1757 \usv\_set:nnn {bf} {num} { "1D7CE}  
 1758 \usv\_set:nnn {bfup} {num} { "1D7CE}  
 1759 \usv\_set:nnn {bfit} {num} { "1D7CE}  
 1760 \usv\_set:nnn {bfup} {Latin}{ "1D400}  
 1761 \usv\_set:nnn {bfup} {latin}{ "1D41A}  
 1762 \usv\_set:nnn {bfup} {Greek}{ "1D6A8}  
 1763 \usv\_set:nnn {bfup} {greek}{ "1D6C2}  
 1764 \usv\_set:nnn {bfit} {Latin}{ "1D468}  
 1765 \usv\_set:nnn {bfit} {latin}{ "1D482}  
 1766 \usv\_set:nnn {bfit} {Greek}{ "1D71C}  
 1767 \usv\_set:nnn {bfit} {greek}{ "1D736}  
 1768 \usv\_set:nnn {bffrak}{Latin}{ "1D56C}  
 1769 \usv\_set:nnn {bffrak}{latin}{ "1D586}  
 1770 \usv\_set:nnn {bfscr} {Latin}{ "1D4D0}  
 1771 \usv\_set:nnn {bfcal} {Latin}{ "1D4D0}  
 1772 \usv\_set:nnn {bfscr} {latin}{ "1D4EA}  
 1773 \usv\_set:nnn {bfsf} {num} { "1D7EC}  
 1774 \usv\_set:nnn {bfsfup}{num} { "1D7EC}  
 1775 \usv\_set:nnn {bfsfit}{num} { "1D7EC}  
 1776 \usv\_set:nnn {bfsfup}{Latin}{ "1D5D4}  
 1777 \usv\_set:nnn {bfsfup}{latin}{ "1D5EE}  
 1778 \usv\_set:nnn {bfsfup}{Greek}{ "1D756}  
 1779 \usv\_set:nnn {bfsfup}{greek}{ "1D770}  
 1780 \usv\_set:nnn {bfsfit}{Latin}{ "1D63C}  
 1781 \usv\_set:nnn {bfsfit}{latin}{ "1D656}  
 1782 \usv\_set:nnn {bfsfit}{Greek}{ "1D790}  
 1783 \usv\_set:nnn {bfsfit}{greek}{ "1D7AA}

1784 \usv\_set:nnn {bfsf}{Latin}{ \bool\_if:NTF \g\_@@\_upLatin\_bool \g\_@@\_bfsfup\_Latin\_usv \g\_@@\_bfsfit\_Lat  
1785 \usv\_set:nnn {bfsf}{latin}{ \bool\_if:NTF \g\_@@\_uplatin\_bool \g\_@@\_bfsfup\_latin\_usv \g\_@@\_bfsfit\_lat  
1786 \usv\_set:nnn {bfsf}{Greek}{ \bool\_if:NTF \g\_@@\_upGreek\_bool \g\_@@\_bfsfup\_Greek\_usv \g\_@@\_bfsfit\_Gre  
1787 \usv\_set:nnn {bfsf}{greek}{ \bool\_if:NTF \g\_@@\_upgreek\_bool \g\_@@\_bfsfup\_greek\_usv \g\_@@\_bfsfit\_gre  
1788 \usv\_set:nnn {bf} {Latin}{ \bool\_if:NTF \g\_@@\_bfupLatin\_bool \g\_@@\_bfup\_Latin\_usv \g\_@@\_bfit\_Latin  
1789 \usv\_set:nnn {bf} {latin}{ \bool\_if:NTF \g\_@@\_bfuplatin\_bool \g\_@@\_bfup\_latin\_usv \g\_@@\_bfit\_latin  
1790 \usv\_set:nnn {bf} {Greek}{ \bool\_if:NTF \g\_@@\_bfupGreek\_bool \g\_@@\_bfup\_Greek\_usv \g\_@@\_bfit\_Greek  
1791 \usv\_set:nnn {bf} {greek}{ \bool\_if:NTF \g\_@@\_bfupgreek\_bool \g\_@@\_bfup\_greek\_usv \g\_@@\_bfit\_greek

#### Greek variants:

1792 \usv\_set:nnn {up}{varTheta} {"3F4}  
1793 \usv\_set:nnn {up}{Digamma} {"3DC}  
1794 \usv\_set:nnn {up}{epsilon}{"3F5}  
1795 \usv\_set:nnn {up}{vartheta} {"3D1}  
1796 \usv\_set:nnn {up}{varkappa} {"3F0}  
1797 \usv\_set:nnn {up}{phi} {"3D5}  
1798 \usv\_set:nnn {up}{varrho} {"3F1}  
1799 \usv\_set:nnn {up}{varpi} {"3D6}  
1800 \usv\_set:nnn {up}{digamma} {"3DD}

#### Bold:

1801 \usv\_set:nnn {bfup}{varTheta} {"1D6B9}  
1802 \usv\_set:nnn {bfup}{Digamma} {"1D7CA}  
1803 \usv\_set:nnn {bfup}{epsilon}{"1D6DC}  
1804 \usv\_set:nnn {bfup}{vartheta} {"1D6DD}  
1805 \usv\_set:nnn {bfup}{varkappa} {"1D6DE}  
1806 \usv\_set:nnn {bfup}{phi} {"1D6DF}  
1807 \usv\_set:nnn {bfup}{varrho} {"1D6E0}  
1808 \usv\_set:nnn {bfup}{varpi} {"1D6E1}  
1809 \usv\_set:nnn {bfup}{digamma} {"1D7CB}

#### Italic Greek variants:

1810 \usv\_set:nnn {it}{varTheta} {"1D6F3}  
1811 \usv\_set:nnn {it}{epsilon}{"1D716}  
1812 \usv\_set:nnn {it}{vartheta} {"1D717}  
1813 \usv\_set:nnn {it}{varkappa} {"1D718}  
1814 \usv\_set:nnn {it}{phi} {"1D719}  
1815 \usv\_set:nnn {it}{varrho} {"1D71A}  
1816 \usv\_set:nnn {it}{varpi} {"1D71B}

#### Bold italic:

1817 \usv\_set:nnn {bfit}{varTheta} {"1D72D}  
1818 \usv\_set:nnn {bfit}{epsilon}{"1D750}  
1819 \usv\_set:nnn {bfit}{vartheta} {"1D751}  
1820 \usv\_set:nnn {bfit}{varkappa} {"1D752}  
1821 \usv\_set:nnn {bfit}{phi} {"1D753}  
1822 \usv\_set:nnn {bfit}{varrho} {"1D754}  
1823 \usv\_set:nnn {bfit}{varpi} {"1D755}

#### Bold sans:

1824 \usv\_set:nnn {bfsfup}{varTheta} {"1D767}

1825 \usv\_set:nnn {bfsfup}{epsilon}{ "1D78A}  
 1826 \usv\_set:nnn {bfsfup}{vartheta} { "1D78B}  
 1827 \usv\_set:nnn {bfsfup}{varkappa} { "1D78C}  
 1828 \usv\_set:nnn {bfsfup}{phi} { "1D78D}  
 1829 \usv\_set:nnn {bfsfup}{varrho} { "1D78E}  
 1830 \usv\_set:nnn {bfsfup}{varpi} { "1D78F}

**Bold sans italic:**

1831 \usv\_set:nnn {bfsfit}{varTheta} { "1D7A1}  
 1832 \usv\_set:nnn {bfsfit}{epsilon}{ "1D7C4}  
 1833 \usv\_set:nnn {bfsfit}{vartheta} { "1D7C5}  
 1834 \usv\_set:nnn {bfsfit}{varkappa} { "1D7C6}  
 1835 \usv\_set:nnn {bfsfit}{phi} { "1D7C7}  
 1836 \usv\_set:nnn {bfsfit}{varrho} { "1D7C8}  
 1837 \usv\_set:nnn {bfsfit}{varpi} { "1D7C9}

**Nabla:**

1838 \usv\_set:nnn {up} {Nabla}{ "02207}  
 1839 \usv\_set:nnn {it} {Nabla}{ "1D6FB}  
 1840 \usv\_set:nnn {bfup} {Nabla}{ "1D6C1}  
 1841 \usv\_set:nnn {bfit} {Nabla}{ "1D735}  
 1842 \usv\_set:nnn {bfsfup}{Nabla}{ "1D76F}  
 1843 \usv\_set:nnn {bfsfit}{Nabla}{ "1D7A9}

**Partial:**

1844 \usv\_set:nnn {up} {partial}{ "02202}  
 1845 \usv\_set:nnn {it} {partial}{ "1D715}  
 1846 \usv\_set:nnn {bfup} {partial}{ "1D6DB}  
 1847 \usv\_set:nnn {bfit} {partial}{ "1D74F}  
 1848 \usv\_set:nnn {bfsfup}{partial}{ "1D789}  
 1849 \usv\_set:nnn {bfsfit}{partial}{ "1D7C3}

*Exceptions* These are need for mapping with the exceptions in other alphabets:  
 (coming up)

1850 \usv\_set:nnn {up}{B}{`\B}  
 1851 \usv\_set:nnn {up}{C}{`\C}  
 1852 \usv\_set:nnn {up}{D}{`\D}  
 1853 \usv\_set:nnn {up}{E}{`\E}  
 1854 \usv\_set:nnn {up}{F}{`\F}  
 1855 \usv\_set:nnn {up}{H}{`\H}  
 1856 \usv\_set:nnn {up}{I}{`\I}  
 1857 \usv\_set:nnn {up}{L}{`\L}  
 1858 \usv\_set:nnn {up}{M}{`\M}  
 1859 \usv\_set:nnn {up}{N}{`\N}  
 1860 \usv\_set:nnn {up}{P}{`\P}  
 1861 \usv\_set:nnn {up}{Q}{`\Q}  
 1862 \usv\_set:nnn {up}{R}{`\R}  
 1863 \usv\_set:nnn {up}{Z}{`\Z}  
 1864 \usv\_set:nnn {it}{B}{ "1D435}  
 1865 \usv\_set:nnn {it}{C}{ "1D436}

1866 \usv\_set:nnn {it}{D}{ "1D437}  
 1867 \usv\_set:nnn {it}{E}{ "1D438}  
 1868 \usv\_set:nnn {it}{F}{ "1D439}  
 1869 \usv\_set:nnn {it}{H}{ "1D43B}  
 1870 \usv\_set:nnn {it}{I}{ "1D43C}  
 1871 \usv\_set:nnn {it}{L}{ "1D43F}  
 1872 \usv\_set:nnn {it}{M}{ "1D440}  
 1873 \usv\_set:nnn {it}{N}{ "1D441}  
 1874 \usv\_set:nnn {it}{P}{ "1D443}  
 1875 \usv\_set:nnn {it}{Q}{ "1D444}  
 1876 \usv\_set:nnn {it}{R}{ "1D445}  
 1877 \usv\_set:nnn {it}{Z}{ "1D44D}  
  
 1878 \usv\_set:nnn {up}{d}{ `d}  
 1879 \usv\_set:nnn {up}{e}{ `e}  
 1880 \usv\_set:nnn {up}{g}{ `g}  
 1881 \usv\_set:nnn {up}{h}{ `h}  
 1882 \usv\_set:nnn {up}{i}{ `i}  
 1883 \usv\_set:nnn {up}{j}{ `j}  
 1884 \usv\_set:nnn {up}{o}{ `o}  
  
 1885 \usv\_set:nnn {it}{d}{ "1D451}  
 1886 \usv\_set:nnn {it}{e}{ "1D452}  
 1887 \usv\_set:nnn {it}{g}{ "1D454}  
 1888 \usv\_set:nnn {it}{h}{ "0210E}  
 1889 \usv\_set:nnn {it}{i}{ "1D456}  
 1890 \usv\_set:nnn {it}{j}{ "1D457}  
 1891 \usv\_set:nnn {it}{o}{ "1D45C}

#### Latin ‘h’:

1892 \usv\_set:nnn {bb} {h}{ "1D559}  
 1893 \usv\_set:nnn {tt} {h}{ "1D691}  
 1894 \usv\_set:nnn {scr} {h}{ "1D4BD}  
 1895 \usv\_set:nnn {frak} {h}{ "1D525}  
 1896 \usv\_set:nnn {bfup} {h}{ "1D421}  
 1897 \usv\_set:nnn {bfit} {h}{ "1D489}  
 1898 \usv\_set:nnn {sfup} {h}{ "1D5C1}  
 1899 \usv\_set:nnn {sfit} {h}{ "1D629}  
 1900 \usv\_set:nnn {bffrak}{h}{ "1D58D}  
 1901 \usv\_set:nnn {bfscr} {h}{ "1D4F1}  
 1902 \usv\_set:nnn {bfsfup}{h}{ "1D5F5}  
 1903 \usv\_set:nnn {bfsfit}{h}{ "1D65D}

#### Dotless ‘i’ and ‘j’:

1904 \usv\_set:nnn {up}{dotlessi}{ "00131}  
 1905 \usv\_set:nnn {up}{dotlessj}{ "00237}  
 1906 \usv\_set:nnn {it}{dotlessi}{ "1D6A4}  
 1907 \usv\_set:nnn {it}{dotlessj}{ "1D6A5}

#### Blackboard:

1908 \usv\_set:nnn {bb}{C}{ "2102}  
 1909 \usv\_set:nnn {bb}{H}{ "210D}

1910 \usv\_set:nnn {bb}{N}{ "2115}  
 1911 \usv\_set:nnn {bb}{P}{ "2119}  
 1912 \usv\_set:nnn {bb}{Q}{ "211A}  
 1913 \usv\_set:nnn {bb}{R}{ "211D}  
 1914 \usv\_set:nnn {bb}{Z}{ "2124}  
 1915 \usv\_set:nnn {up}{Pi} { "003A0}  
 1916 \usv\_set:nnn {up}{pi} { "003C0}  
 1917 \usv\_set:nnn {up}{Gamma} { "00393}  
 1918 \usv\_set:nnn {up}{gamma} { "003B3}  
 1919 \usv\_set:nnn {up}{summation}{ "02211}  
 1920 \usv\_set:nnn {it}{Pi} { "1D6F1}  
 1921 \usv\_set:nnn {it}{pi} { "1D70B}  
 1922 \usv\_set:nnn {it}{Gamma} { "1D6E4}  
 1923 \usv\_set:nnn {it}{gamma} { "1D6FE}  
 1924 \usv\_set:nnn {bb}{Pi} { "0213F}  
 1925 \usv\_set:nnn {bb}{pi} { "0213C}  
 1926 \usv\_set:nnn {bb}{Gamma} { "0213E}  
 1927 \usv\_set:nnn {bb}{gamma} { "0213D}  
 1928 \usv\_set:nnn {bb}{summation}{ "02140}

Italic blackboard:

1929 \usv\_set:nnn {bbit}{D}{ "2145}  
 1930 \usv\_set:nnn {bbit}{d}{ "2146}  
 1931 \usv\_set:nnn {bbit}{e}{ "2147}  
 1932 \usv\_set:nnn {bbit}{i}{ "2148}  
 1933 \usv\_set:nnn {bbit}{j}{ "2149}

Script exceptions:

1934 \usv\_set:nnn {scr}{B}{ "212C}  
 1935 \usv\_set:nnn {scr}{E}{ "2130}  
 1936 \usv\_set:nnn {scr}{F}{ "2131}  
 1937 \usv\_set:nnn {scr}{H}{ "210B}  
 1938 \usv\_set:nnn {scr}{I}{ "2110}  
 1939 \usv\_set:nnn {scr}{L}{ "2112}  
 1940 \usv\_set:nnn {scr}{M}{ "2133}  
 1941 \usv\_set:nnn {scr}{R}{ "211B}  
 1942 \usv\_set:nnn {scr}{e}{ "212F}  
 1943 \usv\_set:nnn {scr}{g}{ "210A}  
 1944 \usv\_set:nnn {scr}{o}{ "2134}  
 1945 \usv\_set:nnn {cal}{B}{ "212C}  
 1946 \usv\_set:nnn {cal}{E}{ "2130}  
 1947 \usv\_set:nnn {cal}{F}{ "2131}  
 1948 \usv\_set:nnn {cal}{H}{ "210B}  
 1949 \usv\_set:nnn {cal}{I}{ "2110}  
 1950 \usv\_set:nnn {cal}{L}{ "2112}  
 1951 \usv\_set:nnn {cal}{M}{ "2133}  
 1952 \usv\_set:nnn {cal}{R}{ "211B}

Fraktur exceptions:

1953 \usv\_set:nnn {frak}{C}{ "212D}

```

1954 \usv_set:nnn {\frak}{H}{\ "210C}
1955 \usv_set:nnn {\frak}{I}{\ "2111}
1956 \usv_set:nnn {\frak}{R}{\ "211C}
1957 \usv_set:nnn {\frak}{Z}{\ "2128}
1958 <*\usv>

```

## L.2 STIX fonts

Version 1.0.0 of the STIX fonts contains a number of alphabets in the private use area of Unicode; i.e., it contains many math glyphs that have not (yet or if ever) been accepted into the Unicode standard.

But we still want to be able to use them if possible.

```

1959 <*\stix>

```

### *Upright*

```

1960 \usv_set:nnn {\stixsfup}{partial}{\ "E17C}
1961 \usv_set:nnn {\stixsfup}{Greek}{\ "E17D}
1962 \usv_set:nnn {\stixsfup}{greek}{\ "E196}
1963 \usv_set:nnn {\stixsfup}{varTheta}{\ "E18E}
1964 \usv_set:nnn {\stixsfup}{epsilon}{\ "E1AF}
1965 \usv_set:nnn {\stixsfup}{vartheta}{\ "E1B0}
1966 \usv_set:nnn {\stixsfup}{varkappa}{\ "0000} % ???
1967 \usv_set:nnn {\stixsfup}{phi}{\ "E1B1}
1968 \usv_set:nnn {\stixsfup}{varrho}{\ "E1B2}
1969 \usv_set:nnn {\stixsfup}{varpi}{\ "E1B3}
1970 \usv_set:nnn {\stixupslash}{Greek}{\ "E2FC}

```

### *Italic*

```

1971 \usv_set:nnn {\stixbbit}{A}{\ "E154}
1972 \usv_set:nnn {\stixbbit}{B}{\ "E155}
1973 \usv_set:nnn {\stixbbit}{E}{\ "E156}
1974 \usv_set:nnn {\stixbbit}{F}{\ "E157}
1975 \usv_set:nnn {\stixbbit}{G}{\ "E158}
1976 \usv_set:nnn {\stixbbit}{I}{\ "E159}
1977 \usv_set:nnn {\stixbbit}{J}{\ "E15A}
1978 \usv_set:nnn {\stixbbit}{K}{\ "E15B}
1979 \usv_set:nnn {\stixbbit}{L}{\ "E15C}
1980 \usv_set:nnn {\stixbbit}{M}{\ "E15D}
1981 \usv_set:nnn {\stixbbit}{O}{\ "E15E}
1982 \usv_set:nnn {\stixbbit}{S}{\ "E15F}
1983 \usv_set:nnn {\stixbbit}{T}{\ "E160}
1984 \usv_set:nnn {\stixbbit}{U}{\ "E161}
1985 \usv_set:nnn {\stixbbit}{V}{\ "E162}
1986 \usv_set:nnn {\stixbbit}{W}{\ "E163}
1987 \usv_set:nnn {\stixbbit}{X}{\ "E164}
1988 \usv_set:nnn {\stixbbit}{Y}{\ "E165}

```



1989 \usv\_set:nnn {stixbbit}{a}{ "E166}  
 1990 \usv\_set:nnn {stixbbit}{b}{ "E167}  
 1991 \usv\_set:nnn {stixbbit}{c}{ "E168}  
 1992 \usv\_set:nnn {stixbbit}{f}{ "E169}  
 1993 \usv\_set:nnn {stixbbit}{g}{ "E16A}  
 1994 \usv\_set:nnn {stixbbit}{h}{ "E16B}  
 1995 \usv\_set:nnn {stixbbit}{k}{ "E16C}  
 1996 \usv\_set:nnn {stixbbit}{l}{ "E16D}  
 1997 \usv\_set:nnn {stixbbit}{m}{ "E16E}  
 1998 \usv\_set:nnn {stixbbit}{n}{ "E16F}  
 1999 \usv\_set:nnn {stixbbit}{o}{ "E170}  
 2000 \usv\_set:nnn {stixbbit}{p}{ "E171}  
 2001 \usv\_set:nnn {stixbbit}{q}{ "E172}  
 2002 \usv\_set:nnn {stixbbit}{r}{ "E173}  
 2003 \usv\_set:nnn {stixbbit}{s}{ "E174}  
 2004 \usv\_set:nnn {stixbbit}{t}{ "E175}  
 2005 \usv\_set:nnn {stixbbit}{u}{ "E176}  
 2006 \usv\_set:nnn {stixbbit}{v}{ "E177}  
 2007 \usv\_set:nnn {stixbbit}{w}{ "E178}  
 2008 \usv\_set:nnn {stixbbit}{x}{ "E179}  
 2009 \usv\_set:nnn {stixbbit}{y}{ "E17A}  
 2010 \usv\_set:nnn {stixbbit}{z}{ "E17B}  
  
 2011 \usv\_set:nnn {stixsfit}{Numerals}{ "E1B4}  
 2012 \usv\_set:nnn {stixsfit}{partial}{ "E1BE}  
 2013 \usv\_set:nnn {stixsfit}{Greek}{ "E1BF}  
 2014 \usv\_set:nnn {stixsfit}{greek}{ "E1D8}  
 2015 \usv\_set:nnn {stixsfit}{varTheta}{ "E1D0}  
 2016 \usv\_set:nnn {stixsfit}{epsilon}{ "E1F1}  
 2017 \usv\_set:nnn {stixsfit}{vartheta}{ "E1F2}  
 2018 \usv\_set:nnn {stixsfit}{varkappa}{0000} % ???  
 2019 \usv\_set:nnn {stixsfit}{phi}{ "E1F3}  
 2020 \usv\_set:nnn {stixsfit}{varrho}{ "E1F4}  
 2021 \usv\_set:nnn {stixsfit}{varpi}{ "E1F5}  
  
 2022 \usv\_set:nnn {stixcal}{Latin}{ "E22D}  
 2023 \usv\_set:nnn {stixcal}{num}{ "E262}  
 2024 \usv\_set:nnn {scr}{num}{48}  
 2025 \usv\_set:nnn {it}{num}{48}  
  
 2026 \usv\_set:nnn {stixsfitslash}{Latin}{ "E294}  
 2027 \usv\_set:nnn {stixsfitslash}{latin}{ "E2C8}  
 2028 \usv\_set:nnn {stixsfitslash}{greek}{ "E32C}  
 2029 \usv\_set:nnn {stixsfitslash}{epsilon}{ "E37A}  
 2030 \usv\_set:nnn {stixsfitslash}{vartheta}{ "E35E}  
 2031 \usv\_set:nnn {stixsfitslash}{varkappa}{ "E374}  
 2032 \usv\_set:nnn {stixsfitslash}{phi}{ "E360}  
 2033 \usv\_set:nnn {stixsfitslash}{varrho}{ "E376}  
 2034 \usv\_set:nnn {stixsfitslash}{varpi}{ "E362}  
 2035 \usv\_set:nnn {stixsfitslash}{digamma}{ "E36A}

## *Bold*

2036 \usv\_set:nnn {stixbfupslash}{Greek}{ $\epsilon$ }

2037 \usv\_set:nnn {stixbfupslash}{Digamma}{ $\epsilon$ }

2038 \usv\_set:nnn {stixbfbb}{A}{ $\epsilon$ }

2039 \usv\_set:nnn {stixbfbb}{B}{ $\epsilon$ }

2040 \usv\_set:nnn {stixbfbb}{E}{ $\epsilon$ }

2041 \usv\_set:nnn {stixbfbb}{F}{ $\epsilon$ }

2042 \usv\_set:nnn {stixbfbb}{G}{ $\epsilon$ }

2043 \usv\_set:nnn {stixbfbb}{I}{ $\epsilon$ }

2044 \usv\_set:nnn {stixbfbb}{J}{ $\epsilon$ }

2045 \usv\_set:nnn {stixbfbb}{K}{ $\epsilon$ }

2046 \usv\_set:nnn {stixbfbb}{L}{ $\epsilon$ }

2047 \usv\_set:nnn {stixbfbb}{M}{ $\epsilon$ }

2048 \usv\_set:nnn {stixbfbb}{O}{ $\epsilon$ }

2049 \usv\_set:nnn {stixbfbb}{S}{ $\epsilon$ }

2050 \usv\_set:nnn {stixbfbb}{T}{ $\epsilon$ }

2051 \usv\_set:nnn {stixbfbb}{U}{ $\epsilon$ }

2052 \usv\_set:nnn {stixbfbb}{V}{ $\epsilon$ }

2053 \usv\_set:nnn {stixbfbb}{W}{ $\epsilon$ }

2054 \usv\_set:nnn {stixbfbb}{X}{ $\epsilon$ }

2055 \usv\_set:nnn {stixbfbb}{Y}{ $\epsilon$ }

2056 \usv\_set:nnn {stixbfbb}{a}{ $\epsilon$ }

2057 \usv\_set:nnn {stixbfbb}{b}{ $\epsilon$ }

2058 \usv\_set:nnn {stixbfbb}{c}{ $\epsilon$ }

2059 \usv\_set:nnn {stixbfbb}{f}{ $\epsilon$ }

2060 \usv\_set:nnn {stixbfbb}{g}{ $\epsilon$ }

2061 \usv\_set:nnn {stixbfbb}{h}{ $\epsilon$ }

2062 \usv\_set:nnn {stixbfbb}{k}{ $\epsilon$ }

2063 \usv\_set:nnn {stixbfbb}{l}{ $\epsilon$ }

2064 \usv\_set:nnn {stixbfbb}{m}{ $\epsilon$ }

2065 \usv\_set:nnn {stixbfbb}{n}{ $\epsilon$ }

2066 \usv\_set:nnn {stixbfbb}{o}{ $\epsilon$ }

2067 \usv\_set:nnn {stixbfbb}{p}{ $\epsilon$ }

2068 \usv\_set:nnn {stixbfbb}{q}{ $\epsilon$ }

2069 \usv\_set:nnn {stixbfbb}{r}{ $\epsilon$ }

2070 \usv\_set:nnn {stixbfbb}{s}{ $\epsilon$ }

2071 \usv\_set:nnn {stixbfbb}{t}{ $\epsilon$ }

2072 \usv\_set:nnn {stixbfbb}{u}{ $\epsilon$ }

2073 \usv\_set:nnn {stixbfbb}{v}{ $\epsilon$ }

2074 \usv\_set:nnn {stixbfbb}{w}{ $\epsilon$ }

2075 \usv\_set:nnn {stixbfbb}{x}{ $\epsilon$ }

2076 \usv\_set:nnn {stixbfbb}{y}{ $\epsilon$ }

2077 \usv\_set:nnn {stixbfbb}{z}{ $\epsilon$ }

2078 \usv\_set:nnn {stixbfsfup}{Numerals}{ $\epsilon$ }

## *Bold Italic*

2079 \usv\_set:nnn {stixbfsfit}{Numerals}{ $\epsilon$ }

2080 \usv\_set:nnn {stixbfbbbit}{A}{ "E200}  
 2081 \usv\_set:nnn {stixbfbbbit}{B}{ "E201}  
 2082 \usv\_set:nnn {stixbfbbbit}{E}{ "E203}  
 2083 \usv\_set:nnn {stixbfbbbit}{F}{ "E204}  
 2084 \usv\_set:nnn {stixbfbbbit}{G}{ "E205}  
 2085 \usv\_set:nnn {stixbfbbbit}{I}{ "E206}  
 2086 \usv\_set:nnn {stixbfbbbit}{J}{ "E207}  
 2087 \usv\_set:nnn {stixbfbbbit}{K}{ "E208}  
 2088 \usv\_set:nnn {stixbfbbbit}{L}{ "E209}  
 2089 \usv\_set:nnn {stixbfbbbit}{M}{ "E20A}  
 2090 \usv\_set:nnn {stixbfbbbit}{O}{ "E20B}  
 2091 \usv\_set:nnn {stixbfbbbit}{S}{ "E20C}  
 2092 \usv\_set:nnn {stixbfbbbit}{T}{ "E20D}  
 2093 \usv\_set:nnn {stixbfbbbit}{U}{ "E20E}  
 2094 \usv\_set:nnn {stixbfbbbit}{V}{ "E20F}  
 2095 \usv\_set:nnn {stixbfbbbit}{W}{ "E210}  
 2096 \usv\_set:nnn {stixbfbbbit}{X}{ "E211}  
 2097 \usv\_set:nnn {stixbfbbbit}{Y}{ "E212}  
  
 2098 \usv\_set:nnn {stixbfbbbit}{a}{ "E213}  
 2099 \usv\_set:nnn {stixbfbbbit}{b}{ "E214}  
 2100 \usv\_set:nnn {stixbfbbbit}{c}{ "E215}  
 2101 \usv\_set:nnn {stixbfbbbit}{e}{ "E217}  
 2102 \usv\_set:nnn {stixbfbbbit}{f}{ "E218}  
 2103 \usv\_set:nnn {stixbfbbbit}{g}{ "E219}  
 2104 \usv\_set:nnn {stixbfbbbit}{h}{ "E21A}  
 2105 \usv\_set:nnn {stixbfbbbit}{k}{ "E21D}  
 2106 \usv\_set:nnn {stixbfbbbit}{l}{ "E21E}  
 2107 \usv\_set:nnn {stixbfbbbit}{m}{ "E21F}  
 2108 \usv\_set:nnn {stixbfbbbit}{n}{ "E220}  
 2109 \usv\_set:nnn {stixbfbbbit}{o}{ "E221}  
 2110 \usv\_set:nnn {stixbfbbbit}{p}{ "E222}  
 2111 \usv\_set:nnn {stixbfbbbit}{q}{ "E223}  
 2112 \usv\_set:nnn {stixbfbbbit}{r}{ "E224}  
 2113 \usv\_set:nnn {stixbfbbbit}{s}{ "E225}  
 2114 \usv\_set:nnn {stixbfbbbit}{t}{ "E226}  
 2115 \usv\_set:nnn {stixbfbbbit}{u}{ "E227}  
 2116 \usv\_set:nnn {stixbfbbbit}{v}{ "E228}  
 2117 \usv\_set:nnn {stixbfbbbit}{w}{ "E229}  
 2118 \usv\_set:nnn {stixbfbbbit}{x}{ "E22A}  
 2119 \usv\_set:nnn {stixbfbbbit}{y}{ "E22B}  
 2120 \usv\_set:nnn {stixbfbbbit}{z}{ "E22C}  
  
 2121 \usv\_set:nnn {stixbfcal}{Latin}{ "E247}  
  
 2122 \usv\_set:nnn {stixbfitslash}{Latin}{ "E295}  
 2123 \usv\_set:nnn {stixbfitslash}{latin}{ "E2C9}  
 2124 \usv\_set:nnn {stixbfitslash}{greek}{ "E32D}  
 2125 \usv\_set:nnn {stixsfitslash}{epsilon}{ "E37B}  
 2126 \usv\_set:nnn {stixsfitslash}{vartheta}{ "E35F}  
 2127 \usv\_set:nnn {stixsfitslash}{varkappa}{ "E375}

```

2128 \usv_set:nnn {stixsfitslash}{phi}{E361}
2129 \usv_set:nnn {stixsfitslash}{varrho}{E377}
2130 \usv_set:nnn {stixsfitslash}{varpi}{E363}
2131 \usv_set:nnn {stixsfitslash}{digamma}{E36B}
2132 </stix>

```

## L.3 Alphabets

```
2133 <*alphabets>
```

### L.3.1 Upright: up

```

2134 \@@_new_alphabet_config:nnn {up} {num}
2135 {
2136   \@@_set_normal_numbers:nn {up} {#1}
2137   \@@_set_mathalphabet_numbers:nnn {up} {up} {#1}
2138 }
2139
2140 \@@_new_alphabet_config:nnn {up} {Latin}
2141 {
2142   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {up} {#1} }
2143   {
2144     \bool_if:NT \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2145   }
2146   \@@_set_mathalphabet_Latin:nnn {up} {up,it} {#1}
2147   \@@_set_mathalphabet_Latin:nnn {literal} {up} {up}
2148   \@@_set_mathalphabet_Latin:nnn {literal} {it} {it}
2149 }
2150
2151 \@@_new_alphabet_config:nnn {up} {latin}
2152 {
2153   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_latin:nn {up} {#1} }
2154   {
2155     \bool_if:NT \g_@@_uplatin_bool
2156     {
2157       \@@_set_normal_latin:nn {up,it} {#1}
2158       \@@_set_normal_char:nnn {h} {up,it} {#1}
2159       \@@_set_normal_char:nnn {dotlessi} {up,it} {#1}
2160       \@@_set_normal_char:nnn {dotlessj} {up,it} {#1}
2161     }
2162   }
2163   \@@_set_mathalphabet_latin:nnn {up} {up,it}{#1}
2164   \@@_set_mathalphabet_latin:nnn {literal} {up} {up}
2165   \@@_set_mathalphabet_latin:nnn {literal} {it} {it}
2166 }
2167
2168 \@@_new_alphabet_config:nnn {up} {Greek}
2169 {
2170   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Greek:nn {up}{#1} }
2171   {

```

```

2172 \bool_if:NT \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2173 }
2174 \@@_set_mathalphabet_Greek:nnn {up} {up,it}{#1}
2175 \@@_set_mathalphabet_Greek:nnn {literal} {up} {up}
2176 \@@_set_mathalphabet_Greek:nnn {literal} {it} {it}
2177 }
2178
2179 \@@_new_alphabet_config:nnn {up} {greek}
2180 {
2181 \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_greek:nn {up} {#1} }
2182 {
2183 \bool_if:NT \g_@@_upgreek_bool
2184 {
2185 \@@_set_normal_greek:nn {up,it} {#1}
2186 }
2187 }
2188 \@@_set_mathalphabet_greek:nnn {up} {up,it} {#1}
2189 \@@_set_mathalphabet_greek:nnn {literal} {up} {up}
2190 \@@_set_mathalphabet_greek:nnn {literal} {it} {it}
2191 }
2192
2193 \@@_new_alphabet_config:nnn {up} {misc}
2194 {
2195 \bool_if:NTF \g_@@_literal_Nabla_bool
2196 {
2197 \@@_set_normal_char:nnn {Nabla}{up}{up}
2198 }
2199 {
2200 \bool_if:NT \g_@@_upNabla_bool
2201 {
2202 \@@_set_normal_char:nnn {Nabla}{up,it}{up}
2203 }
2204 }
2205 \bool_if:NTF \g_@@_literal_partial_bool
2206 {
2207 \@@_set_normal_char:nnn {partial}{up}{up}
2208 }
2209 {
2210 \bool_if:NT \g_@@_uppartial_bool
2211 {
2212 \@@_set_normal_char:nnn {partial}{up,it}{up}
2213 }
2214 }
2215 \@@_set_mathalphabet_pos:nnnn {up} {partial} {up,it} {#1}
2216 \@@_set_mathalphabet_pos:nnnn {up} {Nabla} {up,it} {#1}
2217 \@@_set_mathalphabet_pos:nnnn {up} {dotlessi} {up,it} {#1}
2218 \@@_set_mathalphabet_pos:nnnn {up} {dotlessj} {up,it} {#1}
2219 }

```

### L.3.2 *Italic: it*

```
2220 \@@_new_alphabet_config:nnn {it} {Latin}
2221 {
2222   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {it} {#1} }
2223   {
2224     \bool_if:NF \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2225   }
2226   \@@_set_mathalphabet_Latin:nnn {it}{up,it}{#1}
2227 }
2228
2229 \@@_new_alphabet_config:nnn {it} {latin}
2230 {
2231   \bool_if:NTF \g_@@_literal_bool
2232   {
2233     \@@_set_normal_latin:nn {it} {#1}
2234     \@@_set_normal_char:nnn {h}{it}{#1}
2235   }
2236   {
2237     \bool_if:NF \g_@@_uplatin_bool
2238     {
2239       \@@_set_normal_latin:nn {up,it} {#1}
2240       \@@_set_normal_char:nnn {h}{up,it}{#1}
2241       \@@_set_normal_char:nnn {dotlessi}{up,it}{#1}
2242       \@@_set_normal_char:nnn {dotlessj}{up,it}{#1}
2243     }
2244   }
2245   \@@_set_mathalphabet_latin:nnn {it} {up,it} {#1}
2246   \@@_set_mathalphabet_pos:nnnn {it} {dotlessi} {up,it} {#1}
2247   \@@_set_mathalphabet_pos:nnnn {it} {dotlessj} {up,it} {#1}
2248 }
2249
2250 \@@_new_alphabet_config:nnn {it} {Greek}
2251 {
2252   \bool_if:NTF \g_@@_literal_bool
2253   {
2254     \@@_set_normal_Greek:nn {it}{#1}
2255   }
2256   {
2257     \bool_if:NF \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2258   }
2259   \@@_set_mathalphabet_Greek:nnn {it} {up,it}{#1}
2260 }
2261
2262 \@@_new_alphabet_config:nnn {it} {greek}
2263 {
2264   \bool_if:NTF \g_@@_literal_bool
2265   {
2266     \@@_set_normal_greek:nn {it} {#1}
2267   }
```

```

2268 {
2269   \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_greek:nn {it,up} {#1} }
2270 }
2271 \@@_set_mathalphabet_greek:nnn {it} {up,it} {#1}
2272 }
2273
2274 \@@_new_alphabet_config:nnn {it} {misc}
2275 {
2276   \bool_if:NTF \g_@@_literal_Nabla_bool
2277   {
2278     \@@_set_normal_char:nnn {Nabla}{it}{it}
2279   }
2280   {
2281     \bool_if:NF \g_@@_upNabla_bool
2282     {
2283       \@@_set_normal_char:nnn {Nabla}{up,it}{it}
2284     }
2285   }
2286   \bool_if:NTF \g_@@_literal_partial_bool
2287   {
2288     \@@_set_normal_char:nnn {partial}{it}{it}
2289   }
2290   {
2291     \bool_if:NF \g_@@_uppartial_bool
2292     {
2293       \@@_set_normal_char:nnn {partial}{up,it}{it}
2294     }
2295   }
2296   \@@_set_mathalphabet_pos:nnnn {it} {partial} {up,it}{#1}
2297   \@@_set_mathalphabet_pos:nnnn {it} {Nabla} {up,it}{#1}
2298 }

```

### L.3.3 Blackboard or double-struck: *bb* and *bbit*

```

2299 \@@_new_alphabet_config:nnn {bb} {latin}
2300 {
2301   \@@_set_mathalphabet_latin:nnn {bb} {up,it}{#1}
2302 }
2303
2304 \@@_new_alphabet_config:nnn {bb} {Latin}
2305 {
2306   \@@_set_mathalphabet_Latin:nnn {bb} {up,it}{#1}
2307   \@@_set_mathalphabet_pos:nnnn {bb} {C} {up,it} {#1}
2308   \@@_set_mathalphabet_pos:nnnn {bb} {H} {up,it} {#1}
2309   \@@_set_mathalphabet_pos:nnnn {bb} {N} {up,it} {#1}
2310   \@@_set_mathalphabet_pos:nnnn {bb} {P} {up,it} {#1}
2311   \@@_set_mathalphabet_pos:nnnn {bb} {Q} {up,it} {#1}
2312   \@@_set_mathalphabet_pos:nnnn {bb} {R} {up,it} {#1}
2313   \@@_set_mathalphabet_pos:nnnn {bb} {Z} {up,it} {#1}
2314 }

```

```

2315
2316 \@@_new_alphabet_config:nnn {bb} {num}
2317 {
2318   \@@_set_mathalphabet_numbers:nnn {bb} {up}{#1}
2319 }
2320
2321 \@@_new_alphabet_config:nnn {bb} {misc}
2322 {
2323   \@@_set_mathalphabet_pos:nnnn {bb}      {Pi} {up,it} {#1}
2324   \@@_set_mathalphabet_pos:nnnn {bb}      {pi} {up,it} {#1}
2325   \@@_set_mathalphabet_pos:nnnn {bb}      {Gamma} {up,it} {#1}
2326   \@@_set_mathalphabet_pos:nnnn {bb}      {gamma} {up,it} {#1}
2327   \@@_set_mathalphabet_pos:nnnn {bb} {summation} {up} {#1}
2328 }
2329
2330 \@@_new_alphabet_config:nnn {bbit} {misc}
2331 {
2332   \@@_set_mathalphabet_pos:nnnn {bbit} {D} {up,it} {#1}
2333   \@@_set_mathalphabet_pos:nnnn {bbit} {d} {up,it} {#1}
2334   \@@_set_mathalphabet_pos:nnnn {bbit} {e} {up,it} {#1}
2335   \@@_set_mathalphabet_pos:nnnn {bbit} {i} {up,it} {#1}
2336   \@@_set_mathalphabet_pos:nnnn {bbit} {j} {up,it} {#1}
2337 }

```

#### L.3.4 *Script and caligraphic: scr and cal*

```

2338 \@@_new_alphabet_config:nnn {scr} {Latin}
2339 {
2340   \@@_set_mathalphabet_Latin:nnn {scr}      {up,it}{#1}
2341   \@@_set_mathalphabet_pos:nnnn {scr} {B}{up,it}{#1}
2342   \@@_set_mathalphabet_pos:nnnn {scr} {E}{up,it}{#1}
2343   \@@_set_mathalphabet_pos:nnnn {scr} {F}{up,it}{#1}
2344   \@@_set_mathalphabet_pos:nnnn {scr} {H}{up,it}{#1}
2345   \@@_set_mathalphabet_pos:nnnn {scr} {I}{up,it}{#1}
2346   \@@_set_mathalphabet_pos:nnnn {scr} {L}{up,it}{#1}
2347   \@@_set_mathalphabet_pos:nnnn {scr} {M}{up,it}{#1}
2348   \@@_set_mathalphabet_pos:nnnn {scr} {R}{up,it}{#1}
2349 }
2350
2351 \@@_new_alphabet_config:nnn {scr} {latin}
2352 {
2353   \@@_set_mathalphabet_latin:nnn {scr}      {up,it}{#1}
2354   \@@_set_mathalphabet_pos:nnnn {scr} {e}{up,it}{#1}
2355   \@@_set_mathalphabet_pos:nnnn {scr} {g}{up,it}{#1}
2356   \@@_set_mathalphabet_pos:nnnn {scr} {o}{up,it}{#1}
2357 }

```

These are by default synonyms for the above, but with the STIX fonts we want to use the alternate alphabet.

```

2358 \@@_new_alphabet_config:nnn {cal} {Latin}
2359 {

```



```

2360 \@@_set_mathalphabet_Latin:nnn {cal} {up,it}{#1}
2361 \@@_set_mathalphabet_pos:nnnn {cal} {B}{up,it}{#1}
2362 \@@_set_mathalphabet_pos:nnnn {cal} {E}{up,it}{#1}
2363 \@@_set_mathalphabet_pos:nnnn {cal} {F}{up,it}{#1}
2364 \@@_set_mathalphabet_pos:nnnn {cal} {H}{up,it}{#1}
2365 \@@_set_mathalphabet_pos:nnnn {cal} {I}{up,it}{#1}
2366 \@@_set_mathalphabet_pos:nnnn {cal} {L}{up,it}{#1}
2367 \@@_set_mathalphabet_pos:nnnn {cal} {M}{up,it}{#1}
2368 \@@_set_mathalphabet_pos:nnnn {cal} {R}{up,it}{#1}
2369 }

```

### L.3.5 *Fraktur or fraktur or blackletter: frak*

```

2370 \@@_new_alphabet_config:nnn {frak} {Latin}
2371 {
2372 \@@_set_mathalphabet_Latin:nnn {frak} {up,it}{#1}
2373 \@@_set_mathalphabet_pos:nnnn {frak} {C}{up,it}{#1}
2374 \@@_set_mathalphabet_pos:nnnn {frak} {H}{up,it}{#1}
2375 \@@_set_mathalphabet_pos:nnnn {frak} {I}{up,it}{#1}
2376 \@@_set_mathalphabet_pos:nnnn {frak} {R}{up,it}{#1}
2377 \@@_set_mathalphabet_pos:nnnn {frak} {Z}{up,it}{#1}
2378 }
2379 \@@_new_alphabet_config:nnn {frak} {latin}
2380 {
2381 \@@_set_mathalphabet_latin:nnn {frak} {up,it}{#1}
2382 }

```

### L.3.6 *Sans serif upright: sfup*

```

2383 \@@_new_alphabet_config:nnn {sfup} {num}
2384 {
2385 \@@_set_mathalphabet_numbers:nnn {sf} {up}{#1}
2386 \@@_set_mathalphabet_numbers:nnn {sfup} {up}{#1}
2387 }
2388 \@@_new_alphabet_config:nnn {sfup} {Latin}
2389 {
2390 \bool_if:NTF \g_@@_sfliteral_bool
2391 {
2392 \@@_set_normal_Latin:nn {sfup} {#1}
2393 \@@_set_mathalphabet_Latin:nnn {sf} {up}{#1}
2394 }
2395 {
2396 \bool_if:NT \g_@@_upsans_bool
2397 {
2398 \@@_set_normal_Latin:nn {sfup,sfit} {#1}
2399 \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
2400 }
2401 }
2402 \@@_set_mathalphabet_Latin:nnn {sfup} {up,it}{#1}
2403 }
2404 \@@_new_alphabet_config:nnn {sfup} {latin}

```

```

2405 {
2406   \bool_if:NTF \g_@@_sfliteral_bool
2407   {
2408     \@@_set_normal_latin:nn {sfup} {#1}
2409     \@@_set_mathalphabet_latin:nnn {sf} {up}{#1}
2410   }
2411   {
2412     \bool_if:NT \g_@@_upsans_bool
2413     {
2414       \@@_set_normal_latin:nn {sfup,sfit} {#1}
2415       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
2416     }
2417   }
2418   \@@_set_mathalphabet_latin:nnn {sfup} {up,it}{#1}
2419 }

```

### L.3.7 *Sans serif italic: sfit*

```

2420 \@@_new_alphabet_config:nnn {sfit} {Latin}
2421 {
2422   \bool_if:NTF \g_@@_sfliteral_bool
2423   {
2424     \@@_set_normal_Latin:nn {sfit} {#1}
2425     \@@_set_mathalphabet_Latin:nnn {sf} {it}{#1}
2426   }
2427   {
2428     \bool_if:NF \g_@@_upsans_bool
2429     {
2430       \@@_set_normal_Latin:nn {sfup,sfit} {#1}
2431       \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
2432     }
2433   }
2434   \@@_set_mathalphabet_Latin:nnn {sfit} {up,it}{#1}
2435 }
2436 \@@_new_alphabet_config:nnn {sfit} {latin}
2437 {
2438   \bool_if:NTF \g_@@_sfliteral_bool
2439   {
2440     \@@_set_normal_latin:nn {sfit} {#1}
2441     \@@_set_mathalphabet_latin:nnn {sf} {it}{#1}
2442   }
2443   {
2444     \bool_if:NF \g_@@_upsans_bool
2445     {
2446       \@@_set_normal_latin:nn {sfup,sfit} {#1}
2447       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
2448     }
2449   }
2450   \@@_set_mathalphabet_latin:nnn {sfit} {up,it}{#1}
2451 }

```

### L.3.8 *Typewriter or monospaced: tt*

```
2452 \@@_new_alphabet_config:nnn {tt} {num}
2453 {
2454   \@@_set_mathalphabet_numbers:nnn {tt} {up}{#1}
2455 }
2456 \@@_new_alphabet_config:nnn {tt} {Latin}
2457 {
2458   \@@_set_mathalphabet_Latin:nnn {tt} {up,it}{#1}
2459 }
2460 \@@_new_alphabet_config:nnn {tt} {latin}
2461 {
2462   \@@_set_mathalphabet_latin:nnn {tt} {up,it}{#1}
2463 }
```

### L.3.9 *Bold Italic: bfit*

```
2464 \@@_new_alphabet_config:nnn {bfit} {Latin}
2465 {
2466   \bool_if:NF \g_@@_bfupLatin_bool
2467   {
2468     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
2469   }
2470   \@@_set_mathalphabet_Latin:nnn {bfit} {up,it}{#1}
2471   \bool_if:NTF \g_@@_bfliteral_bool
2472   {
2473     \@@_set_normal_Latin:nn {bfit} {#1}
2474     \@@_set_mathalphabet_Latin:nnn {bf} {it}{#1}
2475   }
2476   {
2477     \bool_if:NF \g_@@_bfupLatin_bool
2478     {
2479       \@@_set_normal_Latin:nn {bfup,bfit} {#1}
2480       \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
2481     }
2482   }
2483 }
2484
2485 \@@_new_alphabet_config:nnn {bfit} {latin}
2486 {
2487   \bool_if:NF \g_@@_bfuplatin_bool
2488   {
2489     \@@_set_normal_latin:nn {bfup,bfit} {#1}
2490   }
2491   \@@_set_mathalphabet_latin:nnn {bfit} {up,it}{#1}
2492   \bool_if:NTF \g_@@_bfliteral_bool
2493   {
2494     \@@_set_normal_latin:nn {bfit} {#1}
2495     \@@_set_mathalphabet_latin:nnn {bf} {it}{#1}
2496   }
2497   {
```

```

2498 \bool_if:NF \g_@@_bfuplatin_bool
2499 {
2500 \@@_set_normal_latin:nn {bfup,bfit} {#1}
2501 \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
2502 }
2503 }
2504 }
2505
2506 \@@_new_alphabet_config:nnn {bfit} {Greek}
2507 {
2508 \@@_set_mathalphabet_Greek:nnn {bfit} {up,it}{#1}
2509 \bool_if:NTF \g_@@_bfliteral_bool
2510 {
2511 \@@_set_normal_Greek:nn {bfit}{#1}
2512 \@@_set_mathalphabet_Greek:nnn {bf} {it}{#1}
2513 }
2514 {
2515 \bool_if:NF \g_@@_bfupGreek_bool
2516 {
2517 \@@_set_normal_Greek:nn {bfup,bfit}{#1}
2518 \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
2519 }
2520 }
2521 }
2522
2523 \@@_new_alphabet_config:nnn {bfit} {greek}
2524 {
2525 \@@_set_mathalphabet_greek:nnn {bfit} {up,it} {#1}
2526 \bool_if:NTF \g_@@_bfliteral_bool
2527 {
2528 \@@_set_normal_greek:nn {bfit} {#1}
2529 \@@_set_mathalphabet_greek:nnn {bf} {it} {#1}
2530 }
2531 {
2532 \bool_if:NF \g_@@_bfupgreek_bool
2533 {
2534 \@@_set_normal_greek:nn {bfit,bfup} {#1}
2535 \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
2536 }
2537 }
2538 }
2539
2540 \@@_new_alphabet_config:nnn {bfit} {misc}
2541 {
2542 \bool_if:NTF \g_@@_literal_Nabla_bool
2543 { \@@_set_normal_char:nnn {Nabla}{bfit}{#1} }
2544 {
2545 \bool_if:NF \g_@@_upNabla_bool
2546 { \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1} }

```

```

2547 }
2548 \bool_if:NTF \g_@@_literal_partial_bool
2549 { \@@_set_normal_char:nnn {partial}{bfit}{#1} }
2550 {
2551   \bool_if:NF \g_@@_uppartial_bool
2552   { \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1} }
2553 }
2554 \@@_set_mathalphabet_pos:nnnn {bfit} {partial} {up,it}{#1}
2555 \@@_set_mathalphabet_pos:nnnn {bfit} {Nabla} {up,it}{#1}
2556 \bool_if:NTF \g_@@_literal_partial_bool
2557 {
2558   \@@_set_mathalphabet_pos:nnnn {bf} {partial} {it}{#1}
2559 }
2560 {
2561   \bool_if:NF \g_@@_uppartial_bool
2562   {
2563     \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
2564   }
2565 }
2566 \bool_if:NTF \g_@@_literal_Nabla_bool
2567 {
2568   \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {it}{#1}
2569 }
2570 {
2571   \bool_if:NF \g_@@_upNabla_bool
2572   {
2573     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
2574   }
2575 }
2576 }

```

### L.3.10 *Bold Upright: bfup*

```

2577 \@@_new_alphabet_config:nnn {bfup} {num}
2578 {
2579   \@@_set_mathalphabet_numbers:nnn {bf} {up}{#1}
2580   \@@_set_mathalphabet_numbers:nnn {bfup} {up}{#1}
2581 }
2582
2583 \@@_new_alphabet_config:nnn {bfup} {Latin}
2584 {
2585   \bool_if:NT \g_@@_bfupLatin_bool
2586   {
2587     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
2588   }
2589   \@@_set_mathalphabet_Latin:nnn {bfup} {up,it}{#1}
2590   \bool_if:NTF \g_@@_bfliteral_bool
2591   {
2592     \@@_set_normal_Latin:nn {bfup} {#1}
2593     \@@_set_mathalphabet_Latin:nnn {bf} {up}{#1}

```

```

2594 }
2595 {
2596   \bool_if:NT \g_@@_bfupLatin_bool
2597   {
2598     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
2599     \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
2600   }
2601 }
2602 }
2603
2604 \@@_new_alphabet_config:nnn {bfup} {latin}
2605 {
2606   \bool_if:NT \g_@@_bfuplatin_bool
2607   {
2608     \@@_set_normal_latin:nn {bfup,bfit} {#1}
2609   }
2610   \@@_set_mathalphabet_latin:nnn {bfup} {up,it}{#1}
2611   \bool_if:NTF \g_@@_bfliteral_bool
2612   {
2613     \@@_set_normal_latin:nn {bfup} {#1}
2614     \@@_set_mathalphabet_latin:nnn {bf} {up}{#1}
2615   }
2616   {
2617     \bool_if:NT \g_@@_bfuplatin_bool
2618     {
2619       \@@_set_normal_latin:nn {bfup,bfit} {#1}
2620       \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
2621     }
2622   }
2623 }
2624 \@@_new_alphabet_config:nnn {bfup} {Greek}
2625 {
2626   \@@_set_mathalphabet_Greek:nnn {bfup} {up,it}{#1}
2627   \bool_if:NTF \g_@@_bfliteral_bool
2628   {
2629     \@@_set_normal_Greek:nn {bfup}{#1}
2630     \@@_set_mathalphabet_Greek:nnn {bf} {up}{#1}
2631   }
2632   {
2633     \bool_if:NT \g_@@_bfupGreek_bool
2634     {
2635       \@@_set_normal_Greek:nn {bfup,bfit}{#1}
2636       \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
2637     }
2638   }
2639 }
2640
2641 \@@_new_alphabet_config:nnn {bfup} {greek}
2642 {

```

```

2643 \@@_set_mathalphabet_greek:nnn {bfup} {up,it} {#1}
2644 \bool_if:NTF \g_@@_bfliteral_bool
2645 {
2646   \@@_set_normal_greek:nn {bfup} {#1}
2647   \@@_set_mathalphabet_greek:nnn {bf} {up} {#1}
2648 }
2649 {
2650   \bool_if:NT \g_@@_bfupgreek_bool
2651   {
2652     \@@_set_normal_greek:nn {bfup,bfit} {#1}
2653     \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
2654   }
2655 }
2656 }
2657
2658 \@@_new_alphabet_config:nnn {bfup} {misc}
2659 {
2660   \bool_if:NTF \g_@@_literal_Nabla_bool
2661   {
2662     \@@_set_normal_char:nnn {Nabla}{bfup}{#1}
2663   }
2664   {
2665     \bool_if:NT \g_@@_upNabla_bool
2666     {
2667       \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1}
2668     }
2669   }
2670   \bool_if:NTF \g_@@_literal_partial_bool
2671   {
2672     \@@_set_normal_char:nnn {partial}{bfup}{#1}
2673   }
2674   {
2675     \bool_if:NT \g_@@_uppartial_bool
2676     {
2677       \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1}
2678     }
2679   }
2680   \@@_set_mathalphabet_pos:nnnn {bfup} {partial} {up,it}{#1}
2681   \@@_set_mathalphabet_pos:nnnn {bfup} {Nabla} {up,it}{#1}
2682   \@@_set_mathalphabet_pos:nnnn {bfup} {digamma} {up}{#1}
2683   \@@_set_mathalphabet_pos:nnnn {bfup} {Digamma} {up}{#1}
2684   \@@_set_mathalphabet_pos:nnnn {bf} {digamma} {up}{#1}
2685   \@@_set_mathalphabet_pos:nnnn {bf} {Digamma} {up}{#1}
2686   \bool_if:NTF \g_@@_literal_partial_bool
2687   {
2688     \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up}{#1}
2689   }
2690   {
2691     \bool_if:NT \g_@@_uppartial_bool

```

```

2692     {
2693       \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
2694     }
2695   }
2696   \bool_if:NTF \g_@@_literal_Nabla_bool
2697   {
2698     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up}{#1}
2699   }
2700   {
2701     \bool_if:NT \g_@@_upNabla_bool
2702     {
2703       \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
2704     }
2705   }
2706 }

```

### L.3.11 *Bold fractur or fraktur or blackletter: bffrak*

```

2707 \@@_new_alphabet_config:nnn {bffrak} {Latin}
2708 {
2709   \@@_set_mathalphabet_Latin:nnn {bffrak} {up,it}{#1}
2710 }
2711
2712 \@@_new_alphabet_config:nnn {bffrak} {latin}
2713 {
2714   \@@_set_mathalphabet_latin:nnn {bffrak} {up,it}{#1}
2715 }

```

### L.3.12 *Bold script or calligraphic: bfscr*

```

2716 \@@_new_alphabet_config:nnn {bfscr} {Latin}
2717 {
2718   \@@_set_mathalphabet_Latin:nnn {bfscr} {up,it}{#1}
2719 }
2720 \@@_new_alphabet_config:nnn {bfscr} {latin}
2721 {
2722   \@@_set_mathalphabet_latin:nnn {bfscr} {up,it}{#1}
2723 }
2724 \@@_new_alphabet_config:nnn {bfcal} {Latin}
2725 {
2726   \@@_set_mathalphabet_Latin:nnn {bfcal} {up,it}{#1}
2727 }

```

### L.3.13 *Bold upright sans serif: bfsfup*

```

2728 \@@_new_alphabet_config:nnn {bfsfup} {num}
2729 {
2730   \@@_set_mathalphabet_numbers:nnn {bfsf} {up}{#1}
2731   \@@_set_mathalphabet_numbers:nnn {bfsfup} {up}{#1}
2732 }
2733 \@@_new_alphabet_config:nnn {bfsfup} {Latin}
2734 {

```



```

2735 \bool_if:NTF \g_@@_sfliteral_bool
2736 {
2737   \@@_set_normal_Latin:nn {bfsfup} {#1}
2738   \@@_set_mathalphabet_Latin:nnn {bfsf} {up}{#1}
2739 }
2740 {
2741   \bool_if:NT \g_@@_upsans_bool
2742   {
2743     \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
2744     \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
2745   }
2746 }
2747 \@@_set_mathalphabet_Latin:nnn {bfsfup} {up,it}{#1}
2748 }
2749
2750 \@@_new_alphabet_config:nnn {bfsfup} {latin}
2751 {
2752   \bool_if:NTF \g_@@_sfliteral_bool
2753   {
2754     \@@_set_normal_latin:nn {bfsfup} {#1}
2755     \@@_set_mathalphabet_latin:nnn {bfsf} {up}{#1}
2756   }
2757   {
2758     \bool_if:NT \g_@@_upsans_bool
2759     {
2760       \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
2761       \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
2762     }
2763   }
2764   \@@_set_mathalphabet_latin:nnn {bfsfup} {up,it}{#1}
2765 }
2766
2767 \@@_new_alphabet_config:nnn {bfsfup} {Greek}
2768 {
2769   \bool_if:NTF \g_@@_sfliteral_bool
2770   {
2771     \@@_set_normal_Greek:nn {bfsfup}{#1}
2772     \@@_set_mathalphabet_Greek:nnn {bfsf} {up}{#1}
2773   }
2774   {
2775     \bool_if:NT \g_@@_upsans_bool
2776     {
2777       \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
2778       \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
2779     }
2780   }
2781   \@@_set_mathalphabet_Greek:nnn {bfsfup} {up,it}{#1}
2782 }
2783

```

```

2784 \@@_new_alphabet_config:nnn {bfsfup} {greek}
2785 {
2786   \bool_if:NTF \g_@@_sfliteral_bool
2787   {
2788     \@@_set_normal_greek:nn {bfsfup} {#1}
2789     \@@_set_mathalphabet_greek:nnn {bfsf} {up} {#1}
2790   }
2791   {
2792     \bool_if:NT \g_@@_upsans_bool
2793     {
2794       \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
2795       \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
2796     }
2797   }
2798   \@@_set_mathalphabet_greek:nnn {bfsfup} {up,it} {#1}
2799 }
2800 \@@_new_alphabet_config:nnn {bfsfup} {misc}
2801 {
2802   \bool_if:NTF \g_@@_literal_Nabla_bool
2803   {
2804     \@@_set_normal_char:nnn {Nabla}{bfsfup}{#1}
2805   }
2806   {
2807     \bool_if:NT \g_@@_upNabla_bool
2808     {
2809       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
2810     }
2811   }
2812   \bool_if:NTF \g_@@_literal_partial_bool
2813   {
2814     \@@_set_normal_char:nnn {partial}{bfsfup}{#1}
2815   }
2816   {
2817     \bool_if:NT \g_@@_uppartial_bool
2818     {
2819       \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
2820     }
2821   }
2822   \@@_set_mathalphabet_pos:nnnn {bfsfup} {partial} {up,it}{#1}
2823   \@@_set_mathalphabet_pos:nnnn {bfsfup} {Nabla} {up,it}{#1}
2824   \bool_if:NTF \g_@@_literal_partial_bool
2825   {
2826     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up}{#1}
2827   }
2828   {
2829     \bool_if:NT \g_@@_uppartial_bool
2830     {
2831       \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
2832     }
2833   }

```

```

2833 }
2834 \bool_if:NTF \g_@@_literal_Nabla_bool
2835 {
2836   \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up}{#1}
2837 }
2838 {
2839   \bool_if:NT \g_@@_upNabla_bool
2840   {
2841     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
2842   }
2843 }
2844 }

```

### L.3.14 *Bold italic sans serif: bfsfit*

```

2845 \@@_new_alphabet_config:nnn {bfsfit} {Latin}
2846 {
2847   \bool_if:NTF \g_@@_sfliteral_bool
2848   {
2849     \@@_set_normal_Latin:nn {bfsfit} {#1}
2850     \@@_set_mathalphabet_Latin:nnn {bfsf} {it}{#1}
2851   }
2852   {
2853     \bool_if:NF \g_@@_upsans_bool
2854     {
2855       \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
2856       \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
2857     }
2858   }
2859   \@@_set_mathalphabet_Latin:nnn {bfsfit} {up,it}{#1}
2860 }
2861
2862 \@@_new_alphabet_config:nnn {bfsfit} {latin}
2863 {
2864   \bool_if:NTF \g_@@_sfliteral_bool
2865   {
2866     \@@_set_normal_latin:nn {bfsfit} {#1}
2867     \@@_set_mathalphabet_latin:nnn {bfsf} {it}{#1}
2868   }
2869   {
2870     \bool_if:NF \g_@@_upsans_bool
2871     {
2872       \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
2873       \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
2874     }
2875   }
2876   \@@_set_mathalphabet_latin:nnn {bfsfit} {up,it}{#1}
2877 }
2878
2879 \@@_new_alphabet_config:nnn {bfsfit} {Greek}

```

```

2880 {
2881   \bool_if:NTF \g_@@_sfliteral_bool
2882   {
2883     \@@_set_normal_Greek:nn {bfsfit}{#1}
2884     \@@_set_mathalphabet_Greek:nnn {bfsf} {it}{#1}
2885   }
2886   {
2887     \bool_if:NF \g_@@_upsans_bool
2888     {
2889       \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
2890       \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
2891     }
2892   }
2893   \@@_set_mathalphabet_Greek:nnn {bfsfit} {up,it}{#1}
2894 }
2895
2896 \@@_new_alphabet_config:nnn {bfsfit} {greek}
2897 {
2898   \bool_if:NTF \g_@@_sfliteral_bool
2899   {
2900     \@@_set_normal_greek:nn {bfsfit} {#1}
2901     \@@_set_mathalphabet_greek:nnn {bfsf} {it} {#1}
2902   }
2903   {
2904     \bool_if:NF \g_@@_upsans_bool
2905     {
2906       \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
2907       \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
2908     }
2909   }
2910   \@@_set_mathalphabet_greek:nnn {bfsfit} {up,it} {#1}
2911 }
2912
2913 \@@_new_alphabet_config:nnn {bfsfit} {misc}
2914 {
2915   \bool_if:NTF \g_@@_literal_Nabla_bool
2916   {
2917     \@@_set_normal_char:nnn {Nabla}{bfsfit}{#1}
2918   }
2919   {
2920     \bool_if:NF \g_@@_upNabla_bool
2921     {
2922       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
2923     }
2924   }
2925   \bool_if:NTF \g_@@_literal_partial_bool
2926   {
2927     \@@_set_normal_char:nnn {partial}{bfsfit}{#1}
2928   }

```

```

2929 {
2930   \bool_if:NF \g_@@_uppartial_bool
2931   {
2932     \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
2933   }
2934 }
2935 \@@_set_mathalphabet_pos:nnnn {bfsfit} {partial} {up,it}{#1}
2936 \@@_set_mathalphabet_pos:nnnn {bfsfit} {Nabla} {up,it}{#1}
2937 \bool_if:NTF \g_@@_literal_partial_bool
2938 {
2939   \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {it}{#1}
2940 }
2941 {
2942   \bool_if:NF \g_@@_uppartial_bool
2943   {
2944     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
2945   }
2946 }
2947 \bool_if:NTF \g_@@_literal_Nabla_bool
2948 {
2949   \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {it}{#1}
2950 }
2951 {
2952   \bool_if:NF \g_@@_upNabla_bool
2953   {
2954     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
2955   }
2956 }
2957 }
2958 </alphabets>

```

## L.4 Compatibility

```

2959 <*compat>

```

```

\@@_check_and_fix:NNnnnn #1 : command
                        #2 : factory command
                        #3 : parameter text
                        #4 : expected replacement text
                        #5 : new replacement text for LuaTEX
                        #6 : new replacement text for XYTEX

```

Tries to patch  $\langle command \rangle$ . If  $\langle command \rangle$  is undefined, do nothing. Otherwise it must be a macro with the given  $\langle parameter text \rangle$  and  $\langle expected replacement text \rangle$ , created by the given  $\langle factory command \rangle$  or equivalent. In this case it will be overwritten using the  $\langle parameter text \rangle$  and the  $\langle new replacement text for LuaT_{E}X \rangle$  or the  $\langle new replacement text for X_{Y}T_{E}X \rangle$ , depending on the engine. Otherwise issue a warning and don't overwrite.

```

2960 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnnn
2961 {

```

```

2962 \cs_if_exist:NT #1
2963 {
2964   \token_if_macro:NTF #1
2965   {
2966     \group_begin:
2967     #2 \@@_tmpa:w #3 { #4 }
2968     \cs_if_eq:NNTF #1 \@@_tmpa:w
2969     {
2970       \msg_info:nxx { unicode-math } { patch-macro }
2971       { \token_to_str:N #1 }
2972       \group_end:
2973       #2 #1 #3
2974       { #6 }
2975       { #5 }
2976     }
2977     {
2978       \msg_warning:nxxx { unicode-math } { wrong-meaning }
2979       { \token_to_str:N #1 } { \token_to_meaning:N #1 }
2980       { \token_to_meaning:N \@@_tmpa:w }
2981       \group_end:
2982     }
2983   }
2984   {
2985     \msg_warning:nxx { unicode-math } { macro-expected }
2986     { \token_to_str:N #1 }
2987   }
2988 }
2989 }

```

`\@@_check_and_fix:NNnnn` #1 : command  
 #2 : factory command  
 #3 : parameter text  
 #4 : expected replacement text  
 #5 : new replacement text

Tries to patch `<command>`. If `<command>` is undefined, do nothing. Otherwise it must be a macro with the given `<parameter text>` and `<expected replacement text>`, created by the given `<factory command>` or equivalent. In this case it will be overwritten using the `<parameter text>` and the `<new replacement text>`. Otherwise issue a warning and don't overwrite.

```

2990 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnn
2991 {
2992   \@@_check_and_fix:NNnnnn #1 #2 { #3 } { #4 } { #5 } { #5 }
2993 }

```

`\@@_check_and_fix luatex:NNnnn` #1 : command  
`\@@_check_and_fix luatex:cNnnn` #2 : factory command  
 #3 : parameter text  
 #4 : expected replacement text

#5 : new replacement text

Tries to patch  $\langle command \rangle$ . If  $\text{\texttt{X}\TeX}$  is the current engine or  $\langle command \rangle$  is undefined, do nothing. Otherwise it must be a macro with the given  $\langle parameter\ text \rangle$  and  $\langle expected\ replacement\ text \rangle$ , created by the given  $\langle factory\ command \rangle$  or equivalent. In this case it will be overwritten using the  $\langle parameter\ text \rangle$  and the  $\langle new\ replacement\ text \rangle$ . Otherwise issue a warning and don't overwrite.

```

2994 \cs_new_protected_nopar:Nn \@@_check_and_fix luatex:NNnnn
2995 {
2996 (LU) \@@_check_and_fix:NNnnn #1 #2 { #3 } { #4 } { #5 }
2997 }
2998 \cs_generate_variant:Nn \@@_check_and_fix luatex:NNnnn { c }

```

*url* Simply need to get `url` in a state such that when it switches to math mode and enters ASCII characters, the maths setup (i.e., unicode-math) doesn't remap the symbols into Plane 1. Which is, of course, what `\mathup` is doing.

This is the same as writing, e.g., `\def\UrlFont{\ttfamily\@@_switchto_up:}` but activates automatically so old documents that might change the `\url` font still work correctly.

```

2999 \AtEndOfPackageFile * {url}
3000 {
3001 \tl_put_left:Nn \Url@FormatString { \@@_switchto_up: }
3002 \tl_put_right:Nn \Url@Specials
3003 {
3004 \do\`\mathchar`\`
3005 \do\'\mathchar`\`
3006 \do\$\mathchar`\$
3007 \do\&\mathchar`\&
3008 }
3009 }

```

*amsmath* Since the mathcode of `\-` is greater than eight bits, this piece of `\AtBeginDocument` code from `amsmath` dies if we try and set the maths font in the preamble:

```

3010 \AtEndOfPackageFile * {amsmath}
3011 {
3012 (*XE)
3013 \tl_remove_once:Nn \@begindocumenthook
3014 {
3015 \mathchardef\std@minus\mathcode`\-\relax
3016 \mathchardef\std@equal\mathcode`\=\relax
3017 }
3018 \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
3019 \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
3020 (/XE)
3021 \cs_set:Npn \@cdots {\mathinner{\cdots}}
3022 \cs_set_eq:NN \dotsb@ \cdots

```

This isn't as clever as the amsmath definition but I think it works:

```

3023 (*XE)
3024 \def \resetMathstrut@
3025 {%
3026 \setbox\z@\hbox{${$}%}
3027 \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
3028 }

```

The subarray environment uses inappropriate font dimensions.

```

3029 \@@_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 }
3030 {
3031 \vcenter
3032 \bgroup
3033 \Let@
3034 \restore@math@cr
3035 \default@tag
3036 \baselineskip \fontdimen 10~ \scriptfont \tw@
3037 \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
3038 \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
3039 \lineskiplimit \lineskip
3040 \ialign
3041 \bgroup
3042 \ifx c #1 \hfil \fi
3043 $ \math \scriptstyle ## $
3044 \hfil
3045 \crr
3046 }
3047 {
3048 \vcenter
3049 \c_group_begin_token
3050 \Let@
3051 \restore@math@cr
3052 \default@tag
3053 \skip_set:Nn \baselineskip
3054 {

```

Here we use stack top shift + stack bottom shift, which sounds reasonable.

```

3055 \@@_stack_num_up:N \scriptstyle
3056 + \@@_stack_denom_down:N \scriptstyle
3057 }

```

Here we use the minimum stack gap.

```

3058 \lineskip \@@_stack_vgap:N \scriptstyle
3059 \lineskiplimit \lineskip
3060 \ialign
3061 \c_group_begin_token
3062 \token_if_eq_meaning:NNT c #1 { \hfil }
3063 \c_math_toggle_token
3064 \math
3065 \scriptstyle

```



```

3066     \c_parameter_token \c_parameter_token
3067     \c_math_toggle_token
3068     \hfil
3069     \c_rcr
3070 }
3071 </XE>

```

The roots need a complete rework.

```

3072 \@@_check_and_fix luatex:NNnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 }
3073 {
3074   \setbox \rootbox \hbox
3075   {
3076     $ \m@th \scriptscriptstyle { #1 } $
3077   }
3078   \mathchoice
3079     { \r@@@et \displaystyle { #2 } }
3080     { \r@@@et \textstyle { #2 } }~
3081     { \r@@@et \scriptstyle { #2 } }
3082     { \r@@@et \scriptscriptstyle { #2 } }
3083   \egroup
3084 }
3085 {
3086   \bool_if:nTF
3087   {
3088     \int_compare_p:nNn { \uproot@ } = { \c_zero }
3089     && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
3090   }
3091   {
3092     \Uroot \l_@@_radical_sqrt_tl { #1 } { #2 }
3093   }
3094   {
3095     \hbox_set:Nn \rootbox
3096     {
3097       \c_math_toggle_token
3098       \m@th
3099       \scriptscriptstyle { #1 }
3100       \c_math_toggle_token
3101     }
3102     \mathchoice
3103       { \r@@@et \displaystyle { #2 } }
3104       { \r@@@et \textstyle { #2 } }
3105       { \r@@@et \scriptstyle { #2 } }
3106       { \r@@@et \scriptscriptstyle { #2 } }
3107   }
3108   \c_group_end_token
3109 }
3110 \@@_check_and_fix:NNnnnn \r@@@et \cs_set_nopar:Npn { #1 #2 }
3111 {
3112   \setboxz@h { $ \m@th #1 \sqrt{sign { #2 } } $ }
3113   \dimen@ \ht\z@

```

```

3114 \advance \dimen@ -\dp\z@
3115 \setbox\@ne \hbox { $ \m@th #1 \mskip \uproot@ mu $ }
3116 \advance \dimen@ by 1.667 \wd\@ne
3117 \mkern -\leftroot@ mu
3118 \mkern 5mu
3119 \raise .6\dimen@ \copy\rootbox
3120 \mkern -10mu
3121 \mkern \leftroot@ mu
3122 \boxz@
3123 }
3124 {
3125 \hbox_set:Nn \l_tmpa_box
3126 {
3127 \c_math_toggle_token
3128 \m@th
3129 #1
3130 \mskip \uproot@ mu
3131 \c_math_toggle_token
3132 }
3133 \Uroot \l_@@_radical_sqrt_tl
3134 {
3135 \box_move_up:nn { \box_wd:N \l_tmpa_box }
3136 {
3137 \hbox:n
3138 {
3139 \c_math_toggle_token
3140 \m@th
3141 \mkern -\leftroot@ mu
3142 \box_use:N \rootbox
3143 \mkern \leftroot@ mu
3144 \c_math_toggle_token
3145 }
3146 }
3147 }
3148 { #2 }
3149 }
3150 {
3151 \hbox_set:Nn \l_tmpa_box
3152 {
3153 \c_math_toggle_token
3154 \m@th
3155 #1
3156 \sqrtsign { #2 }
3157 \c_math_toggle_token
3158 }
3159 \hbox_set:Nn \l_tmpb_box
3160 {
3161 \c_math_toggle_token
3162 \m@th

```

```

3163      #1
3164      \mskip \uproot@ mu
3165      \c_math_toggle_token
3166    }
3167    \mkern -\leftroot@ mu
3168    \@@_mathstyle_scale:Nnn #1 { \kern }
3169    {
3170      \fontdimen 63 \l_@@_font
3171    }
3172    \box_move_up:nn
3173    {
3174      \box_wd:N \l_tmpb_box
3175      + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
3176      * \number \fontdimen 65 \l_@@_font / 100
3177    }
3178    {
3179      \box_use:N \rootbox
3180    }
3181    \@@_mathstyle_scale:Nnn #1 { \kern }
3182    {
3183      \fontdimen 64 \l_@@_font
3184    }
3185    \mkern \leftroot@ mu
3186    \box_use_clear:N \l_tmpa_box
3187  }
3188 }

```

*amsopn* This code is to improve the output of alphabetic symbols in text of operator names ( $\sin$ ,  $\cos$ , etc.). Just comment out the offending lines for now:

```

3189 (*XE)
3190 \AtEndOfPackageFile * {amsopn}
3191 {
3192   \cs_set:Npn \newmcodes@
3193   {
3194     \mathcode'\ '39\scan_stop:
3195     \mathcode'\ *42\scan_stop:
3196     \mathcode'\ ."613A\scan_stop:
3197     %% \ifnum\mathcode'\ -=45 \else
3198     %%   \mathchardef\std@minus\mathcode'\ -\relax
3199     %% \fi
3200     \mathcode'\ -45\scan_stop:
3201     \mathcode'\ /47\scan_stop:
3202     \mathcode'\ ":"603A\scan_stop:
3203   }
3204 }
3205 (/XE)

```

*mathtools* *mathtools*’s `\cramped` command and others that make use of its internal version use an incorrect font dimension.

```

3206 (*XE)
3207 \AtEndOfPackageFile * { mathtools }
3208 {
3209   \@@_check_and_fix:NNnnn
3210     \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }
3211   {
3212     \sbox \z@
3213     {
3214       $
3215       \m@th
3216       #1
3217       \nulldelimiterspace = \z@
3218       \radical \z@ { #2 }
3219       $
3220     }
3221     \ifx #1 \displaystyle
3222       \dimen@ = \fontdimen 8 \textfont 3
3223       \advance \dimen@ .25 \fontdimen 5 \textfont 2
3224     \else
3225       \dimen@ = 1.25 \fontdimen 8
3226       \ifx #1 \textstyle
3227         \textfont
3228       \else
3229         \ifx #1 \scriptstyle
3230           \scriptfont
3231         \else
3232           \scriptscriptfont
3233         \fi
3234       \fi
3235       3
3236     \fi
3237     \advance \dimen@ -\ht\z@
3238     \ht\z@ = -\dimen@
3239     \box\z@
3240   }

```

The  $\text{\LaTeX}$  version is pretty similar to the legacy version, only using the correct font dimensions. Note we used ‘`\XeTeXradical`’ with the family 255 to be almost sure that the radical rule width is not set. Former use of ‘`\newfam`’ had an upsetting effect on legacy math alphabets.

```

3241 {
3242   \hbox_set:Nn \l_tmpa_box
3243   {
3244     \color@setgroup
3245     \c_math_toggle_token
3246     \m@th
3247     #1

```

```

3248 \dim_zero:N \nulldelimiterspace
3249 \XeTeXradical \c_two_hundred_fifty_five \c_zero { #2 }
3250 \c_math_toggle_token
3251 \color@endgroup
3252 }
3253 \box_set_ht:Nn \l_tmpa_box
3254 {
3255 \box_ht:N \l_tmpa_box

```

Here we use the radical vertical gap.

```

3256 - \@@_radical_vgap:N #1
3257 }
3258 \box_use_clear:N \l_tmpa_box
3259 }
3260 }
3261 /XE

```

`\overbracket` and `\underbracket` are defined in `mathtools`. `\overbracket` and `\underbracket` take optional arguments and are defined in terms of rules, so we keep them, and rename ours to `\Uoverbracket` and `\Underbracket`.

```

3262 \AtEndOfPackageFile * { mathtools }
3263 {
3264 \cs_set_eq:NN \MToverbracket \overbracket
3265 \cs_set_eq:NN \MTunderbracket \underbracket
3266
3267 \AtBeginDocument
3268 {
3269 \msg_warning:nn { unicode-math } { mathtools-overbracket }
3270
3271 \def\downbracketfill#1#2
3272 {%

```

Original definition used the height of `\bracketd` which is not available with Unicode fonts, so we are hard coding the 5/18ex suggested by `mathtools`'s documentation.

```

3273 \edef\l_MT_bracketheight_fdim{.27ex}%
3274 \downbracketend{#1}{#2}
3275 \leaders \vrule \@height #1 \@depth \z@ \hfill
3276 \downbracketend{#1}{#2}%
3277 }
3278 \def\upbracketfill#1#2
3279 {%
3280 \edef\l_MT_bracketheight_fdim{.27ex}%
3281 \upbracketend{#1}{#2}
3282 \leaders \vrule \@height \z@ \@depth #1 \hfill
3283 \upbracketend{#1}{#2}%
3284 }
3285 \let\Uoverbracket =\overbracket
3286 \let\Underbracket=\underbracket
3287 \let\overbracket =\MToverbracket

```

```

3288 \let\underbracket =\MTunderbracket
3289 }% end of AtBeginDocument

```

`\dblcolon` `mathtools` defines several commands as combinations of colons and other characters, but with meanings incompatible to `unicode-math`. Thus we issue a warning. `\coloneqq` `\Coloneqq` Because `mathtools` uses `\providecommand` `\AtBeginDocument`, we can just define the offending commands here. `\eqqcolon`

```

3290 \msg_warning:nn { unicode-math } { mathtools-colon }
3291 \NewDocumentCommand \dblcolon { } { \Colon }
3292 \NewDocumentCommand \coloneqq { } { \coloneq }
3293 \NewDocumentCommand \Coloneqq { } { \Coloneq }
3294 \NewDocumentCommand \eqqcolon { } { \eqcolon }
3295 }

```

### *colonequals*

`\ratio` Similarly to `mathtools`, the `colonequals` defines several colon combinations. Fortunately there are no name clashes, so we can just overwrite their definitions.

```

\coloncolon 3296 \AtEndOfPackageFile * { colonequals }
\minuscolon 3297 {
\colonequals 3298 \msg_warning:nn { unicode-math } { colonequals }
\equalscolon 3299 \RenewDocumentCommand \ratio { } { \mathratio }
\coloncolonequals 3300 \RenewDocumentCommand \coloncolon { } { \Colon }
3301 \RenewDocumentCommand \minuscolon { } { \dashcolon }
3302 \RenewDocumentCommand \colonequals { } { \coloneq }
3303 \RenewDocumentCommand \equalscolon { } { \eqcolon }
3304 \RenewDocumentCommand \coloncolonequals { } { \Coloneq }
3305 }
3306 </compat>

```