

# Experimental Unicode mathematical typesetting: The unicode-math package

*Will Robertson*

*Philipp Stephani, Joseph Wright, Khaled Hosny*

<http://github.com/wspr/unicode-math>

2017/11/18 v0.8i

## *Contents*

<b>I</b>	<b>unicode-math.dtx</b>	<b>5</b>
1	Package declaration	5
<b>II</b>	<b>um-code-base.dtx</b>	<b>7</b>
2	The unicode-math.sty loading file	7
<b>III</b>	<b>um-code-opening.dtx</b>	<b>8</b>
3	Start of the package code	8
3.1	expl3 variants . . . . .	8
3.2	Primitive font commands . . . . .	8
3.2.1	Mathcode and friends . . . . .	8
3.2.2	Font parameters . . . . .	10
3.3	Alphabet Unicode positions (USVs) . . . . .	10
3.4	Overcoming \@onlypreamble . . . . .	10
<b>IV</b>	<b>um-code-variables.dtx</b>	<b>12</b>
4	Variable initialisation	12
<b>V</b>	<b>um-code-api.dtx</b>	<b>15</b>

5	Programmers' interface	15
<b>VI</b>	<b>um-code-ui.dtx</b>	<b>16</b>
6	The user interface commands	16
<b>VII</b>	<b>um-code-pkgopt.dtx</b>	<b>18</b>
7	setup and package options	18
7.1	Defaults . . . . .	23
<b>VIII</b>	<b>um-code-msg.dtx</b>	<b>24</b>
8	Error messages	24
<b>IX</b>	<b>um-code-usv.dtx</b>	<b>27</b>
9	Alphabet Unicode positions	27
9.1	STIX fonts . . . . .	32
<b>X</b>	<b>um-code-setchar.dtx</b>	<b>37</b>
10	Setting up maths chars	37
10.1	A token list to contain the data of the math table . . . . .	37
10.2	Definitions of the active math characters . . . . .	37
10.3	Commands for each symbol/glyph/char . . . . .	38
<b>XI</b>	<b>um-code-mathtext.dtx</b>	<b>42</b>
11	Maths text commands	42
11.1	\setmathfontface . . . . .	42
11.2	Hooks into fontspec . . . . .	42
11.2.1	Text font . . . . .	43
11.2.2	Maths font . . . . .	43
<b>XII</b>	<b>um-code-main.dtx</b>	<b>45</b>
12	The main \setmathfont macro	45
12.1	Functions for setting up symbols with mathcodes . . . . .	50
12.2	Active math characters . . . . .	51
12.3	Delimiter codes . . . . .	52
12.4	(Big) operators . . . . .	53

12.5 Radicals . . . . .	54
<b>XIII um-code-fontopt.dtx</b>	<b>55</b>
<b>13 Font loading options</b>	<b>55</b>
13.1 Math version . . . . .	55
13.2 Script and scriptscript font options . . . . .	55
13.3 Range processing . . . . .	55
<b>XIV um-code-fontparam.dtx</b>	<b>60</b>
<b>14 Common interface for font parameters</b>	<b>60</b>
14.1 Historical commands . . . . .	65
<b>XV um-code-mathmap.dtx</b>	<b>66</b>
<b>15 Mapping in maths alphabets</b>	<b>66</b>
15.1 Hooks into $\LaTeX 2_{\epsilon}$ . . . . .	66
15.2 Setting styles . . . . .	67
15.3 Defining the math style macros . . . . .	68
15.4 Definition of alphabets and styles . . . . .	68
15.4.1 Define symbol style commands . . . . .	70
15.4.2 New names for legacy textmath alphabet selection . . . . .	70
15.4.3 Replacing legacy pure-maths alphabets . . . . .	70
15.4.4 New commands for ambiguous alphabets . . . . .	71
15.5 Defining the math alphabets per style . . . . .	71
15.6 Mapping ‘naked’ math characters . . . . .	73
15.6.1 Functions . . . . .	73
15.6.2 Functions for ‘normal’ alphabet symbols . . . . .	74
15.7 Mapping chars inside a math style . . . . .	76
15.7.1 Functions for setting up the maths alphabets . . . . .	76
15.7.2 Individual mapping functions for different alphabets . . . . .	77
<b>XVI um-code-epilogue.dtx</b>	<b>79</b>
<b>16 Epilogue</b>	<b>79</b>
16.1 Resolving Greek symbol name control sequences . . . . .	79
16.2 Unicode radicals . . . . .	79
16.2.1 Active fractions . . . . .	80
16.3 Synonyms and all the rest . . . . .	82
16.3.1 <code>\not</code> . . . . .	83

<b>XVII</b>	<b>um-code-primes.dtx</b>	<b>85</b>
17	Primes	85
<b>XVIII</b>	<b>um-code-sscript.dtx</b>	<b>92</b>
18	Unicode sub- and super-scripts	92
<b>XIX</b>	<b>um-code-compat.dtx</b>	<b>96</b>
19	Compatibility	96
<b>XX</b>	<b>um-code-alphabets.dtx</b>	<b>106</b>
20	Setting up alphabets	106
20.1	Upright: up . . . . .	106
20.2	Italic: it . . . . .	107
20.3	Blackboard or double-struck: bb and bbit . . . . .	109
20.4	Script and calligraphic: scr and cal . . . . .	110
20.5	Fraktur or fraktur or blackletter: frak . . . . .	111
20.6	Sans serif upright: sfup . . . . .	111
20.7	Sans serif italic: sfit . . . . .	112
20.8	Typewriter or monospaced: tt . . . . .	113
20.9	Bold Italic: bfit . . . . .	113
20.10	Bold Upright: bfup . . . . .	115
20.11	Bold fraktur or fraktur or blackletter: bffrak . . . . .	118
20.12	Bold script or calligraphic: bfscr . . . . .	118
20.13	Bold upright sans serif: bfsfup . . . . .	118
20.14	Bold italic sans serif: bfsfit . . . . .	121

## File I

# unicode-math.dtx

### 1 *Package declaration*

List all dtx files for (a) the ins file and (b) typesetting the code.

```
1 (*dtx)
2 \def\DTXFILES{
3   \DTX{unicode-math.dtx}
4   \DTX{um-code-base.dtx}
5   \DTX{um-code-opening.dtx}
6   \DTX{um-code-variables.dtx}
7   \DTX{um-code-api.dtx}
8   \DTX{um-code-ui.dtx}
9   \DTX{um-code-pkgopt.dtx}
10  \DTX{um-code-msg.dtx}
11  \DTX{um-code-usv.dtx}
12  \DTX{um-code-setchar.dtx}
13  \DTX{um-code-mathtext.dtx}
14  \DTX{um-code-main.dtx}
15  \DTX{um-code-fontopt.dtx}
16  \DTX{um-code-fontparam.dtx}
17  \DTX{um-code-mathmap.dtx}
18  \DTX{um-code-epilogue.dtx}
19  \DTX{um-code-primes.dtx}
20  \DTX{um-code-sscript.dtx}
21  \DTX{um-code-compatible.dtx}
22  \DTX{um-code-alphabets.dtx}
23 }
24 (/dtx)
```

Now exit if we're using plain T<sub>E</sub>X; this would usually be the case when loading this file with `unicode-math.ins`.

```
25 (*dtx)
26 \def\tmpa{plain}
27 \ifx\tmpa\fmtname\expandafter\endinput\fi
28 (/dtx)
```

Declare the package version and date. For loading this file directly as a dtx file, `\fileversion` and `\filedate` will be set correctly when using `\GetFileInfo` without having to load the package directly.

```
29 (base)\ProvidesPackage{unicode-math}
30 (package&XE)\ProvidesPackage{unicode-math-xetex}
31 (package&LU)\ProvidesPackage{unicode-math-luatex}
32 (*dtx)
33 \ProvidesFile{unicode-math.dtx}
34 (/dtx)
35 (*base|package)
```

36 [2017/11/18 v0.8i Unicode maths in XeLaTeX and LuaLaTeX]  
37 </base|package>

## File II

# um-code-base.dtx

## 2 *The unicode-math.sty loading file*

The `unicode-math.sty` file is a stub which loads necessary packages and then bifurcates into a XeTeX- or LuaTeX-specific version of the package.

```
1 (*base)
```

Bail early if necessary.

```
2 \ifdefined\XeTeXversion
3   \ifdim\number\XeTeXversion\XeTeXrevision in<0.9998in%
4     \PackageError{unicode-math}{%
5       Cannot run with this version of XeTeX!\MessageBreak
6       You need XeTeX 0.9998 or newer.%
7     }\@ehd
8   \fi
9 \else\ifdefined\luatexversion
10  \ifnum\luatexversion<64%
11    \PackageError{unicode-math}{%
12      Cannot run with this version of LuaTeX!\MessageBreak
13      You need LuaTeX 0.64 or newer.%
14    }\@ehd
15  \fi
16 \else
17   \PackageError{unicode-math}{%
18     Cannot be run with pdfLaTeX!\MessageBreak
19     Use XeLaTeX or LuaLaTeX instead.%
20   }\@ehd
21 \fi\fi
```

*Packages* Assuming people are running up-to-date packages.

```
22 \RequirePackage{expl3,xparse,l3keys2e}
23 \RequirePackage{fontspec}
24 \RequirePackage{ucharcat}
25 \RequirePackage{fix-cm} % avoid some warnings (still necessary? check...)
26 \RequirePackage{filehook}
```

*Bifurcate*

```
27 \ExplSyntaxOn
28 \sys_if_engine luatex:T { \RequirePackageWithOptions{unicode-math-luatex} }
29 \sys_if_engine xetex:T { \RequirePackageWithOptions{unicode-math-xetex} }
30 \ExplSyntaxOff
31 (</base)
```

## File III

# um-code-opening.dtx

### 3 *Start of the package code*

The prefix for unicode-math is um:

```
1 <@=um>
2 <*package>
3 <*LU>
4 \RequirePackage{lualatex-math}
5 </LU>
6 \ExplSyntaxOn
```

#### 3.1 *expl3 variants*

Variants needed from expl3:

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

For fontspec:

```
8 \cs_generate_variant:Nn \fontspec_set_family:Nnn {Nx}
9 \cs_generate_variant:Nn \fontspec_set_fontface:NNnn {NNx}
```

#### 3.2 *Primitive font commands*

What might end up being provided by the kernel.

```
\@@_glyph_if_exist:NnTF
```

```
10 \prg_new_conditional:Nnn \@@_glyph_if_exist:Nn {p,TF,T,F}
11 {
12   \etex_iffontchar:D #1 #2 \scan_stop:
13   \prg_return_true:
14   \else:
15     \prg_return_false:
16   \fi:
17 }
```

##### 3.2.1 *Mathcode and friends*

```
\@@_set_mathcode:nnnn
\@@_set_mathcode:nnn
```

These are all wrappers for the primitive commands that take numerical input only.

```
18 \cs_set:Npn \@@_set_mathcode:nnnn #1#2#3#4
19 {
20   \Umathcode \int_eval:n {#1} =
21   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
22 }
```

```

23 \cs_set:Npn \@@_set_mathcode:nnn #1#2#3
24 {
25   \Umathcode \int_eval:n {#1} =
26   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
27 }

```

\@@\_set\_mathchar:NNnn

\@@\_set\_mathchar:cNnn

```

28 \cs_set:Npn \@@_set_mathchar:NNnn #1#2#3#4
29 {
30   \Umathchardef #1 =
31   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
32 }

```

```

33 \cs_generate_variant:Nn \@@_set_mathchar:NNnn {c}

```

\@@\_set\_delcode:nnn

```

34 \cs_new:Nn \@@_set_delcode:nnn
35 {
36   \Udelcode#2 = \csname sym#1\endcsname #3 \scan_stop:
37 }

```

\@@\_radical:nn

```

38 \cs_new:Nn \@@_radical:nn
39 {
40   \Uradical \csname sym#1\endcsname #2 \scan_stop:
41 }

```

\@@\_delimiter:Nnn

```

42 \cs_new:Nn \@@_delimiter:Nnn
43 {
44   \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
45 }

```

\@@\_accent:nnn

```

46 \cs_new:Nn \@@_accent:nnn
47 {
48   \Umathaccent #1~ \mathchar@type\mathaccent \use:c { sym #2 } #3 \scan_stop:
49 }

```

\@@\_char\_gmake\_mathactive:N

\@@\_char\_gmake\_mathactive:n

```

50 \cs_new:Nn \@@_char_gmake_mathactive:N
51 {
52   \global\mathcode `#1 = "8000 \scan_stop:
53 }
54 \cs_new:Nn \@@_char_gmake_mathactive:n
55 {
56   \global\mathcode #1 = "8000 \scan_stop:
57 }

```

### 3.2.2 Font parameters

`\@@_copy_fontparam:nnn`

```
58 \cs_new:Nn \@@_copy_fontparam:nnn
59 {
60   \fontdimen #1 \font = \@@_get_fontparam:nn {#2} {#3}
61 }
```

`\@@_zero_fontparam:n`

```
62 \cs_new:Nn \@@_zero_fontparam:n
63 {
64   \fontdimen #1 \font = 0pt\relax
65 }
```

`\@@_get_fontparam:nn`

```
66 \cs_new:Nn \@@_get_fontparam:nn
67 {
68 (XE)   \the\fontdimen#1\l_@@_font\relax
69 (LU)   \directlua{fontspec.mathfontdimen("l_@@_font", "#2")}
70 }
```

### 3.3 Alphabet Unicode positions (USVs)

Before we begin, let's define the positions of the various Unicode alphabets so that our code is a little more readable.<sup>1</sup>

`\usv_set:nnn, \@@_to_usv:nn` Rather than 'readable', in the end, this makes the code more extensible.

```
71 \cs_new:Nn \usv_set:nnn { \tl_const:cn { c_@@_#1_#2_usv } {#3} }
72 \cs_new:Nn \@@_to_usv:nn { \use:c { c_@@_#1_#2_usv } }
```

[ TF]@\_usv\_if\_exist:nn

```
73 \prg_new_conditional:Nnn \@@_usv_if_exist:nn {T,F,TF}
74 {
75   \cs_if_exist:cTF { c_@@_#1_#2_usv }
76   \prg_return_true: \prg_return_false:
77 }
```

### 3.4 Overcoming \@onlypreamble

The requirement of only setting up the maths fonts in the preamble is lifted. (Perhaps unwisely.)

```
78 \tl_map_inline:nn
79 {
80   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
81   \DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
82   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
83   \version@list\version@elt\alpha@list\alpha@elt
```

<sup>1</sup>'u.s.v.' stands for 'Unicode scalar value'.

```

84 \restore@mathversion\init@restore@version\dorestore@version\process@table
85 \new@mathversion\DeclareSymbolFont\group@list\group@elt
86 \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
87 \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
88 \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
89 \set@mathsymbol\DeclareMathDelimiter\@xx\DeclareMathDelimiter
90 \@DeclareMathDelimiter\@x\DeclareMathDelimiter\set@mathdelimiter
91 \set@@mathdelimiter\DeclareMathRadical\mathchar@type
92 \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
93 }
94 {
95 \tl_remove_once:Nn \@preamblecmds {\do#1}
96 }
97 </package>

```

## File IV

# um-code-variables.dtx

### 4 *Variable initialisation*

1 *<package>*

*Conditionals* True if using a proper OpenType font with unicode maths

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

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

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

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

For math-style:

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

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

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

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

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

For bold-style:

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

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

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

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

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

For sans-style:

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

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

For assorted package options:

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

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

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

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

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

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

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

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

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

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

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

*Variables*

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

For displaying in warning messages, etc.:

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

```

31 \tl_const:Nn \c_@@_math_alphabet_name_Latin_tl {Latin,~uppercase}
32 \tl_const:Nn \c_@@_math_alphabet_name_greek_tl {Greek,~lowercase}
33 \tl_const:Nn \c_@@_math_alphabet_name_Greek_tl {Greek,~uppercase}
34 \tl_const:Nn \c_@@_math_alphabet_name_num_tl {Numerals}
35 \tl_const:Nn \c_@@_math_alphabet_name_misc_tl {Misc.}

36 \tl_new:N \l_@@_mathstyle_tl
37 \tl_new:N \l_@@_radicals_tl
38 \tl_new:N \l_@@_nolimits_tl

```

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

```

39 \tl_new:N \g_@@_operator_mathfont_tl

40 \seq_new:N \l_@@_missing_alph_seq
41 \seq_new:N \l_@@_mathalph_seq
42 \seq_new:N \l_@@_char_range_seq
43 \seq_new:N \l_@@_mclass_range_seq
44 \seq_new:N \l_@@_cmd_range_seq

```

`\g_@@_alphabets_seq` Each of math ‘style’ (bfup, sfit, etc.) 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.

```

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

47 \clist_new:N \g_@@_named_ranges_clist
48 \clist_new:N \g_@@_char_nrange_clist
49 \clist_new:N \g_@@_unknown_keys_clist
50 \clist_new:N \g_@@_alphabet_clist

```

`\g_@@_mathclasses_seq` Every math class.

```

51 \seq_new:N \g_@@_mathclasses_seq
52 \seq_set_from_clist:Nn \g_@@_mathclasses_seq
53 {
54   \mathord,\mathalpha,\mathbin,\mathrel,\mathpunct,
55   \mathop,
56   \mathopen,\mathclose,
57   \mathfence,\mathover,\mathunder,
58   \mathaccent,\mathbotaccent,\mathaccentwide,\mathbotaccentwide
59 }

```

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

```

60 \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 “bfup” 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!

```

61 \seq_new:N \g_@@_named_ranges_seq
62 \seq_new:N \g_@@_mathstyles_seq

```

```
63 \muskip_new:N \g_@@_primekern_muskip
64 \muskip_gset:Nn \g_@@_primekern_muskip { -\thinmuskip/2 }% arbitrary
65 \int_new:N \l_@@_primecount_int
66 \prop_new:N \g_@@_supers_prop
67 \prop_new:N \g_@@_subs_prop
68 \tl_new:N \l_not_token_name_tl

69 \tl_new:N \g_@@_slash_delimiter_usv
70 \tl_new:N \g_@@_mathtable_tl
71 \tl_new:N \g_@@_fontname_tl
72 \tl_new:N \g_@@_mversion_tl
73 \tl_new:N \g_@@_symfont_tl
74 \tl_new:N \g_@@_font_keyval_tl
75 \tl_new:N \g_@@_family_tl
76 \tl_new:N \g_@@_style_tl
77 \tl_new:N \g_@@_remap_style_tl

78 </package>
```

## File V

# um-code-api.dtx

## 5 *Programmers' interface*

1 *<package>*

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

2 `\cs_new:Nn \unimath_get_mathstyle:`

3 `{`

4 `\tl_use:N \l_@_mathstyle_tl`

5 `}`

6 *</package>*

## File VI

# um-code-ui.dtx

## 6 *The user interface commands*

1 *(\*package)*

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

```
2 \NewDocumentCommand \unimathsetup {m} { \keys_set:nn {unicode-math} {#1} }
```

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

#2 : font name

[#3]: font features

```
3 \NewDocumentCommand \setmathfont { O{} m O{} }
```

```
4 {
```

```
5   \@@_setmathfont:nn {#1,#3} {#2}
```

```
6 }
```

`\setmathfontface`

```
7 \NewDocumentCommand \setmathfontface { m O{} m O{} }
```

```
8 {
```

```
9   \@@_setmathfontface:Nnn #1 {#2,#4} {#3}
```

```
10 }
```

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.

```
11 \onlypreamble \setmathfontface
```

`\setoperatorfont` TODO: add check?

```
12 \NewDocumentCommand \setoperatorfont {m}
```

```
13 {
```

```
14   \tl_set:Nn \g_@@_operator_mathfont_tl {#1}
```

```
15 }
```

```
16 \setoperatorfont{\mathrm}
```

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

```
17 \NewDocumentCommand \addnolimits {m}
```

```
18 {
```

```
19   \tl_put_right:Nn \l_@@_nolimits_tl {#1}
```

```
20 }
```

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

```
21 \NewDocumentCommand \removenolimits {m}
```

```
22 {
```

```
23   \tl_remove_all:Nn \l_@@_nolimits_tl {#1}
```

```
24 }
```

25 `</package>`

## File VII

# um-code-pkgopt.dtx

## 7 *setup and package options*

1 *(\*package)*

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

```
2 \cs_new:Nn \@@_keys_choices:nn
3 {
4   \cs_set:Npn \@@_keys_choices_fn:nn { \@@_keys_choices_aux:nnn {#1} }
5   \use:x
6   {
7     \exp_not:N \keys_define:nn {unicode-math}
8     {
9       #1 .choice: ,
10      \@@_tl_map_dbl:nN {#2} \@@_keys_choices_fn:nn
11    }
12  }
13 }
14 \cs_new:Nn \@@_keys_choices_aux:nnn { #1 / #2 .code:n = { \exp_not:n {#3} } , }
15 \cs_new:Nn \@@_tl_map_dbl:nN
16 {
17   \__@@_tl_map_dbl:Nnn #2 #1 \q_recursion_tail {}{} \q_recursion_stop
18 }
19 \cs_new:Nn \__@@_tl_map_dbl:Nnn
20 {
21   \quark_if_recursion_tail_stop:n {#2}
22   \quark_if_recursion_tail_stop:n {#3}
23   #1 {#2} {#3}
24   \__@@_tl_map_dbl:Nnn #1
25 }
```

### *Compatibility*

```
26 \@@_keys_choices:nn {mathup}
27 {
28   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
29   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
30 }
31 \@@_keys_choices:nn {mathrm}
32 {
33   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
34   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
35 }
36 \@@_keys_choices:nn {mathit}
```

```

37 {
38 {sym} { \bool_set_false:N \g_@@_mathit_text_bool }
39 {text} { \bool_set_true:N \g_@@_mathit_text_bool }
40 }
41 \@@_keys_choices:nn {mathbf}
42 {
43 {sym} { \bool_set_false:N \g_@@_mathbf_text_bool }
44 {text} { \bool_set_true:N \g_@@_mathbf_text_bool }
45 }
46 \@@_keys_choices:nn {mathsf}
47 {
48 {sym} { \bool_set_false:N \g_@@_mathsf_text_bool }
49 {text} { \bool_set_true:N \g_@@_mathsf_text_bool }
50 }
51 \@@_keys_choices:nn {mathtt}
52 {
53 {sym} { \bool_set_false:N \g_@@_mathtt_text_bool }
54 {text} { \bool_set_true:N \g_@@_mathtt_text_bool }
55 }

```

### *math-style*

```

56 \@@_keys_choices:nn {normal-style}
57 {
58   {ISO} {
59     \bool_set_false:N \g_@@_literal_bool
60     \bool_set_false:N \g_@@_upGreek_bool
61     \bool_set_false:N \g_@@_upgreek_bool
62     \bool_set_false:N \g_@@_upLatin_bool
63     \bool_set_false:N \g_@@_uplatin_bool
64   }
65   {TeX} {
66     \bool_set_false:N \g_@@_literal_bool
67     \bool_set_true:N \g_@@_upGreek_bool
68     \bool_set_false:N \g_@@_upgreek_bool
69     \bool_set_false:N \g_@@_upLatin_bool
70     \bool_set_false:N \g_@@_uplatin_bool
71   }
72   {french} {
73     \bool_set_false:N \g_@@_literal_bool
74     \bool_set_true:N \g_@@_upGreek_bool
75     \bool_set_true:N \g_@@_upgreek_bool
76     \bool_set_true:N \g_@@_upLatin_bool
77     \bool_set_false:N \g_@@_uplatin_bool
78   }
79   {upright} {
80     \bool_set_false:N \g_@@_literal_bool
81     \bool_set_true:N \g_@@_upGreek_bool
82     \bool_set_true:N \g_@@_upgreek_bool
83     \bool_set_true:N \g_@@_upLatin_bool

```

```

84         \bool_set_true:N \g_@@_uplatin_bool
85     }
86     {literal} {
87         \bool_set_true:N \g_@@_literal_bool
88     }
89 }
90 \@@_keys_choices:nn {math-style}
91 {
92     {ISO} {
93         \unimathsetup { nabla=upright, partial=italic,
94             normal-style=ISO, bold-style=ISO, sans-style=italic }
95     }
96     {TeX} {
97         \unimathsetup { nabla=upright, partial=italic,
98             normal-style=TeX, bold-style=TeX, sans-style=upright }
99     }
100    {french} {
101        \unimathsetup { nabla=upright, partial=upright,
102            normal-style=french, bold-style=upright, sans-style=upright }
103    }
104    {upright} {
105        \unimathsetup { nabla=upright, partial=upright,
106            normal-style=upright, bold-style=upright, sans-style=upright }
107    }
108    {literal} {
109        \unimathsetup { colon=literal, nabla=literal, partial=literal,
110            normal-style=literal, bold-style=literal, sans-style=literal }
111    }
112 }

```

### *bold-style*

```

113 \@@_keys_choices:nn {bold-style}
114 {
115     {ISO} {
116         \bool_set_false:N \g_@@_bfliteral_bool
117         \bool_set_false:N \g_@@_bfupgreek_bool
118         \bool_set_false:N \g_@@_bfupgreek_bool
119         \bool_set_false:N \g_@@_bfuplatin_bool
120         \bool_set_false:N \g_@@_bfuplatin_bool
121     }
122     {TeX} {
123         \bool_set_false:N \g_@@_bfliteral_bool
124         \bool_set_true:N \g_@@_bfupgreek_bool
125         \bool_set_false:N \g_@@_bfupgreek_bool
126         \bool_set_true:N \g_@@_bfuplatin_bool
127         \bool_set_true:N \g_@@_bfuplatin_bool
128     }
129     {upright} {

```

```

130         \bool_set_false:N \g_@@_bfliteral_bool
131         \bool_set_true:N \g_@@_bfupGreek_bool
132         \bool_set_true:N \g_@@_bfupgreek_bool
133         \bool_set_true:N \g_@@_bfupLatin_bool
134         \bool_set_true:N \g_@@_bfuplatin_bool
135     }
136     {literal} {
137         \bool_set_true:N \g_@@_bfliteral_bool
138     }
139 }

```

### *sans-style*

```

140 \@@_keys_choices:nn {sans-style}
141 {
142     {italic} { \bool_set_false:N \g_@@_upsans_bool }
143     {upright} { \bool_set_true:N \g_@@_upsans_bool }
144     {literal} { \bool_set_true:N \g_@@_sfliteral_bool }
145 }

```

### *Nabla and partial*

```

146 \@@_keys_choices:nn {nabla}
147 {
148     {upright} {
149         \bool_set_false:N \g_@@_literal_Nabla_bool
150         \bool_set_true:N \g_@@_upNabla_bool
151     }
152     {italic} {
153         \bool_set_false:N \g_@@_literal_Nabla_bool
154         \bool_set_false:N \g_@@_upNabla_bool
155     }
156     {literal} { \bool_set_true:N \g_@@_literal_Nabla_bool }
157 }
158 \@@_keys_choices:nn {partial}
159 {
160     {upright} {
161         \bool_set_false:N \g_@@_literal_partial_bool
162         \bool_set_true:N \g_@@_uppartial_bool
163     }
164     {italic} {
165         \bool_set_false:N \g_@@_literal_partial_bool
166         \bool_set_false:N \g_@@_uppartial_bool
167     }
168     {literal} { \bool_set_true:N \g_@@_literal_partial_bool }
169 }

```

### *Colon style*

```

170 \@@_keys_choices:nn {colon}

```

```

171 {
172 {literal} { \bool_set_true:N \g_@@_literal_colon_bool }
173 {TeX} { \bool_set_false:N \g_@@_literal_colon_bool }
174 }

```

### *Slash delimiter style*

```

175 \@@_keys_choices:nn {slash-delimiter}
176 {
177 {ascii} { \tl_set:Nn \g_@@_slash_delimiter_usv {"002F} }
178 {frac} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2044} }
179 {div} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2215} }
180 }

```

### *Active fraction style*

```

181 \@@_keys_choices:nn {active-frac}
182 {
183 {small}
184 {
185 \cs_if_exist:NTF \tfrac
186 { \bool_set_true:N \l_@@_smallfrac_bool }
187 {
188 \@@_warning:n {no-tfrac}
189 \bool_set_false:N \l_@@_smallfrac_bool
190 }
191 \use:c {@@_setup_active_frac:}
192 }
193
194 {normalsize}
195 {
196 \bool_set_false:N \l_@@_smallfrac_bool
197 \use:c {@@_setup_active_frac:}
198 }
199 }

```

### *Debug/tracing*

```

200 \keys_define:nn {unicode-math}
201 {
202 warnings-off .code:n =
203 {
204 \clist_map_inline:nn {#1}
205 { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
206 }
207 }
208 \@@_keys_choices:nn {trace}
209 {
210 {on} {} % default
211 {debug} { \msg_redirect_module:nnn { unicode-math } { log } { warning } }

```

```
212 {off} { \msg_redirect_module:nnn { unicode-math } { log } { none } }  
213 }
```

## 7.1 Defaults

```
214 \unimathsetup {math-style=TeX}  
215 \unimathsetup {slash-delimiter=ascii}  
216 \unimathsetup {trace=off}  
217 \unimathsetup {mathrm=text,mathit=text,mathbf=text,mathsf=text,mathtt=text}  
218 \cs_if_exist:NT \tfrac { \unimathsetup {active-frac=small} }  
219 \ProcessKeysOptions {unicode-math}  
220 </package>
```

## File VIII

# um-code-msg.dtx

## 8 *Error messages*

1 *(\*package)*

Wrapper functions:

2 \cs\_new:Npn \@@\_error:n { \msg\_error:nn {unicode-math} }

3 \cs\_new:Npn \@@\_warning:n { \msg\_warning:nn {unicode-math} }

4 \cs\_new:Npn \@@\_warning:nnn { \msg\_warning:nnxx {unicode-math} }

5 \cs\_new:Npn \@@\_log:n { \msg\_log:nn {unicode-math} }

6 \cs\_new:Npn \@@\_log:nx { \msg\_log:nnx {unicode-math} }

7 \msg\_new:nnn {unicode-math} {no-tfrac}

8 {

9 Small~ fraction~ command~ \protect\tfrac~ not~ defined.\

10 Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.

11 }

12 \msg\_new:nnn {unicode-math} {default-math-font}

13 {

14 Defining~ the~ default~ maths~ font~ as~ '\l\_@@\_fontname\_tl'.

15 }

16 \msg\_new:nnn {unicode-math} {setup-implicit}

17 {

18 Setup~ alphabets:~ implicit~ mode.

19 }

20 \msg\_new:nnn {unicode-math} {setup-explicit}

21 {

22 Setup~ alphabets:~ explicit~ mode.

23 }

24 \msg\_new:nnn {unicode-math} {alph-initialise}

25 {

26 Initialising~ \@backslashchar math#1.

27 }

28 \msg\_new:nnn {unicode-math} {setup-alph}

29 {

30 Setup~ alphabet:~ #1.

31 }

32 \msg\_new:nnn {unicode-math} {no-alphabet}

33 {

34 I~ am~ trying~ to~ set~ up~ alphabet~"#1"~ but~ there~ are~ no~ configura-  
tion~ settings~ for~ it.~

35 (See~ source~ file~ "unicode-math-alphabets.dtx"~ to~ debug.)

36 }

37 \msg\_new:nnn { unicode-math } { no-named-range }

38 {

39 I~ am~ trying~ to~ define~ new~ alphabet~ "#2"~ in~ range~ "#1",~ but~ range~ "#1"~ hasn't~ been~ de-  
fined~ yet.

```

40 }
41 \msg_new:nnn { unicode-math } { missing-alphabets }
42 {
43   Missing~math~alphabets~in~font~ "\fontname\l_@_font" \ \ \
44   \seq_map_function:NN \l_@_missing_alph_seq \@@_print_indent:n
45 }
46 \cs_new:Nn \@@_print_indent:n { \space\space\space\space #1 \ \ }
47 \msg_new:nnn { unicode-math } { macro-expected }
48 {
49   I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
50 }
51 \msg_new:nnn { unicode-math } { wrong-meaning }
52 {
53   I've~ expected~ #1~ to~ have~ the~ meaning~ #3,~ but~ it~ has~ the~ meaning~ #2.
54 }
55 \msg_new:nnn { unicode-math } { patch-macro }
56 {
57   I'm~ going~ to~ patch~ macro~ #1.
58 }
59 \msg_new:nnn { unicode-math } { mathtools-overbracket } {
60   Using~ \token_to_str:N \overbracket\ and~
61   \token_to_str:N \underbracket\ from~
62   `mathtools'~ package.\ \
63   \ \
64   Use~ \token_to_str:N \Uoverbracket\ and~
65   \token_to_str:N \Uunderbracket\ for~
66   original~ `unicode-math'~ definition.
67 }
68 \msg_new:nnn { unicode-math } { mathtools-colon } {
69   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
70   the~ `mathtools'~ package: \ \ \
71   \ \ \ \ \token_to_str:N \dblcolon,~
72   \token_to_str:N \coloneqq,~
73   \token_to_str:N \Coloneqq,~
74   \token_to_str:N \eqqcolon. \ \ \
75   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
76   commands,~ using~ them~ will~ lead~ to~ inconsistencies.
77 }
78 \msg_new:nnn { unicode-math } { colonequals } {
79   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
80   the~ `colonequals'~ package: \ \ \
81   \ \ \ \ \token_to_str:N \ratio,~
82   \token_to_str:N \coloncolon,~
83   \token_to_str:N \minuscolon, \ \
84   \ \ \ \ \token_to_str:N \colonequals,~
85   \token_to_str:N \equalscolon,~
86   \token_to_str:N \coloncolonequals. \ \ \
87   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
88   commands,~ using~ them~ will~ lead~ to~ inconsistencies.~

```

```
89 Furthermore,~ changing~ \token_to_str:N \colonsep \c_space_tl
90 or~ \token_to_str:N \doublecolonsep \c_space_tl won't~ have~
91 any~ effect~ on~ the~ re-defined~ commands.
92 }
93 </package>
```

## File IX

# um-code-usv.dtx

## 9 *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>2</sup>

```
1 (*package)
```

### *Alphabets*

```
2 \usv_set:nnn {normal} {num} {48}
3 \usv_set:nnn {normal} {Latin}{"1D434}
4 \usv_set:nnn {normal} {latin}{"1D44E}
5 \usv_set:nnn {normal} {Greek}{"1D6E2}
6 \usv_set:nnn {normal} {greek}{"1D6FC}
7 \usv_set:nnn {normal}{\vartheta} {"1D6F3}
8 \usv_set:nnn {normal}{\epsilon} {"1D716}
9 \usv_set:nnn {normal}{\vartheta} {"1D717}
10 \usv_set:nnn {normal}{\varkappa} {"1D718}
11 \usv_set:nnn {normal}{\phi} {"1D719}
12 \usv_set:nnn {normal}{\varrho} {"1D71A}
13 \usv_set:nnn {normal}{\varpi} {"1D71B}
14 \usv_set:nnn {normal} {\Nabla}{"1D6FB}
15 \usv_set:nnn {normal} {\partial}{"1D715}
16
17 \usv_set:nnn {up} {num} {48}
18 \usv_set:nnn {up} {Latin}{65}
19 \usv_set:nnn {up} {latin}{97}
20 \usv_set:nnn {up} {Greek}{"391}
21 \usv_set:nnn {up} {greek}{"3B1}
22 \usv_set:nnn {it} {Latin}{"1D434}
23 \usv_set:nnn {it} {latin}{"1D44E}
24 \usv_set:nnn {it} {Greek}{"1D6E2}
25 \usv_set:nnn {it} {greek}{"1D6FC}
26 \usv_set:nnn {bb} {num} {"1D7D8}
27 \usv_set:nnn {bb} {Latin}{"1D538}
28 \usv_set:nnn {bb} {latin}{"1D552}
29 \usv_set:nnn {scr} {Latin}{"1D49C}
30 \usv_set:nnn {cal} {Latin}{"1D49C}
31 \usv_set:nnn {scr} {latin}{"1D4B6}
32 \usv_set:nnn {frac}{Latin}{"1D504}
33 \usv_set:nnn {frac}{latin}{"1D51E}
34 \usv_set:nnn {sf} {num} {"1D7E2}
35 \usv_set:nnn {sfup}{num} {"1D7E2}
36 \usv_set:nnn {sfit}{num} {"1D7E2}
```

---

<sup>2</sup>'u.s.v.' stands for 'Unicode scalar value'.

37 \usv\_set:nnn {sfup}{Latin}{"1D5A0}  
 38 \usv\_set:nnn {sf} {Latin}{"1D5A0}  
 39 \usv\_set:nnn {sfup}{latin}{"1D5BA}  
 40 \usv\_set:nnn {sf} {latin}{"1D5BA}  
 41 \usv\_set:nnn {sfit}{Latin}{"1D608}  
 42 \usv\_set:nnn {sfit}{latin}{"1D622}  
 43 \usv\_set:nnn {tt} {num} {"1D7F6}  
 44 \usv\_set:nnn {tt} {Latin}{"1D670}  
 45 \usv\_set:nnn {tt} {latin}{"1D68A}

**Bold:**

46 \usv\_set:nnn {bf} {num} {"1D7CE}  
 47 \usv\_set:nnn {bfup} {num} {"1D7CE}  
 48 \usv\_set:nnn {bfit} {num} {"1D7CE}  
 49 \usv\_set:nnn {bfup} {Latin}{"1D400}  
 50 \usv\_set:nnn {bfup} {latin}{"1D41A}  
 51 \usv\_set:nnn {bfup} {Greek}{"1D6A8}  
 52 \usv\_set:nnn {bfup} {greek}{"1D6C2}  
 53 \usv\_set:nnn {bfit} {Latin}{"1D468}  
 54 \usv\_set:nnn {bfit} {latin}{"1D482}  
 55 \usv\_set:nnn {bfit} {Greek}{"1D71C}  
 56 \usv\_set:nnn {bfit} {greek}{"1D736}  
 57 \usv\_set:nnn {bffrak}{Latin}{"1D56C}  
 58 \usv\_set:nnn {bffrak}{latin}{"1D586}  
 59 \usv\_set:nnn {bfscr} {Latin}{"1D4D0}  
 60 \usv\_set:nnn {bfcal} {Latin}{"1D4D0}  
 61 \usv\_set:nnn {bfscr} {latin}{"1D4EA}  
 62 \usv\_set:nnn {bfsf} {num} {"1D7EC}  
 63 \usv\_set:nnn {bfsfup}{num} {"1D7EC}  
 64 \usv\_set:nnn {bfsfit}{num} {"1D7EC}  
 65 \usv\_set:nnn {bfsfup}{Latin}{"1D5D4}  
 66 \usv\_set:nnn {bfsfup}{latin}{"1D5EE}  
 67 \usv\_set:nnn {bfsfup}{Greek}{"1D756}  
 68 \usv\_set:nnn {bfsfup}{greek}{"1D770}  
 69 \usv\_set:nnn {bfsfit}{Latin}{"1D63C}  
 70 \usv\_set:nnn {bfsfit}{latin}{"1D656}  
 71 \usv\_set:nnn {bfsfit}{Greek}{"1D790}  
 72 \usv\_set:nnn {bfsfit}{greek}{"1D7AA}

73 \usv\_set:nnn {bfsf}{Latin}{ \bool\_if:NTF \g\_@@\_upLatin\_bool \g\_@@\_bfsfup\_Latin\_usv \g\_@@\_bfsfit\_Lati  
 74 \usv\_set:nnn {bfsf}{latin}{ \bool\_if:NTF \g\_@@\_uplatin\_bool \g\_@@\_bfsfup\_latin\_usv \g\_@@\_bfsfit\_lati  
 75 \usv\_set:nnn {bfsf}{Greek}{ \bool\_if:NTF \g\_@@\_upGreek\_bool \g\_@@\_bfsfup\_Greek\_usv \g\_@@\_bfsfit\_Gree  
 76 \usv\_set:nnn {bfsf}{greek}{ \bool\_if:NTF \g\_@@\_upgreek\_bool \g\_@@\_bfsfup\_greek\_usv \g\_@@\_bfsfit\_gree  
 77 \usv\_set:nnn {bf} {Latin}{ \bool\_if:NTF \g\_@@\_bfupLatin\_bool \g\_@@\_bfup\_Latin\_usv \g\_@@\_bfit\_Latin\_u  
 78 \usv\_set:nnn {bf} {latin}{ \bool\_if:NTF \g\_@@\_bfuplatin\_bool \g\_@@\_bfup\_latin\_usv \g\_@@\_bfit\_latin\_u  
 79 \usv\_set:nnn {bf} {Greek}{ \bool\_if:NTF \g\_@@\_bfupGreek\_bool \g\_@@\_bfup\_Greek\_usv \g\_@@\_bfit\_Greek\_u  
 80 \usv\_set:nnn {bf} {greek}{ \bool\_if:NTF \g\_@@\_bfupgreek\_bool \g\_@@\_bfup\_greek\_usv \g\_@@\_bfit\_greek\_u

**Greek variants:**

81 \usv\_set:nnn {up}{varTheta} {"3F4}  
 82 \usv\_set:nnn {up}{Digamma} {"3DC}

83 \usv\_set:nnn {up}{epsilon}{"3F5}  
 84 \usv\_set:nnn {up}{vartheta} {"3D1}  
 85 \usv\_set:nnn {up}{varkappa} {"3F0}  
 86 \usv\_set:nnn {up}{phi} {"3D5}  
 87 \usv\_set:nnn {up}{varrho} {"3F1}  
 88 \usv\_set:nnn {up}{varpi} {"3D6}  
 89 \usv\_set:nnn {up}{digamma} {"3DD}

**Bold:**

90 \usv\_set:nnn {bfup}{varTheta} {"1D6B9}  
 91 \usv\_set:nnn {bfup}{Digamma} {"1D7CA}  
 92 \usv\_set:nnn {bfup}{epsilon}{"1D6DC}  
 93 \usv\_set:nnn {bfup}{vartheta} {"1D6DD}  
 94 \usv\_set:nnn {bfup}{varkappa} {"1D6DE}  
 95 \usv\_set:nnn {bfup}{phi} {"1D6DF}  
 96 \usv\_set:nnn {bfup}{varrho} {"1D6E0}  
 97 \usv\_set:nnn {bfup}{varpi} {"1D6E1}  
 98 \usv\_set:nnn {bfup}{digamma} {"1D7CB}

**Italic Greek variants:**

99 \usv\_set:nnn {it}{varTheta} {"1D6F3}  
 100 \usv\_set:nnn {it}{epsilon}{"1D716}  
 101 \usv\_set:nnn {it}{vartheta} {"1D717}  
 102 \usv\_set:nnn {it}{varkappa} {"1D718}  
 103 \usv\_set:nnn {it}{phi} {"1D719}  
 104 \usv\_set:nnn {it}{varrho} {"1D71A}  
 105 \usv\_set:nnn {it}{varpi} {"1D71B}

**Bold italic:**

106 \usv\_set:nnn {bfit}{varTheta} {"1D72D}  
 107 \usv\_set:nnn {bfit}{epsilon}{"1D750}  
 108 \usv\_set:nnn {bfit}{vartheta} {"1D751}  
 109 \usv\_set:nnn {bfit}{varkappa} {"1D752}  
 110 \usv\_set:nnn {bfit}{phi} {"1D753}  
 111 \usv\_set:nnn {bfit}{varrho} {"1D754}  
 112 \usv\_set:nnn {bfit}{varpi} {"1D755}

**Bold sans:**

113 \usv\_set:nnn {bfsfup}{varTheta} {"1D767}  
 114 \usv\_set:nnn {bfsfup}{epsilon}{"1D78A}  
 115 \usv\_set:nnn {bfsfup}{vartheta} {"1D78B}  
 116 \usv\_set:nnn {bfsfup}{varkappa} {"1D78C}  
 117 \usv\_set:nnn {bfsfup}{phi} {"1D78D}  
 118 \usv\_set:nnn {bfsfup}{varrho} {"1D78E}  
 119 \usv\_set:nnn {bfsfup}{varpi} {"1D78F}

**Bold sans italic:**

120 \usv\_set:nnn {bfsfit}{varTheta} {"1D7A1}  
 121 \usv\_set:nnn {bfsfit}{epsilon}{"1D7C4}  
 122 \usv\_set:nnn {bfsfit}{vartheta} {"1D7C5}  
 123 \usv\_set:nnn {bfsfit}{varkappa} {"1D7C6}

124 \usv\_set:nnn {bfsfit}{phi} {"1D7C7}  
 125 \usv\_set:nnn {bfsfit}{varrho} {"1D7C8}  
 126 \usv\_set:nnn {bfsfit}{varpi} {"1D7C9}

**Nabla:**

127 \usv\_set:nnn {up} {Nabla}{"02207}  
 128 \usv\_set:nnn {it} {Nabla}{"1D6FB}  
 129 \usv\_set:nnn {bfup} {Nabla}{"1D6C1}  
 130 \usv\_set:nnn {bfit} {Nabla}{"1D735}  
 131 \usv\_set:nnn {bfsfup}{Nabla}{"1D76F}  
 132 \usv\_set:nnn {bfsfit}{Nabla}{"1D7A9}

**Partial:**

133 \usv\_set:nnn {up} {partial}{"02202}  
 134 \usv\_set:nnn {it} {partial}{"1D715}  
 135 \usv\_set:nnn {bfup} {partial}{"1D6DB}  
 136 \usv\_set:nnn {bfit} {partial}{"1D74F}  
 137 \usv\_set:nnn {bfsfup}{partial}{"1D789}  
 138 \usv\_set:nnn {bfsfit}{partial}{"1D7C3}

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

139 \usv\_set:nnn {up}{B}{`\B}  
 140 \usv\_set:nnn {up}{C}{`\C}  
 141 \usv\_set:nnn {up}{D}{`\D}  
 142 \usv\_set:nnn {up}{E}{`\E}  
 143 \usv\_set:nnn {up}{F}{`\F}  
 144 \usv\_set:nnn {up}{H}{`\H}  
 145 \usv\_set:nnn {up}{I}{`\I}  
 146 \usv\_set:nnn {up}{L}{`\L}  
 147 \usv\_set:nnn {up}{M}{`\M}  
 148 \usv\_set:nnn {up}{N}{`\N}  
 149 \usv\_set:nnn {up}{P}{`\P}  
 150 \usv\_set:nnn {up}{Q}{`\Q}  
 151 \usv\_set:nnn {up}{R}{`\R}  
 152 \usv\_set:nnn {up}{Z}{`\Z}

153 \usv\_set:nnn {it}{B}{"1D435}  
 154 \usv\_set:nnn {it}{C}{"1D436}  
 155 \usv\_set:nnn {it}{D}{"1D437}  
 156 \usv\_set:nnn {it}{E}{"1D438}  
 157 \usv\_set:nnn {it}{F}{"1D439}  
 158 \usv\_set:nnn {it}{H}{"1D43B}  
 159 \usv\_set:nnn {it}{I}{"1D43C}  
 160 \usv\_set:nnn {it}{L}{"1D43F}  
 161 \usv\_set:nnn {it}{M}{"1D440}  
 162 \usv\_set:nnn {it}{N}{"1D441}  
 163 \usv\_set:nnn {it}{P}{"1D443}  
 164 \usv\_set:nnn {it}{Q}{"1D444}  
 165 \usv\_set:nnn {it}{R}{"1D445}

166 \usv\_set:nnn {it}{Z}{"1D44D}  
167 \usv\_set:nnn {up}{d}{`\d}  
168 \usv\_set:nnn {up}{e}{`\e}  
169 \usv\_set:nnn {up}{g}{`\g}  
170 \usv\_set:nnn {up}{h}{`\h}  
171 \usv\_set:nnn {up}{i}{`\i}  
172 \usv\_set:nnn {up}{j}{`\j}  
173 \usv\_set:nnn {up}{o}{`\o}  
174 \usv\_set:nnn {it}{d}{"1D451}  
175 \usv\_set:nnn {it}{e}{"1D452}  
176 \usv\_set:nnn {it}{g}{"1D454}  
177 \usv\_set:nnn {it}{h}{"0210E}  
178 \usv\_set:nnn {it}{i}{"1D456}  
179 \usv\_set:nnn {it}{j}{"1D457}  
180 \usv\_set:nnn {it}{o}{"1D45C}

#### Latin ‘h’:

181 \usv\_set:nnn {bb} {h}{"1D559}  
182 \usv\_set:nnn {tt} {h}{"1D691}  
183 \usv\_set:nnn {scr} {h}{"1D4BD}  
184 \usv\_set:nnn {frak} {h}{"1D525}  
185 \usv\_set:nnn {bfup} {h}{"1D421}  
186 \usv\_set:nnn {bfit} {h}{"1D489}  
187 \usv\_set:nnn {sfup} {h}{"1D5C1}  
188 \usv\_set:nnn {sfit} {h}{"1D629}  
189 \usv\_set:nnn {bffrak}{h}{"1D58D}  
190 \usv\_set:nnn {bfscr} {h}{"1D4F1}  
191 \usv\_set:nnn {bfsfup}{h}{"1D5F5}  
192 \usv\_set:nnn {bfsfit}{h}{"1D65D}

#### Dotless ‘i’ and ‘j’:

193 \usv\_set:nnn {up}{dotlessi}{"00131}  
194 \usv\_set:nnn {up}{dotlessj}{"00237}  
195 \usv\_set:nnn {it}{dotlessi}{"1D6A4}  
196 \usv\_set:nnn {it}{dotlessj}{"1D6A5}

#### Blackboard:

197 \usv\_set:nnn {bb}{C}{"2102}  
198 \usv\_set:nnn {bb}{H}{"210D}  
199 \usv\_set:nnn {bb}{N}{"2115}  
200 \usv\_set:nnn {bb}{P}{"2119}  
201 \usv\_set:nnn {bb}{Q}{"211A}  
202 \usv\_set:nnn {bb}{R}{"211D}  
203 \usv\_set:nnn {bb}{Z}{"2124}  
204 \usv\_set:nnn {up}{Pi} {"003A0}  
205 \usv\_set:nnn {up}{pi} {"003C0}  
206 \usv\_set:nnn {up}{Gamma} {"00393}  
207 \usv\_set:nnn {up}{gamma} {"003B3}  
208 \usv\_set:nnn {up}{summation}{"02211}  
209 \usv\_set:nnn {it}{Pi} {"1D6F1}

```

210 \usv_set:nnn {it}{pi}      {"1D70B}
211 \usv_set:nnn {it}{Gamma} {"1D6E4}
212 \usv_set:nnn {it}{gamma} {"1D6FE}
213 \usv_set:nnn {bb}{Pi}    {"0213F}
214 \usv_set:nnn {bb}{pi}    {"0213C}
215 \usv_set:nnn {bb}{Gamma} {"0213E}
216 \usv_set:nnn {bb}{gamma} {"0213D}
217 \usv_set:nnn {bb}{summation}{"02140}

```

Italic blackboard:

```

218 \usv_set:nnn {bbit}{D}{"2145}
219 \usv_set:nnn {bbit}{d}{"2146}
220 \usv_set:nnn {bbit}{e}{"2147}
221 \usv_set:nnn {bbit}{i}{"2148}
222 \usv_set:nnn {bbit}{j}{"2149}

```

Script exceptions:

```

223 \usv_set:nnn {scr}{B}{"212C}
224 \usv_set:nnn {scr}{E}{"2130}
225 \usv_set:nnn {scr}{F}{"2131}
226 \usv_set:nnn {scr}{H}{"210B}
227 \usv_set:nnn {scr}{I}{"2110}
228 \usv_set:nnn {scr}{L}{"2112}
229 \usv_set:nnn {scr}{M}{"2133}
230 \usv_set:nnn {scr}{R}{"211B}
231 \usv_set:nnn {scr}{e}{"212F}
232 \usv_set:nnn {scr}{g}{"210A}
233 \usv_set:nnn {scr}{o}{"2134}

234 \usv_set:nnn {cal}{B}{"212C}
235 \usv_set:nnn {cal}{E}{"2130}
236 \usv_set:nnn {cal}{F}{"2131}
237 \usv_set:nnn {cal}{H}{"210B}
238 \usv_set:nnn {cal}{I}{"2110}
239 \usv_set:nnn {cal}{L}{"2112}
240 \usv_set:nnn {cal}{M}{"2133}
241 \usv_set:nnn {cal}{R}{"211B}

```

Fraktur exceptions:

```

242 \usv_set:nnn {frak}{C}{"212D}
243 \usv_set:nnn {frak}{H}{"210C}
244 \usv_set:nnn {frak}{I}{"2111}
245 \usv_set:nnn {frak}{R}{"211C}
246 \usv_set:nnn {frak}{Z}{"2128}

```

```

247 </package>

```

## 9.1 *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.

248 *(\*stix)*

### *Upright*

249 \usv\_set:nnn {stixsfup}{partial}{E17C}  
250 \usv\_set:nnn {stixsfup}{Greek}{E17D}  
251 \usv\_set:nnn {stixsfup}{greek}{E196}  
252 \usv\_set:nnn {stixsfup}{varTheta}{E18E}  
253 \usv\_set:nnn {stixsfup}{epsilon}{E1AF}  
254 \usv\_set:nnn {stixsfup}{vartheta}{E1B0}  
255 \usv\_set:nnn {stixsfup}{varkappa}{0000} % ???  
256 \usv\_set:nnn {stixsfup}{phi}{E1B1}  
257 \usv\_set:nnn {stixsfup}{varrho}{E1B2}  
258 \usv\_set:nnn {stixsfup}{varpi}{E1B3}  
259 \usv\_set:nnn {stixupslash}{Greek}{E2FC}

### *Italic*

260 \usv\_set:nnn {stixbbit}{A}{E154}  
261 \usv\_set:nnn {stixbbit}{B}{E155}  
262 \usv\_set:nnn {stixbbit}{E}{E156}  
263 \usv\_set:nnn {stixbbit}{F}{E157}  
264 \usv\_set:nnn {stixbbit}{G}{E158}  
265 \usv\_set:nnn {stixbbit}{I}{E159}  
266 \usv\_set:nnn {stixbbit}{J}{E15A}  
267 \usv\_set:nnn {stixbbit}{K}{E15B}  
268 \usv\_set:nnn {stixbbit}{L}{E15C}  
269 \usv\_set:nnn {stixbbit}{M}{E15D}  
270 \usv\_set:nnn {stixbbit}{O}{E15E}  
271 \usv\_set:nnn {stixbbit}{S}{E15F}  
272 \usv\_set:nnn {stixbbit}{T}{E160}  
273 \usv\_set:nnn {stixbbit}{U}{E161}  
274 \usv\_set:nnn {stixbbit}{V}{E162}  
275 \usv\_set:nnn {stixbbit}{W}{E163}  
276 \usv\_set:nnn {stixbbit}{X}{E164}  
277 \usv\_set:nnn {stixbbit}{Y}{E165}  
  
278 \usv\_set:nnn {stixbbit}{a}{E166}  
279 \usv\_set:nnn {stixbbit}{b}{E167}  
280 \usv\_set:nnn {stixbbit}{c}{E168}  
281 \usv\_set:nnn {stixbbit}{f}{E169}  
282 \usv\_set:nnn {stixbbit}{g}{E16A}  
283 \usv\_set:nnn {stixbbit}{h}{E16B}  
284 \usv\_set:nnn {stixbbit}{k}{E16C}  
285 \usv\_set:nnn {stixbbit}{l}{E16D}  
286 \usv\_set:nnn {stixbbit}{m}{E16E}  
287 \usv\_set:nnn {stixbbit}{n}{E16F}  
288 \usv\_set:nnn {stixbbit}{o}{E170}  
289 \usv\_set:nnn {stixbbit}{p}{E171}  
290 \usv\_set:nnn {stixbbit}{q}{E172}

291 \usv\_set:nnn {stixbbit}{r}{E173}  
 292 \usv\_set:nnn {stixbbit}{s}{E174}  
 293 \usv\_set:nnn {stixbbit}{t}{E175}  
 294 \usv\_set:nnn {stixbbit}{u}{E176}  
 295 \usv\_set:nnn {stixbbit}{v}{E177}  
 296 \usv\_set:nnn {stixbbit}{w}{E178}  
 297 \usv\_set:nnn {stixbbit}{x}{E179}  
 298 \usv\_set:nnn {stixbbit}{y}{E17A}  
 299 \usv\_set:nnn {stixbbit}{z}{E17B}  
  
 300 \usv\_set:nnn {stixsfit}{Numerals}{E1B4}  
 301 \usv\_set:nnn {stixsfit}{partial}{E1BE}  
 302 \usv\_set:nnn {stixsfit}{Greek}{E1BF}  
 303 \usv\_set:nnn {stixsfit}{greek}{E1D8}  
 304 \usv\_set:nnn {stixsfit}{varTheta}{E1D0}  
 305 \usv\_set:nnn {stixsfit}{epsilon}{E1F1}  
 306 \usv\_set:nnn {stixsfit}{vartheta}{E1F2}  
 307 \usv\_set:nnn {stixsfit}{varkappa}{0000} % ???  
 308 \usv\_set:nnn {stixsfit}{phi}{E1F3}  
 309 \usv\_set:nnn {stixsfit}{varrho}{E1F4}  
 310 \usv\_set:nnn {stixsfit}{varpi}{E1F5}  
  
 311 \usv\_set:nnn {stixcal}{Latin}{E22D}  
 312 \usv\_set:nnn {stixcal}{num}{E262}  
 313 \usv\_set:nnn {scr}{num}{48}  
 314 \usv\_set:nnn {it}{num}{48}  
  
 315 \usv\_set:nnn {stixsfitslash}{Latin}{E294}  
 316 \usv\_set:nnn {stixsfitslash}{latin}{E2C8}  
 317 \usv\_set:nnn {stixsfitslash}{greek}{E32C}  
 318 \usv\_set:nnn {stixsfitslash}{epsilon}{E37A}  
 319 \usv\_set:nnn {stixsfitslash}{vartheta}{E35E}  
 320 \usv\_set:nnn {stixsfitslash}{varkappa}{E374}  
 321 \usv\_set:nnn {stixsfitslash}{phi}{E360}  
 322 \usv\_set:nnn {stixsfitslash}{varrho}{E376}  
 323 \usv\_set:nnn {stixsfitslash}{varpi}{E362}  
 324 \usv\_set:nnn {stixsfitslash}{digamma}{E36A}

*Bold*

325 \usv\_set:nnn {stixbfupslash}{Greek}{E2FD}  
 326 \usv\_set:nnn {stixbfupslash}{Digamma}{E369}  
  
 327 \usv\_set:nnn {stixbfbb}{A}{E38A}  
 328 \usv\_set:nnn {stixbfbb}{B}{E38B}  
 329 \usv\_set:nnn {stixbfbb}{E}{E38D}  
 330 \usv\_set:nnn {stixbfbb}{F}{E38E}  
 331 \usv\_set:nnn {stixbfbb}{G}{E38F}  
 332 \usv\_set:nnn {stixbfbb}{I}{E390}  
 333 \usv\_set:nnn {stixbfbb}{J}{E391}  
 334 \usv\_set:nnn {stixbfbb}{K}{E392}  
 335 \usv\_set:nnn {stixbfbb}{L}{E393}

336 \usv\_set:nnn {stixbfbb}{M}{E394}  
337 \usv\_set:nnn {stixbfbb}{O}{E395}  
338 \usv\_set:nnn {stixbfbb}{S}{E396}  
339 \usv\_set:nnn {stixbfbb}{T}{E397}  
340 \usv\_set:nnn {stixbfbb}{U}{E398}  
341 \usv\_set:nnn {stixbfbb}{V}{E399}  
342 \usv\_set:nnn {stixbfbb}{W}{E39A}  
343 \usv\_set:nnn {stixbfbb}{X}{E39B}  
344 \usv\_set:nnn {stixbfbb}{Y}{E39C}  
  
345 \usv\_set:nnn {stixbfbb}{a}{E39D}  
346 \usv\_set:nnn {stixbfbb}{b}{E39E}  
347 \usv\_set:nnn {stixbfbb}{c}{E39F}  
348 \usv\_set:nnn {stixbfbb}{f}{E3A2}  
349 \usv\_set:nnn {stixbfbb}{g}{E3A3}  
350 \usv\_set:nnn {stixbfbb}{h}{E3A4}  
351 \usv\_set:nnn {stixbfbb}{k}{E3A7}  
352 \usv\_set:nnn {stixbfbb}{l}{E3A8}  
353 \usv\_set:nnn {stixbfbb}{m}{E3A9}  
354 \usv\_set:nnn {stixbfbb}{n}{E3AA}  
355 \usv\_set:nnn {stixbfbb}{o}{E3AB}  
356 \usv\_set:nnn {stixbfbb}{p}{E3AC}  
357 \usv\_set:nnn {stixbfbb}{q}{E3AD}  
358 \usv\_set:nnn {stixbfbb}{r}{E3AE}  
359 \usv\_set:nnn {stixbfbb}{s}{E3AF}  
360 \usv\_set:nnn {stixbfbb}{t}{E3B0}  
361 \usv\_set:nnn {stixbfbb}{u}{E3B1}  
362 \usv\_set:nnn {stixbfbb}{v}{E3B2}  
363 \usv\_set:nnn {stixbfbb}{w}{E3B3}  
364 \usv\_set:nnn {stixbfbb}{x}{E3B4}  
365 \usv\_set:nnn {stixbfbb}{y}{E3B5}  
366 \usv\_set:nnn {stixbfbb}{z}{E3B6}  
  
367 \usv\_set:nnn {stixbfsfup}{Numerals}{E3B7}

*Bold Italic*

368 \usv\_set:nnn {stixbfsfit}{Numerals}{E1F6}  
  
369 \usv\_set:nnn {stixfbbit}{A}{E200}  
370 \usv\_set:nnn {stixfbbit}{B}{E201}  
371 \usv\_set:nnn {stixfbbit}{E}{E203}  
372 \usv\_set:nnn {stixfbbit}{F}{E204}  
373 \usv\_set:nnn {stixfbbit}{G}{E205}  
374 \usv\_set:nnn {stixfbbit}{I}{E206}  
375 \usv\_set:nnn {stixfbbit}{J}{E207}  
376 \usv\_set:nnn {stixfbbit}{K}{E208}  
377 \usv\_set:nnn {stixfbbit}{L}{E209}  
378 \usv\_set:nnn {stixfbbit}{M}{E20A}  
379 \usv\_set:nnn {stixfbbit}{O}{E20B}  
380 \usv\_set:nnn {stixfbbit}{S}{E20C}  
381 \usv\_set:nnn {stixfbbit}{T}{E20D}

382 \usv\_set:nnn {stixbfbbbit}{U}{E20E}  
383 \usv\_set:nnn {stixbfbbbit}{V}{E20F}  
384 \usv\_set:nnn {stixbfbbbit}{W}{E210}  
385 \usv\_set:nnn {stixbfbbbit}{X}{E211}  
386 \usv\_set:nnn {stixbfbbbit}{Y}{E212}  
  
387 \usv\_set:nnn {stixbfbbbit}{a}{E213}  
388 \usv\_set:nnn {stixbfbbbit}{b}{E214}  
389 \usv\_set:nnn {stixbfbbbit}{c}{E215}  
390 \usv\_set:nnn {stixbfbbbit}{e}{E217}  
391 \usv\_set:nnn {stixbfbbbit}{f}{E218}  
392 \usv\_set:nnn {stixbfbbbit}{g}{E219}  
393 \usv\_set:nnn {stixbfbbbit}{h}{E21A}  
394 \usv\_set:nnn {stixbfbbbit}{k}{E21D}  
395 \usv\_set:nnn {stixbfbbbit}{l}{E21E}  
396 \usv\_set:nnn {stixbfbbbit}{m}{E21F}  
397 \usv\_set:nnn {stixbfbbbit}{n}{E220}  
398 \usv\_set:nnn {stixbfbbbit}{o}{E221}  
399 \usv\_set:nnn {stixbfbbbit}{p}{E222}  
400 \usv\_set:nnn {stixbfbbbit}{q}{E223}  
401 \usv\_set:nnn {stixbfbbbit}{r}{E224}  
402 \usv\_set:nnn {stixbfbbbit}{s}{E225}  
403 \usv\_set:nnn {stixbfbbbit}{t}{E226}  
404 \usv\_set:nnn {stixbfbbbit}{u}{E227}  
405 \usv\_set:nnn {stixbfbbbit}{v}{E228}  
406 \usv\_set:nnn {stixbfbbbit}{w}{E229}  
407 \usv\_set:nnn {stixbfbbbit}{x}{E22A}  
408 \usv\_set:nnn {stixbfbbbit}{y}{E22B}  
409 \usv\_set:nnn {stixbfbbbit}{z}{E22C}  
  
410 \usv\_set:nnn {stixbfcal}{Latin}{E247}  
  
411 \usv\_set:nnn {stixbfitslash}{Latin}{E295}  
412 \usv\_set:nnn {stixbfitslash}{latin}{E2C9}  
413 \usv\_set:nnn {stixbfitslash}{greek}{E32D}  
414 \usv\_set:nnn {stixsfitslash}{epsilon}{E37B}  
415 \usv\_set:nnn {stixsfitslash}{vartheta}{E35F}  
416 \usv\_set:nnn {stixsfitslash}{varkappa}{E375}  
417 \usv\_set:nnn {stixsfitslash}{phi}{E361}  
418 \usv\_set:nnn {stixsfitslash}{varrho}{E377}  
419 \usv\_set:nnn {stixsfitslash}{varpi}{E363}  
420 \usv\_set:nnn {stixsfitslash}{digamma}{E36B}  
  
421 *(/stix)*

## File X

# um-code-setchar.dtx

## 10 *Setting up maths chars*

```
1 (*package)
```

### 10.1 *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.

```
2 \cs_new:Npn \@@_symbol_setup:
3 {
4   \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4
5   {
6     \exp_not:n { \_@@_sym:nnn {##1} {##2} {##3} }
7   }
8 }
9 \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.

```
10 \cs_new:Nn \@@_input_math_symbol_table: {\g_@@_mathtable_tl}
```

### 10.2 *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`.

```
11 \AtBeginDocument{\@@_define_math_chars:}
12 \cs_new:Nn \@@_define_math_chars:
13 {
14   \group_begin:
15   \cs_set:Npn \_@@_sym:nnn ##1##2##3
16   {
17     \tl_if_in:nnT
18     { \mathord \mathalpha \mathbin \mathrel \mathpunct \mathop \mathfence }
19     {##3}
20   }
21   \exp_last_unbraced:NNx \cs_gset_eq:NN ##2 { \Ucharcat ##1 ~ 12 ~ }
22 }
```

```

23     }
24     \@@_input_math_symbol_table:
25     \group_end:
26     }

```

### 10.3 Commands for each symbol/glyph/char

```

\@@_set_mathsymbol:nNn #1 : A LATEX 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.

```

27 \cs_set:Nn \@@_set_mathsymbol:nNn
28 {
29     \bool_lazy_and:nnT
30     {
31         \int_compare_p:nNn {#4} > {127}
32     }
33     {
34         \int_compare_p:nNn { \char_value_catcode:n {#4} } = {11}
35     }
36     { \char_set_catcode_other:n {#4} }
37
38     \tl_case:Nn #3
39     {
40         \mathord { \@@_set_mathcode:nnn {#4} {#3} {#1} }
41         \mathalpha { \@@_set_mathcode:nnn {#4} {#3} {#1} }
42         \mathbin { \@@_set_mathcode:nnn {#4} {#3} {#1} }
43         \mathrel { \@@_set_mathcode:nnn {#4} {#3} {#1} }
44         \mathpunct { \@@_set_mathcode:nnn {#4} {#3} {#1} }
45         \mathop { \@@_set_big_operator:nnn {#1} {#2} {#4} }
46         \mathopen { \@@_set_math_open:nnn {#1} {#2} {#4} }
47         \mathclose { \@@_set_math_close:nnn {#1} {#2} {#4} }
48         \mathfence { \@@_set_math_fence:nnnn {#1} {#2} {#3} {#4} }
49         \mathaccent
50         { \@@_set_math_accent:Nnnn #2 {fixed} {#1} {#4} }
51         \mathbotaccent
52         { \@@_set_math_accent:Nnnn #2 {bottom~ fixed} {#1} {#4} }
53         \mathaccentwide
54         { \@@_set_math_accent:Nnnn #2 {} {#1} {#4} }
55         \mathbotaccentwide
56         { \@@_set_math_accent:Nnnn #2 {bottom} {#1} {#4} }
57         \mathover

```

```

58     { \@@_set_math_overunder:Nnnn #2 {} {#1} {#4} }
59     \mathunder
60     { \@@_set_math_overunder:Nnnn #2 {bottom} {#1} {#4} }
61   }
62 }

63 \edef\mathfence{\string\mathfence}
64 \edef\mathover{\string\mathover}
65 \edef\mathunder{\string\mathunder}
66 \edef\mathbotaccent{\string\mathbotaccent}
67 \edef\mathaccentwide{\string\mathaccentwide}
68 \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  $\LaTeX$ /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 → )  $\Sigma$  → \sum_sym → \sumop\nolimits
( \int → )  $\int$  → \int_sym → \intop

```

```

69 \cs_new:Nn \@@_set_big_operator:nnn
70 {
71   \@@_char_gmake_mathactive:n {#3}
72   \cs_set_protected_nopar:Npx \@@_tmpa: { \exp_not:c { \cs_to_str:N #2 _sym } }
73   \char_gset_active_eq:nN {#3} \@@_tmpa:
74
75   \@@_set_mathchar:cNnn { \cs_to_str:N #2 op } \mathop {#1} {#3}
76
77   \cs_gset:cpx { \cs_to_str:N #2 _sym }
78   {
79     \exp_not:c { \cs_to_str:N #2 op }
80     \exp_not:n { \tl_if_in:NnT \l_@@_nolimits_tl {#2} \nolimits }
81   }
82 }

```

`\@@_set_math_open:nnn` #1 : Symbol font name  
 #2 : Macro to assign  
 #3 : Glyph slot

```

83 \cs_new:Nn \@@_set_math_open:nnn
84 {
85   \tl_if_in:NnTF \l_@@_radicals_tl {#2}
86   {
87     \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
88     { \@@_radical:nn {#1} {#3} }
89     \tl_set:cn {l_@@_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #3}
90   }
91   {
92     \@@_set_delcode:nnn {#1} {#3} {#3}
93     \@@_set_mathcode:nnn {#3} \mathopen {#1}
94     \cs_gset_protected_nopar:Npx #2
95     { \@@_delimiter:Nnn \mathopen {#1} {#3} }
96   }
97 }
  
```

`\@@_set_math_close:nnn` #1 : Symbol font name  
 #2 : Macro to assign  
 #3 : Glyph slot

```

98 \cs_new:Nn \@@_set_math_close:nnn
99 {
100  \@@_set_delcode:nnn {#1} {#3} {#3}
101  \@@_set_mathcode:nnn {#3} \mathclose {#1}
102  \cs_gset_protected_nopar:Npx #2
103  { \@@_delimiter:Nnn \mathclose {#1} {#3} }
104 }
  
```

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

```

105 \cs_new:Nn \@@_set_math_fence:nnnn
106 {
107  \@@_set_mathcode:nnn {#4} {#3} {#1}
108  \@@_set_delcode:nnn {#1} {#4} {#4}
109  \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
110  { \@@_delimiter:Nnn \mathopen {#1} {#4} }
111  \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
112  { \@@_delimiter:Nnn \mathclose {#1} {#4} }
113 }
  
```

`\@@_set_math_accent:Nnnn` #1 : Accend command  
 #2 : Accent type (string)  
 #3 : Symbol font name  
 #4 : Glyph slot

```

114 \cs_new:Nn \@@_set_math_accent:Nnnn
115 {
116   \cs_gset_protected_nopar:Npx #1
117   { \@@_accent:nnn {#2} {#3} {#4} }
118 }

```

`\@@_set_math_overunder:Nnnn` #1 : Accend command  
 #2 : Accent type (string)  
 #3 : Symbol font name  
 #4 : Glyph slot

```

119 \cs_new:Nn \@@_set_math_overunder:Nnnn
120 {
121   \cs_gset_protected_nopar:Npx #1 ##1
122   {
123     \mathop
124     { \@@_accent:nnn {#2} {#3} {#4} {##1} }
125     \limits
126   }
127 }

```

```

128 </package>

```

## File XI

# um-code-mathtext.dtx

## 11 Maths text commands

1 *(\*package)*

### 11.1 `\setmathfontface`

`\@@_setmathfontface:Nnn`

```
2 \keys_define:nn {@@_mathface}
3 {
4   version .code:n =
5   { \tl_set:Nn \l_@@_mversion_tl {#1} }
6 }
7 \cs_set:Nn \@@_setmathfontface:Nnn
8 {
9   \tl_clear:N \l_@@_mversion_tl
10
11   \keys_set_known:nnN {@@_mathface} {#2} \l_@@_keyval_clist
12
13   \exp_args:Nnx \fontspec_set_family:Nxn \l_@@_tmpa_tl
14   { ItalicFont={}, BoldFont={}, \exp_not:V \l_@@_keyval_clist } {#3}
15
16   \tl_if_empty:NT \l_@@_mversion_tl
17   {
18     \tl_set:Nn \l_@@_mversion_tl {normal}
19     \DeclareMathAlphabet #1 {\g_fontspeg_encoding_tl} {\l_@@_tmpa_tl} {\mdde-
20     fault} {\updefault}
21
22     \SetMathAlphabet #1 {\l_@@_mversion_tl} {\g_fontspeg_encoding_tl} {\l_@@_tmpa_tl} {\md-
23     default} {\updefault}
24
25     % integrate with fontspec's \setmathrm etc:
26     \tl_case:Nn #1
27     {
28       \mathrm { \cs_set_eq:NN \g__fontspec_mathrm_tl \l_@@_tmpa_tl }
29       \mathsf { \cs_set_eq:NN \g__fontspec_mathsf_tl \l_@@_tmpa_tl }
30       \mathtt { \cs_set_eq:NN \g__fontspec_mathtt_tl \l_@@_tmpa_tl }
31     }
32 }
```

### 11.2 *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.

### 11.2.1 Text font

```

32 \cs_generate_variant:Nn \tl_if_eq:nnT {o}
33 \cs_set:Nn \__fontspec_setmainfont_hook:nn
34   {
35     \tl_if_eq:onT {\g__fontspec_mathrm_tl} {\rmdefault}
36     {
37 (XE) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
38 (LU) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
39     \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
40     \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
41     \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
42     }
43   }
44
45 \cs_set:Nn \__fontspec_setsansfont_hook:nn
46   {
47     \tl_if_eq:onT {\g__fontspec_mathsf_tl} {\sfdefault}
48     {
49 (XE) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
50 (LU) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}
51     \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
52     \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
53     }
54   }
55
56 \cs_set:Nn \__fontspec_setmonofont_hook:nn
57   {
58     \tl_if_eq:onT {\g__fontspec_mathtt_tl} {\ttdefault}
59     {
60 (XE) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {#1} {#2}
61 (LU) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {Renderer=Basic,#1} {#2}
62     \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\mddefault\updefault
63     \SetMathAlphabet\mathtt{bold} \g_fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
64     }
65   }

```

### 11.2.2 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.

```

66 \cs_set:Nn \__fontspec_setmathrm_hook:nn
67   {
68     \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
69     \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
70     \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault

```

```

71 }
72 \cs_set:Nn \__fontspec_setboldmathrm_hook:nn
73 {
74   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mdefault\updefault
75   \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\bfdefault\updefault
76   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mdefault\itdefault
77 }
78 \cs_set:Nn \__fontspec_setmathsf_hook:nn
79 {
80   \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mdefault\updefault
81   \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
82 }
83 \cs_set:Nn \__fontspec_setmathtt_hook:nn
84 {
85   \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\mdefault\updefault
86   \SetMathAlphabet\mathtt{bold} \g_fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
87 }
88 </package>

```

## File XII

# um-code-main.dtx

## 12 *The main \setmathfont macro*

```
1 (*package)
```

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

```
\@@_setmathfont:nn
```

```
2 \cs_set:Nn \@@_setmathfont:nn
3 {
4   \tl_set:Nn \l_@@_fontname_tl {#2}
```

Erase any conception L<sup>A</sup>T<sub>E</sub>X has of previously defined math symbol fonts; this allows `\DeclareSymbolFont` at any point in the document.

```
5   \cs_set_eq:NN \glb@currsizel @scan_stop:
```

Initialise all local variables:

```
6   \@@_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.

```
7   \cs_if_exist:cf { S@ \f@size } { \calculate@math@sizes }
8   \csname S@\f@size\endcsname
```

Parse options and tell people what's going on:

```
9   \keys_set:known:nnN {unicode-math} {#1} \l_@@_unknown_keys_clist
10  \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.

```
11 (<debug> \csname TIC\endcsname
12   \@@_fontspec_select_font:
13 (<debug> \csname TOC\endcsname
14   \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
15   {
16     \@@_declare_math_sizes:
17     \@@_fontspec_select_font:
18   }
```

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

```
19   \cs_if_exist:cf { sym \l_@@_symfont_label_tl }
20   {
21     \DeclareSymbolFont{\l_@@_symfont_label_tl}
22       {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
23   }
```

```

24 \SetSymbolFont{\l_@@_symfont_label_tl}{\l_@@_mversion_tl}
25   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}

```

Set the bold math version.

```

26 \str_if_eq_x:nnT {\l_@@_mversion_tl} {normal}
27   {
28     \SetSymbolFont{\l_@@_symfont_label_tl}{bold}
29     {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
30   }

```

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:

```

31 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
32   {
33     \bool_set_true:N \g_@@_mainfont_already_set_bool
34     \@@_setup_legacy_fam_two:
35     \@@_setup_legacy_fam_three:
36   }

```

And now we input every single maths char.

```

37 <debug> \csname TIC\endcsname
38   \@@_input_math_symbol_table:
39 <debug> \csname TOC\endcsname

```

Finally,

- remap symbols that don't take their natural mathcode;
- activate any symbols that need to be math-active;
- assign delimiter codes for symbols that need to grow;
- setup the maths alphabets (`\symbf` etc.) — this is an extensive part of the code; see Section 15;
- setup negations, which are handled on an ad hoc basis; see Section 16.3.1.

```

40 \@@_remap_symbols:
41 \@@_setup_mathactives:
42 \@@_setup_delcodes:
43 <debug> \csname TIC\endcsname
44 \@@_setup_alphabets:
45 <debug> \csname TOC\endcsname
46 \@@_setup_negations:
47   }

```

*Fall-back font* Want to load Latin Modern Math if nothing else. This needs to happen early so that all of the font-loading machinery executes before the other 'At-BeginDocument' code.

```

48 \AtBeginDocument { \@@_load_lm_if_necessary: }
49 \cs_new:Nn \@@_load_lm_if_necessary:
50   {

```

```

51 \cs_if_exist:NF \l_@@_fontname_tl
52 {
53   % TODO: update this when lmmath-bold.otf is released
54   \setmathfont{latinmodern-math.otf}[BoldFont={latinmodern-math.otf}]
55   \bool_set_false:N \g_@@_mainfont_already_set_bool
56 }
57 }

```

Note that here we reset the ‘font already loaded’ boolean so that a new font being set will do the right thing in terms of setting up defaults.

TODO: need a better way to do this for the general case. (Maybe a ‘reset’ command option?)

\@@\_init:

```

58 \cs_new:Nn \@@_init:
59 {
60   \bool_set_true:N \l_@@_ot_math_bool
61   \tl_set:Nn \l_@@_mversion_tl {normal}
62   \tl_set:Nn \l_@@_symfont_label_tl {operators}

```

Defaults for the script and scriptscript font.

```

63   \tl_set:Nn \l_@@_script_features_tl {Style=MathScript}
64   \tl_set:Nn \l_@@_sscript_features_tl {Style=MathScriptScript}
65   \tl_set_eq:NN \l_@@_script_font_tl \l_@@_fontname_tl
66   \tl_set_eq:NN \l_@@_sscript_font_tl \l_@@_fontname_tl

```

Default to defining the font for every math symbol character.

```

67   \bool_set_true:N \l_@@_init_bool
68   \seq_clear:N \l_@@_char_range_seq
69   \clist_clear:N \l_@@_char_nrange_clist
70   \seq_clear:N \l_@@_mathalph_seq
71   \seq_clear:N \l_@@_missing_alph_seq

```

Other range initialisations.

```

72   \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_noparse:nnn
73   \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
74   \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_noparse:nnn
75   \cs_set_eq:NN \@@_maybe_init_alphabet:n \@@_init_alphabet:n
76   \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
77   \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_noparse:nn
78   \cs_set_eq:NN \@@_make_mathactive:nNN \@@_make_mathactive_noparse:nNN
79 }

```

\@@\_declare\_math\_sizes: Set the math sizes according to the recommended font parameters. TODO: this shouldn’t need to be per-engine; check out why the wrappers aren’t used.

```

80 \cs_new:Nn \@@_declare_math_sizes:
81 {
82   (*LU)
83   \fp_compare:nF { \@@_script_style_size:n {ScriptPercentScaleDown} == 0 }
84   {
85     \DeclareMathSizes { \f@size } { \f@size }

```

```

86     { \l_@@_script_style_size:n {ScriptPercentScaleDown} }
87     { \l_@@_script_style_size:n {ScriptScriptPercentScaleDown} }
88   }
89 </LU>
90 < *XE>
91   \dim_compare:nF { \fontdimen 10 \l_@@_font == 0pt }
92   {
93     \DeclareMathSizes { \f@size } { \f@size }
94     { \l_@@_fontdimen_to_scale:nn {10} { \l_@@_font } }
95     { \l_@@_fontdimen_to_scale:nn {11} { \l_@@_font } }
96   }
97 </XE>
98 }

```

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

```

99 < *LU>
100 \cs_new:Nn \l_@@_script_style_size:n
101 {
102   \fp_eval:n { \directlua{tex.sprint(luaotfload.aux.get_math_dimension("l_@@_font", "#1"))} * \f@size }
103 }
104 </LU>
105 % \end{macrocode}
106 % \end{macro}
107 %
108 % \begin{macro}{\l_@@_setup_legacy_fam_two:}
109 % \TeX\ won't load the same font twice at the same scale, so we need to mag-
110 % nify this one by an imperceptable amount.
111 % \begin{macrocode}
112 \cs_new:Nn \l_@@_setup_legacy_fam_two:
113 {
114   \fontspec_set_family:Nxn \l_@@_family_tl
115   {
116     \l_@@_font_keyval_tl,
117     Scale=1.00001,
118     FontAdjustment =
119     {
120       \l_@@_copy_fontparam:nnn { 8 } {43} {FractionNumeratorDis-
121       playStyleShiftUp}\relax
122       \l_@@_copy_fontparam:nnn { 9 } {42} {FractionNumeratorShiftUp}\relax
123       \l_@@_copy_fontparam:nnn {10} {32} {StackTopShiftUp}\relax
124       \l_@@_copy_fontparam:nnn {11} {45} {FractionDenominatorDis-
125       playStyleShiftDown}\relax
126       \l_@@_copy_fontparam:nnn {12} {44} {FractionDenominatorShiftDown}\relax
127       \l_@@_copy_fontparam:nnn {13} {21} {SuperscriptShiftUp}\relax
128       \l_@@_copy_fontparam:nnn {14} {21} {SuperscriptShiftUp}\relax
129       \l_@@_copy_fontparam:nnn {15} {22} {SuperscriptShiftUpCramped}\relax
130       \l_@@_copy_fontparam:nnn {16} {18} {SubscriptShiftDown}\relax
131       \l_@@_copy_fontparam:nnn {17} {18} {SubscriptShiftDownWithSuper-
132       script}\relax

```

```

129     \@@_copy_fontparam:nnn {18} {24} {SuperscriptBaselineDropMax}\relax
130     \@@_copy_fontparam:nnn {19} {20} {SubscriptBaselineDropMin}\relax
131     \@@_zero_fontparam:n {20} % delim1 = FractionDelimiterDisplaySize
132     \@@_zero_fontparam:n {21} % delim2 = FractionDelimiterSize
133     \@@_copy_fontparam:nnn {22} {15} {AxisHeight}\relax
134   }
135 } {\l_@@_fontname_tl}
136
137 \SetSymbolFont{symbols}{\l_@@_mversion_tl}
138   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
139
140 \str_if_eq_x:nnT {\l_@@_mversion_tl} {normal}
141   {
142     \SetSymbolFont{symbols}{bold}
143     {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
144   }
145 }

```

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

```

146 \cs_new:Nn \@@_setup_legacy_fam_three:
147   {
148     \fontspec_set_family:Nxn \l_@@_family_tl
149     {
150       \l_@@_font_keyval_tl,
151       Scale=0.99999,
152       FontAdjustment = {
153         \@@_copy_fontparam:nnn { 8} {48} {FractionRuleThickness}\relax
154         \@@_copy_fontparam:nnn { 9} {28} {UpperLimitGapMin}\relax
155         \@@_copy_fontparam:nnn {10} {30} {LowerLimitGapMin}\relax
156         \@@_copy_fontparam:nnn {11} {29} {UpperLimitBaselineRiseMin}\relax
157         \@@_copy_fontparam:nnn {12} {31} {LowerLimitBaselineDropMin}\relax
158         \@@_zero_fontparam:n {13}
159       }
160     } {\l_@@_fontname_tl}
161
162     \SetSymbolFont{largesymbols}{\l_@@_mversion_tl}
163     {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
164
165     \str_if_eq_x:nnT {\l_@@_mversion_tl} {normal}
166     {
167       \SetSymbolFont{largesymbols}{bold}
168       {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
169     }
170   }

```

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

```

171 \cs_new:Nn \@@_fontspec_select_font:
172   {
173     \tl_set:Nx \l_@@_font_keyval_tl {

```

```

174 (LU)   Renderer = Basic,
175   BoldItalicFont = {}, ItalicFont = {},
176   Script = Math,
177   SizeFeatures =
178   {
179     {
180       Size = \tf@size-
181     } ,
182     {
183       Size = \sf@size-\tf@size ,
184       Font = \l_@@_script_font_tl ,
185       \l_@@_script_features_tl
186     } ,
187     {
188       Size = -\sf@size ,
189       Font = \l_@@_sscript_font_tl ,
190       \l_@@_sscript_features_tl
191     }
192   } ,
193   \l_@@_unknown_keys_clist
194 }
195
196 \fontspec_set_fontface:NNxn \l_@@_font \l_@@_family_tl
197   {\l_@@_font_keyval_tl} {\l_@@_fontname_tl}

```

Check whether we're using a real maths font:

```

198   \group_begin:
199     \fontfamily{\l_@@_family_tl}\selectfont
200     \fontspec_if_script:nF {math} {\bool_gset_false:N \l_@@_ot_math_bool}
201   \group_end:
202 }

```

## 12.1 Functions for setting up symbols with mathcodes

\@@\_process\_symbol\_noparse:nnn \@@\_process\_symbol\_parse:nnn If the range font feature has been used, then only a subset of the Unicode glyphs are to be defined. See section §13.3 for the code that enables this.

```

203 \cs_set:Nn \@@_process_symbol_noparse:nnn
204   {
205     \@@_set_mathsymbol:nNNn {\l_@@_symfont_label_tl} #2 #3 {#1}
206   }
207 \cs_set:Nn \@@_process_symbol_parse:nnn
208   {
209     \@@_if_char_spec:nNNT {#1} {#2} {#3}
210     {
211       \@@_process_symbol_noparse:nnn {#1} {#2} {#3}
212     }
213   }

```

```

\@@_remap_symbols: This function is used to define the mathcodes for those chars which should be
\@@_remap_symbol_noparse:nnn mapped to a different glyph than themselves.
\@@_remap_symbol_parse:nnn
214 \cs_new:Npn \@@_remap_symbols:
215 {
216   \@@_remap_symbol:nnn{\-}{\mathbin}{"02212}% hyphen to minus
217   \@@_remap_symbol:nnn{\*}{\mathbin}{"02217}% text asterisk to "centred as-
      terisk"
218   \bool_if:NF \g_@@_literal_colon_bool
219     {
220       \@@_remap_symbol:nnn{\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
221     }
222 }

```

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

```

223 \cs_new:Nn \@@_remap_symbol_parse:nnn
224 {
225   \@@_if_char_spec:nNT {#3} {\@nil} {#2}
226   { \@@_remap_symbol_noparse:nnn {#1} {#2} {#3} }
227 }
228 \cs_new:Nn \@@_remap_symbol_noparse:nnn
229 {
230   \clist_map_inline:nn {#1}
231     { \@@_set_mathcode:nnnn {##1} {#2} {\l_@@_symfont_label_tl} {#3} }
232 }

```

## 12.2 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:`

```

233 \cs_new:Npn \@@_setup_mathactives:
234 {
235   \@@_make_mathactive:nNN {"2032} \@@_prime_single_mchar \mathord
236   \@@_make_mathactive:nNN {"2033} \@@_prime_double_mchar \mathord
237   \@@_make_mathactive:nNN {"2034} \@@_prime_triple_mchar \mathord
238   \@@_make_mathactive:nNN {"2057} \@@_prime_quad_mchar \mathord
239   \@@_make_mathactive:nNN {"2035} \@@_backprime_single_mchar \mathord
240   \@@_make_mathactive:nNN {"2036} \@@_backprime_double_mchar \mathord
241   \@@_make_mathactive:nNN {"2037} \@@_backprime_triple_mchar \mathord
242   \@@_make_mathactive:nNN {\`'} \mathstraightquote \mathord
243   \@@_make_mathactive:nNN {\`\`'} \mathbacktick \mathord
244 }

```

`\@@_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!

```

245 \cs_new:Nn \@@_make_mathactive_parse:nNN
246 {

```

```

247 \@@_if_char_spec:nNNT {#1} #2 #3
248   { \@@_make_mathactive_noparse:nNN {#1} #2 #3 }
249   }
250 \cs_new:Nn \@@_make_mathactive_noparse:nNN
251   {
252     \@@_set_mathchar:NNnn #2 #3 {\l_@@_symfont_label_tl} {#1}
253     \@@_char_gmake_mathactive:n {#1}
254   }

```

### 12.3 Delimiter codes

\@@\_assign\_delcode:nn

```

255 \cs_new:Nn \@@_assign_delcode_noparse:nn
256   {
257     \@@_set_delcode:nnn \l_@@_symfont_label_tl {#1} {#2}
258   }
259 \cs_new:Nn \@@_assign_delcode_parse:nn
260   {
261     \@@_if_char_spec:nNNT {#2} {\@nil} {\@nil}
262     {
263       \@@_assign_delcode_noparse:nn {#1} {#2}
264     }
265   }

```

\@@\_assign\_delcode:n Shorthand.

```

266 \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.

```

267 \cs_new:Npn \@@_setup_delcodes:
268   {
269     % ensure \left. and \right. work:
270     \@@_set_delcode:nnn \l_@@_symfont_label_tl {\`.} {\c_zero}
271     % this is forcefully done to fix a bug -- indicates a larger problem!
272
273     \@@_assign_delcode:nn {\`\/} {\g_@@_slash_delimiter_usv}
274     \@@_assign_delcode:nn {"2044} {\g_@@_slash_delimiter_usv} % fracslash
275     \@@_assign_delcode:nn {"2215} {\g_@@_slash_delimiter_usv} % divslash
276     \@@_assign_delcode:n {"005C} % backslash
277     \@@_assign_delcode:nn {\`<} {"27E8} % angle brackets with ascii notation
278     \@@_assign_delcode:nn {\`>} {"27E9} % angle brackets with ascii notation
279     \@@_assign_delcode:n {"2191} % up arrow
280     \@@_assign_delcode:n {"2193} % down arrow
281     \@@_assign_delcode:n {"2195} % updown arrow
282     \@@_assign_delcode:n {"219F} % up arrow twohead
283     \@@_assign_delcode:n {"21A1} % down arrow twohead

```

```

284 \@@_assign_delcode:n {"21A5} % up arrow from bar
285 \@@_assign_delcode:n {"21A7} % down arrow from bar
286 \@@_assign_delcode:n {"21A8} % updown arrow from bar
287 \@@_assign_delcode:n {"21BE} % up harpoon right
288 \@@_assign_delcode:n {"21BF} % up harpoon left
289 \@@_assign_delcode:n {"21C2} % down harpoon right
290 \@@_assign_delcode:n {"21C3} % down harpoon left
291 \@@_assign_delcode:n {"21C5} % arrows up down
292 \@@_assign_delcode:n {"21F5} % arrows down up
293 \@@_assign_delcode:n {"21C8} % arrows up up
294 \@@_assign_delcode:n {"21CA} % arrows down down
295 \@@_assign_delcode:n {"21D1} % double up arrow
296 \@@_assign_delcode:n {"21D3} % double down arrow
297 \@@_assign_delcode:n {"21D5} % double updown arrow
298 \@@_assign_delcode:n {"21DE} % up arrow double stroke
299 \@@_assign_delcode:n {"21DF} % down arrow double stroke
300 \@@_assign_delcode:n {"21E1} % up arrow dashed
301 \@@_assign_delcode:n {"21E3} % down arrow dashed
302 \@@_assign_delcode:n {"21E7} % up white arrow
303 \@@_assign_delcode:n {"21E9} % down white arrow
304 \@@_assign_delcode:n {"21EA} % up white arrow from bar
305 \@@_assign_delcode:n {"21F3} % updown white arrow
306 }

```

## 12.4 (Big) operators

The engine does what is necessary to deal with big operators for us automatically with `\Umathchardef`. However, the limits aren't set automatically; that is, we want to define, a la Plain TeX *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.

```

307 \tl_set:Nn \l_@@_nolimits_tl
308 {
309   \int\iint\iiint\iiiiint\oint\oiint\oiiint
310   \intclockwise\varointclockwise\ointctrlockwise\sumint
311   \intbar\intBar\fint\cirfnint\awint\rppolint
312   \scpolint\npolint\pointint\sqint\intlarkh\intx
313   \intcap\intcup\upint\lowint
314 }

```

## 12.5 Radicals

`\l_@@_radicals_tl` The radicals are organised in `\@@_set_mathsymbol:nNNn`. We organise radicals in the same way as `nolimits-operators`. (`\cuberoot` and `\fourthroot`, don't seem to behave as proper radicals.)

```
315 \tl_set:Nn \l_@@_radicals_tl {\sqrt \longdivision}
```

```
316 </package>
```

## File XIII

# um-code-fontopt.dtx

## 13 Font loading options

```
1 (*package)
```

### 13.1 Math version

```
2 \keys_define:nn {unicode-math}
3 {
4   version .code:n =
5   {
6     \tl_set:Nn \l_@@_mversion_tl {#1}
7     \DeclareMathVersion {\l_@@_mversion_tl}
8   }
9 }
```

### 13.2 Script and scriptscript font options

```
10 \keys_define:nn {unicode-math}
11 {
12   script-features .tl_set:N = \l_@@_script_features_tl ,
13   sscript-features .tl_set:N = \l_@@_sscript_features_tl ,
14   script-font .tl_set:N = \l_@@_script_font_tl ,
15   sscript-font .tl_set:N = \l_@@_sscript_font_tl ,
16 }
```

### 13.3 Range processing

```
17 \keys_define:nn {unicode-math}
18 {
19   range .code:n =
20   {
21     \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 §12.1 for the individual definitions

```
22   \int_incr:N \g_@@_fam_int
23   \tl_set:Nx \l_@@_symfont_label_tl {@@_fam\int_use:N\g_@@_fam_int}
24   \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_parse:nnn
25   \cs_set_eq:NN \@@_set_mathalphabet_char:Nnn \@@_mathmap_parse:Nnn
26   \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_parse:nnn
27   \cs_set_eq:NN \@@_maybe_init_alphabet:n \use_none:n
28   \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_parse:nn
29   \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_parse:nn
30   \cs_set_eq:NN \@@_make_mathactive:nNN \@@_make_mathactive_parse:nNN
```

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

```
31   \seq_clear:N \l_@@_char_range_seq
```

```

32 \seq_clear:N \l_@@_mclass_range_seq
33 \seq_clear:N \l_@@_cmd_range_seq
34 \seq_clear:N \l_@@_mathalph_seq
35
36 \clist_map_inline:nn {#1}
37 {
38   \@@_if_mathalph_decl:nTF {##1}
39   {
40     \seq_put_right:Nx \l_@@_mathalph_seq
41     {
42       { \exp_not:V \l_@@_tmpa_tl }
43       { \exp_not:V \l_@@_tmpb_tl }
44       { \exp_not:V \l_@@_tmpc_tl }
45     }
46   }
47 {

```

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.

```

48   \seq_if_in:NnTF \g_@@_mathclasses_seq {##1}
49   { \seq_put_right:Nn \l_@@_mclass_range_seq {##1} }
50   {
51     \bool_lazy_and:nnTF { \tl_if_single_p:n {##1} } { \token_if_cs_p:N ##1 }
52     { \seq_put_right:Nn \l_@@_cmd_range_seq {##1} }
53     { \seq_put_right:Nn \l_@@_char_range_seq {##1} }
54   }
55 }
56 }
57 }
58 }

```

\@@\_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  $\mathcal$ -> $\stical$ , which I marginally prefer in the general case.

```

59 \prg_new_conditional:Nnn \@@_if_mathalph_decl:n {TF}
60 {
61   \tl_set:Nn \l_@@_tmpa_tl {#1}
62   \tl_clear:N \l_@@_tmpb_tl
63   \tl_clear:N \l_@@_tmpc_tl
64

```

```

65 \tl_if_in:NnT \l_@@_tmpa_tl {->}
66 { \exp_after:wN \@@_split_arrow:w \l_@@_tmpa_tl \q_nil }
67
68 \tl_if_in:NnT \l_@@_tmpa_tl {/}
69 { \exp_after:wN \@@_split_slash:w \l_@@_tmpa_tl \q_nil }
70
71 \tl_set:Nx \l_@@_tmpa_tl { \tl_to_str:N \l_@@_tmpa_tl }
72 \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \math }
73 \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \sym }
74 \tl_trim_spaces:N \l_@@_tmpa_tl
75
76 \tl_if_empty:NT \l_@@_tmpc_tl
77 { \tl_set_eq:NN \l_@@_tmpc_tl \l_@@_tmpa_tl }
78
79 \seq_if_in:NVTF \g_@@_named_ranges_seq \l_@@_tmpa_tl
80 { \prg_return_true: } { \prg_return_false: }
81 }
82 \cs_set:Npn \@@_split_arrow:w #1->#2 \q_nil
83 {
84   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
85   \tl_set:Nx \l_@@_tmpc_tl { \tl_trim_spaces:n {#2} }
86 }
87 \cs_set:Npn \@@_split_slash:w #1/#2 \q_nil
88 {
89   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
90   \tl_set:Nx \l_@@_tmpb_tl { \tl_trim_spaces:n {#2} }
91 }

```

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

```

\@@_if_char_spec:nNNT #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 1.

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

```

92 \cs_new:Nn \@@_if_char_spec:nNNT
93 {
94   % math class:
95   \seq_if_in:NnT \l_@@_mclass_range_seq {#3}
96   { \use_none_delimit_by_q_nil:w }

```

Table 1: Ranges accepted by `\@@_if_char_spec:nNNT`.

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

```

97
98 % command name:
99 \seq_if_in:NnT \l_@@_cmd_range_seq {#2}
100   { \use_none_delimit_by_q_nil:w }
101
102 % character slot:
103 \seq_map_inline:Nn \l_@@_char_range_seq
104   {
105     \@@_int_if_slot_in_range:nnT {#1} {##1}
106     { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
107   }
108
109 % the following expands to nil if no match was found:
110 \use_none:nnn
111 \q_nil
112 \use:n
113   {
114     \clist_put_right:Nx \l_@@_char_nrange_clist { \int_eval:n {#1} }
115     #4
116   }
117 }

```

`\@@_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

```

```

118 \cs_new:Nn \@@_int_if_slot_in_range:nnT
119   { \@@_numrange_parse:nwT {#1} #2 - \q_nil - \q_stop {#3} }
120 \cs_set:Npn \@@_numrange_parse:nwT #1 #2 - #3 - #4 \q_stop #5
121   {
122     \tl_if_empty:nTF {#4} { \int_compare:nT {#1=#2} {#5} }
123     {
124       \tl_if_empty:nTF {#3} { \int_compare:nT {#1>=#2} {#5} }
125       {
126         \tl_if_empty:nTF {#2} { \int_compare:nT {#1<=#3} {#5} }
127         {

```

```
128     \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
129     } } }
130 }

131 </package>
```

## File XIV

# um-code-fontparam.dtx

## 14 Common interface for font parameters

1 *(\*package)*

X<sub>Y</sub>TeX and LuaTeX have different interfaces for math font parameters. We use LuaTeX's interface because it's much better, but rename the primitives to be more L<sup>A</sup>T<sub>E</sub>X3-like. There are getter and setter commands for each font parameter. The names of the parameters is derived from the LuaTeX names, with underscores inserted between words. For every parameter  $\Umath\langle\text{LuaTeX name}\rangle$ , we define an expandable getter command  $\@@_\langle\text{L<sup>A</sup>T<sub>E</sub>X3 name}\rangle:N$  and a protected setter command  $\@@_set_\langle\text{L<sup>A</sup>T<sub>E</sub>X3 name}\rangle:Nn$ . 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 X<sub>Y</sub>TeX doesn't seem to provide  $\fontdimens$ ; in this case the getter and setter commands are left undefined.

*Cramped style tokens* LuaTeX has  $\crampeddisplaystyle$  etc., but they are loaded as  $\luatexcrampeddisplaystyle$  etc. by the `luatextra` package. X<sub>Y</sub>TeX, 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  $\langle\text{command}\rangle$  as a new cramped style switch. For LuaTeX, simply rename the corresponding primitive if it is not already defined. For X<sub>Y</sub>TeX, define  $\langle\text{command}\rangle$  as a new quark.

```
2 \cs_new_protected_nopar:Nn \@@_new_cramped_style:N
3 (XE) { \quark_new:N #1 }
4 (LU) {
5 (LU)   \cs_if_exist:NF #1
6 (LU)     { \cs_new_eq:Nc #1 { luatex \cs_to_str:N #1 } }
7 (LU) }
```

```

\crampeddisplaystyle The cramped style commands.
  \crampedtextstyle   8 \@@_new_cramped_style:N \crampeddisplaystyle
  \crampedscriptstyle 9 \@@_new_cramped_style:N \crampedtextstyle
\crampedscriptscriptstyle 10 \@@_new_cramped_style:N \crampedscriptstyle
  \@@_new_cramped_style:N \crampedscriptscriptstyle 11

```

*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  $\XintE\TeX$ , 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  $\XintE\TeX$  math font dimension. <style token> 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.

```

```

12 (*XE)
13 \cs_new_nopar:Npn \@@_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
14   \fontdimen
15   \cs_if_eq:NNTF #1 \displaystyle {
16     #2 \textfont
17   } {
18     \cs_if_eq:NNTF #1 \crampeddisplaystyle {
19       #3 \textfont
20     } {
21       \cs_if_eq:NNTF #1 \textstyle {
22         #4 \textfont
23       } {
24         \cs_if_eq:NNTF #1 \crampedtextstyle {
25           #5 \textfont
26         } {
27           \cs_if_eq:NNTF #1 \scriptstyle {
28             #4 \scriptfont
29           } {
30             \cs_if_eq:NNTF #1 \crampedscriptstyle {
31               #5 \scriptfont
32             } {
33               \cs_if_eq:NNTF #1 \scriptscriptstyle {
34                 #4 \scriptscriptfont
35               } {
36                 #5 \scriptscriptfont
37               }

```

Should we check here if the style is invalid?

```

38         }
39     }
40 }
41 }
42 }
43 }

```

Which family to use?

```

44     \c_two
45 }
46 </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 $\TeX$ .

```

\@@_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 $\TeX$  font parameter name is produced by removing all underscores and prefixing the result with  $Umath$ . The X $\TeX$  font dimension numbers must be integer constants.

```

47 \cs_new_protected_nopar:Nn \@@_font_param:nnnnn
48 <*XE>
49 {
50   \@@_font_param_aux:ccnnnn { @@_ #1 :N } { @@_set_ #1 :Nn }
51     { #2 } { #3 } { #4 } { #5 }
52 }
53 </XE>
54 <*LU>
55 {
56   \tl_set:Nn \l_@@_tmpa_tl { #1 }
57   \tl_remove_all:Nn \l_@@_tmpa_tl { _ }
58   \@@_font_param_aux:ccc { @@_ #1 :N } { @@_set_ #1 :Nn }
59     { Umath \l_@@_tmpa_tl }
60 }
61 </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 Lua $\TeX$  font parameter name is produced by removing all underscores and prefixing the result with  $Umath$ . The X $\TeX$  font dimension numbers must be integer constants.

```

62 \cs_new_protected_nopar:Nn \@@_font_param:nnn

```

```

63 {
64 \@@_font_param:nnnn { #1 } { #2 } { #2 } { #3 } { #3 }
65 }

```

`\@@_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 prefixing the result with `Umath`. The XeTeX font dimension number must be an integer constant.

```

66 \cs_new_protected_nopar:Nn \@@_font_param:nn
67 {
68 \@@_font_param:nnnn { #1 } { #2 } { #2 } { #2 } { #2 }
69 }

```

`\@@_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`.

```

70 \cs_new_protected_nopar:Nn \@@_font_param:n
71 <XE> { }
72 <LU> { \@@_font_param:nnnn { #1 } { 0 } { 0 } { 0 } { 0 } }

```

`\@@_font_param_aux:NNnnnn` Auxiliary macros for generating font parameter accessor macros.

`\@@_font_param_aux:NNN`

```

73 <*XE>
74 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNnnnn
75 {
76 \cs_new_nopar:Npn #1 ##1
77 {
78 \@@_font_dimen:Nnnnn ##1 { #3 } { #4 } { #5 } { #6 }
79 }
80 \cs_new_protected_nopar:Npn #2 ##1 ##2
81 {
82 #1 ##1 \dim_eval:n { ##2 }
83 }
84 }
85 \cs_generate_variant:Nn \@@_font_param_aux:NNnnnn { cc }
86 </XE>
87 <*LU>
88 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNN
89 {
90 \cs_new_nopar:Npn #1 ##1
91 {
92 #3 ##1
93 }
94 \cs_new_protected_nopar:Npn #2 ##1 ##2
95 {
96 #3 ##1 \dim_eval:n { ##2 }

```

```

97     }
98   }
99   \cs_generate_variant:Nn \@_font_param_aux:NNN { ccc }
100 </LU>

```

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

```

101 \@_font_param:nn { axis } { 15 }
102 \@_font_param:nn { operator_size } { 13 }
103 \@_font_param:n { fraction_del_size }
104 \@_font_param:nnn { fraction_denom_down } { 45 } { 44 }
105 \@_font_param:nnn { fraction_denom_vgap } { 50 } { 49 }
106 \@_font_param:nnn { fraction_num_up } { 43 } { 42 }
107 \@_font_param:nnn { fraction_num_vgap } { 47 } { 46 }
108 \@_font_param:nn { fraction_rule } { 48 }
109 \@_font_param:nn { limit_above_bgap } { 29 }
110 \@_font_param:n { limit_above_kern }
111 \@_font_param:nn { limit_above_vgap } { 28 }
112 \@_font_param:nn { limit_below_bgap } { 31 }
113 \@_font_param:n { limit_below_kern }
114 \@_font_param:nn { limit_below_vgap } { 30 }
115 \@_font_param:nn { over_delimiter_vgap } { 41 }
116 \@_font_param:nn { over_delimiter_bgap } { 38 }
117 \@_font_param:nn { under_delimiter_vgap } { 40 }
118 \@_font_param:nn { under_delimiter_bgap } { 39 }
119 \@_font_param:nn { overbar_kern } { 55 }
120 \@_font_param:nn { overbar_rule } { 54 }
121 \@_font_param:nn { overbar_vgap } { 53 }
122 \@_font_param:n { quad }
123 \@_font_param:nn { radical_kern } { 62 }
124 \@_font_param:nn { radical_rule } { 61 }
125 \@_font_param:nnn { radical_vgap } { 60 } { 59 }
126 \@_font_param:nn { radical_degree_before } { 63 }
127 \@_font_param:nn { radical_degree_after } { 64 }
128 \@_font_param:nn { radical_degree_raise } { 65 }
129 \@_font_param:nn { space_after_script } { 27 }
130 \@_font_param:nnn { stack_denom_down } { 35 } { 34 }
131 \@_font_param:nnn { stack_num_up } { 33 } { 32 }
132 \@_font_param:nnn { stack_vgap } { 37 } { 36 }
133 \@_font_param:nn { sub_shift_down } { 18 }
134 \@_font_param:nn { sub_shift_drop } { 20 }
135 \@_font_param:n { subsup_shift_down }
136 \@_font_param:nn { sub_top_max } { 19 }
137 \@_font_param:nn { subsup_vgap } { 25 }
138 \@_font_param:nn { sup_bottom_min } { 23 }
139 \@_font_param:nn { sup_shift_drop } { 24 }
140 \@_font_param:nnnn { sup_shift_up } { 21 } { 22 } { 21 } { 22 }
141 \@_font_param:nn { supsub_bottom_max } { 26 }
142 \@_font_param:nn { underbar_kern } { 58 }
143 \@_font_param:nn { underbar_rule } { 57 }

```

```

144 \@@_font_param:nn { underbar_vgap } { 56 }
145 \@@_font_param:n { connector_overlap_min }

```

## 14.1 Historical commands

TODO: maybe no longer necessary?

```

\@@_fontdimen_to_percent:nn #1 : Font dimen number
\@@_fontdimen_to_scale:nn #2 : Font 'variable'

```

\fontdimens 10, 11, and 65 aren't actually dimensions, they're percentage values given in units of sp. \@@\_fontdimen\_to\_percent:nn takes a font dimension number and outputs the decimal value of the associated parameter. \@@\_fontdimen\_to\_scale:nn returns a dimension correspond to the current font size relative proportion based on that percentage.

```

146 \cs_new:Nn \@@_fontdimen_to_percent:nn
147 {
148   \fp_eval:n { \dim_to_decimal:n { \fontdimen #1 #2 } * 65536 / 100 }
149 }
150 \cs_new:Nn \@@_fontdimen_to_scale:nn
151 {
152   \fp_eval:n { \@@_fontdimen_to_percent:nn {#1} {#2} * \f@size } pt
153 }

```

```

\@@_mathstyle_scale:Nnn #1 : A math style (\scriptstyle, say)
#2 : Macro that takes a non-delimited length argument (like \kern)
#3 : Length control sequence to be scaled according to the math style

```

This macro is used to scale the lengths reported by \fontdimen according to the scale factor for script- and scriptscript-size objects.

```

154 \cs_new:Nn \@@_mathstyle_scale:Nnn
155 {
156   \ifx#1\scriptstyle
157     #2 \@@_fontdimen_to_percent:nn {10} \l_@@_font #3
158   \else
159     \ifx#1\scriptscriptstyle
160       #2 \@@_fontdimen_to_percent:nn {11} \l_@@_font #3
161     \else
162       #2 #3
163     \fi
164   \fi
165 }
166 </package>

```

## File XV

# um-code-mathmap.dtx

## 15 Mapping in maths alphabets

```
1 (*package)
```

Switching to a different style of alphabetic symbols was traditionally performed with commands like `\mathbf`, which literally changes fonts to access alternate symbols. This is not as simple with Unicode fonts.

In traditional T<sub>E</sub>X 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 `\symbf` and so on. However, this means that ‘classical’ T<sub>E</sub>X 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.

### 15.1 Hooks into L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

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 `\DeclareSymbolFontAlpha`.) 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.

```
2 \cs_set:Npn \use@mathgroup #1 #2
3 {
4   \mode_if_math:T % <- not sure if this is really necessary since we've just checked for mmode and raised
   ror if not!
5   {
```

```

6   \math@bgroup
7   \cs_if_eq:cNF {M@\f@encoding} #1 {#1}
8   \@@_switchto_literal:
9   \mathgroup #2 \relax
10  \math@egroup
11  }
12  }

```

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}`. I thought that hooking into `\operator@font` would be hard because all other maths font selection in 2e uses `\mathrm{...}` style. Then reading source2e a little more I stumbled upon:

`\operator@font`

```

13 \cs_set:Npn \operator@font
14 {
15   \@@_switchto_literal:
16   \@fontswitch {} { \g_@@_operator_mathfont_t1 }
17 }

```

## 15.2 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.

### 15.3 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.

```
18 \cs_new:Nn \@@_prepare_mathstyle:n
19 {
20   \seq_put_right:Nn \g_@@_mathstyles_seq {#1}
21   \@@_init_alphabet:n {#1}
22   \cs_set:cpn {_@@_sym_#1_aux:n}
23   { \use:c {@@_switchto_#1:} \math@egroup }
24   \cs_set_protected:cpx {sym#1}
25   {
26     \exp_not:n
27     {
28       \math@bgroup
29       \mode_if_math:F
30       {
31         \egroup\expandafter
32         \non@alpherr\expandafter{\csname sym#1\endcsname\space}
33       }
34       \tl_set:Nn \l_@@_mathstyle_tl {#1}
35     }
36     \exp_not:c {_@@_sym_#1_aux:n}
37   }
38 }
```

`\@@_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.

```
39 \cs_set:Nn \@@_init_alphabet:n
40 {
41   \@@_log:nx {alph-initialise} {#1}
42   \cs_set_eq:cN {@@_switchto_#1:} \prg_do_nothing:
43 }
```

### 15.4 Definition of alphabets and styles

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

alphabetical symbols).

```

44 \cs_new:Nn \@@_new_named_range:n
45 {
46   \prop_new:c {g_@@_named_range_#1_prop}
47 }
48 \clist_set:Nn \g_@@_named_ranges_clist
49 {
50   up, it, tt, bfup, bfit, bb , bbit, scr, bfscr, cal, bfcalf,
51   frak, bffrak, sfup, sfit, bfsfup, bfsfit, bfsf
52 }
53 \clist_map_inline:Nn \g_@@_named_ranges_clist
54 { \@@_new_named_range:n {#1} }

```

Each alphabet style needs to be configured. This happens in Section 20.

```

55 \cs_new:Nn \@@_new_alphabet_config:nnn
56 {
57   \prop_if_exist:cF {g_@@_named_range_#1_prop}
58   { \@@_warning:nnn {no-named-range} {#1} {#2} }
59
60   \prop_gput:cnn {g_@@_named_range_#1_prop} { alpha_tl }
61   {
62     \prop_item:cn {g_@@_named_range_#1_prop} { alpha_tl }
63     {#2}
64   }
65   % Q: do I need to bother removing duplicates?
66
67   \cs_new:cn { @@_config_#1_#2:n } {#3}
68 }
69 \cs_new:Nn \@@_alphabet_config:nnn
70 {
71   \use:c {@@_config_#1_#2:n} {#3}
72 }
73 \prg_new_conditional:Nnn \@@_if_alphabet_exists:nn {T,TF}
74 {
75   \cs_if_exist:cTF {@@_config_#1_#2:n}
76   \prg_return_true: \prg_return_false:
77 }

```

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.

```

78 \cs_new:Nn \@@_default_mathalph:nnn
79 {
80   \seq_put_right:Nx \g_@@_named_ranges_seq { \tl_to_str:n {#1} }
81   \seq_put_right:Nn \g_@@_default_mathalph_seq {{#1}{#2}{#3}}
82   \prop_gput:cnn { g_@@_named_range_#1_prop } { default-alpha } {#2}
83 }
84 \@@_default_mathalph:nnn {up    } {latin, Latin, greek, Greek, num, misc} {up    }
85 \@@_default_mathalph:nnn {it    } {latin, Latin, greek, Greek, misc}   {it    }

```

```

86 \@@_default_mathalph:nnn {bb    } {latin, Latin, num, misc}      {bb    }
87 \@@_default_mathalph:nnn {bbit  } {misc}                       {bbit  }
88 \@@_default_mathalph:nnn {scr   } {latin, Latin}                {scr   }
89 \@@_default_mathalph:nnn {cal   } {Latin}                       {scr   }
90 \@@_default_mathalph:nnn {bfcal } {Latin}                    {bfscr }
91 \@@_default_mathalph:nnn {frac  } {latin, Latin}                {frac  }
92 \@@_default_mathalph:nnn {tt    } {latin, Latin, num}           {tt    }
93 \@@_default_mathalph:nnn {sfup  } {latin, Latin, num}           {sfup  }
94 \@@_default_mathalph:nnn {sfit  } {latin, Latin}                {sfit  }
95 \@@_default_mathalph:nnn {bfup  } {latin, Latin, greek, Greek, num, misc} {bfup  }
96 \@@_default_mathalph:nnn {bfit  } {latin, Latin, greek, Greek, misc} {bfit  }
97 \@@_default_mathalph:nnn {bfscr } {latin, Latin}                {bfscr }
98 \@@_default_mathalph:nnn {bffrak} {latin, Latin}                {bffrak}
99 \@@_default_mathalph:nnn {bfsfup} {latin, Latin, greek, Greek, num, misc} {bfsfup}
100 \@@_default_mathalph:nnn {bfsfit} {latin, Latin, greek, Greek, misc} {bfsfit}

```

#### 15.4.1 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!

```

101 \clist_map_inline:nn
102 {
103   up, it, bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf,
104   tt, bb, bbit, scr, bfscr, cal, bfcal, frak, bffrak,
105   normal, literal, sf, bf,
106 }
107 { \@@_prepare_mathstyle:n {#1} }

```

#### 15.4.2 New names for legacy textmath alphabet selection

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

```

108 \clist_map_inline:nn
109 { rm, it, bf, sf, tt }
110 { \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.

#### 15.4.3 Replacing legacy pure-maths alphabets

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

```

111 \clist_map_inline:nn
112 {
113   normal, bb , bbit, scr, bfscr, cal, bfcal, frak, bffrak, tt,
114   bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf
115 }
116 {
117   \cs_set:cpx { math #1 } { \exp_not:c { sym #1 } }
118 }

```

#### 15.4.4 New commands for ambiguous alphabets

```
119 \AtBeginDocument{
120 \clist_map_inline:nn
121 { rm, it, bf, sf, tt }
122 {
123 \cs_set_protected:cpx { math #1 }
124 {
125 \exp_not:n { \bool_if:NTF } \exp_not:c { g_@@_ math #1 _text_bool}
126 { \exp_not:c { mathtext #1 } }
127 { \exp_not:c { sym #1 } }
128 }
129 }}
```

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

```
130 \cs_set_protected:Npn \mathup { \mathrm }
131 \cs_set_protected:Npn \symrm { \symup }
```

#### 15.5 Defining the math alphabets per style

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

```
132 \cs_new:Npn \@@_setup_alphabets:
133 {
```

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

```
134 \seq_if_empty:NTF \l_@@_mathalph_seq
135 {
136 \@@_log:n {setup-implicit}
137 \seq_set_eq:NN \l_@@_mathalph_seq \g_@@_default_mathalph_seq
138 \bool_set_true:N \l_@@_implicit_alph_bool
139 \@@_maybe_init_alphabet:n {sf}
140 \@@_maybe_init_alphabet:n {bf}
141 \@@_maybe_init_alphabet:n {bfsf}
142 }
```

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

```
143 {
144 \@@_log:n {setup-explicit}
145 \bool_set_false:N \l_@@_implicit_alph_bool
146 \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
147 \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
148 }
149
150 % Now perform the mapping:
151 \seq_map_inline:Nn \l_@@_mathalph_seq
152 {
153 \tl_set:N0 \l_@@_style_tl { \use_i:nnn ##1 }
154 \clist_set:N0 \l_@@_alphabet_clist { \use_ii:nnn ##1 }
```

```

155 \tl_set:Nn \l_@@_remap_style_tl { \use_iii:nnn ##1 }
156
157 % If no set of alphabets is defined:
158 \clist_if_empty:NT \l_@@_alphabet_clist
159 {
160   \cs_set_eq:NN \@@_maybe_init_alphabet:n \@@_init_alphabet:n
161   \prop_get:cnN { g_@@_named_range_ \l_@@_style_tl _prop }
162   { default-alpha } \l_@@_alphabet_clist
163 }
164
165 \@@_setup_math_alphabet:
166 }
167 \seq_if_empty:NF \l_@@_missing_alph_seq { \@@_log:n { missing-alphabets } }
168 }

```

\@@\_setup\_math\_alphabet:

```

169 \cs_new:Nn \@@_setup_math_alphabet:
170 {

```

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

```

171 \clist_map_inline:Nn \l_@@_alphabet_clist
172 {
173   \tl_set:Nn \l_@@_alphabet_tl {##1}
174   \@@_if_alphabet_exists:nnTF \l_@@_style_tl \l_@@_alphabet_tl
175   {
176     \str_if_eq_x:nnTF {\l_@@_alphabet_tl} {misc}
177     {
178       \@@_maybe_init_alphabet:n \l_@@_style_tl
179       \clist_map_break:
180     }
181     {
182       \@@_glyph_if_exist:NnT \l_@@_font { \@@_to_usv:nn {\l_@@_style_tl} {\l_@@_alphabet_tl} }
183       {
184         \@@_maybe_init_alphabet:n \l_@@_style_tl
185         \clist_map_break:
186       }
187     }
188   }
189   { \msg_warning:nxx {unicode-math} {no-alphabet} { \l_@@_style_tl / \l_@@_alphabet_tl } }
190 }

```

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

```

191 (debug) \csname TIC\endcsname
192 \clist_map_inline:Nn \l_@@_alphabet_clist
193 {
194   \tl_set:Nx \l_@@_alphabet_tl { \tl_trim_spaces:n {##1} }
195   \cs_if_exist:cT {@@_config_ \l_@@_style_tl _ \l_@@_alphabet_tl :n}
196   {

```

```

197 \exp_args:No \tl_if_eq:nnTF \l_@@_alphabet_tl {misc}
198 {
199 \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
200 \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
201 }
202 {
203 \@@_glyph_if_exist:NnTF \l_@@_font { \@@_to_usv:nn {\l_@@_remap_style_tl} {\l_@@_alphabet_tl} }
204 {
205 \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
206 \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
207 }
208 {
209 \bool_if:NTF \l_@@_implicit_alph_bool
210 {
211 \seq_put_right:Nx \l_@@_missing_alph_seq
212 {
213 \backslashchar sym \l_@@_style_tl \space
214 (\tl_use:c{c_@@_math_alphabet_name_ \l_@@_alphabet_tl _tl})
215 }
216 }
217 {
218 \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {up}
219 }
220 }
221 }
222 }
223 }
224 (debug) \csname TOC\endcsname
225 }

```

## 15.6 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.

### 15.6.1 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`

```

226 \cs_new:Nn \@@_map_char_noparse:nn
227 { \@@_set_mathcode:nnnn {#1}{\mathalpha}{\l_@@_symfont_label_tl}{#2} }
228 \cs_new:Nn \@@_map_char_parse:nn
229 {
230 \@@_if_char_spec:nNNT {#1} {\@nil} {\mathalpha}
231 { \@@_map_char_noparse:nn {#1}{#2} }
232 }

```

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

```

233 \cs_new:Nn \@@_map_char_single:nnn
234 {
235   \@@_map_char_single:nn { \@@_to_usv:nn {#1}{#3} }
236                       { \@@_to_usv:nn {#2}{#3} }
237 }
```

`\@@_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:

```

238 \cs_set:Nn \@@_map_chars_range:nnn
239 {
240   \int_step_inline:nnnn {0}{1}{#1-1}
241   { \@@_map_char_single:nn {#2+##1}{#3+##1} }
242 }
```

And the wrapper with names:

```

243 \cs_new:Nn \@@_map_chars_range:nnnn
244 {
245   \@@_map_chars_range:nnn {#1} { \@@_to_usv:nn {#2}{#4} }
246                       { \@@_to_usv:nn {#3}{#4} }
247 }
```

### 15.6.2 Functions for 'normal' alphabet symbols

`\@@_set_normal_char:nnn`

```

248 \cs_set:Nn \@@_set_normal_char:nnn
249 {
250   \@@_usv_if_exist:nnT {#3} {#1}
251   {
252     \clist_map_inline:nn {#2}
253     {
254       \@@_set_mathalphabet_pos:nnnn {normal} {#1} {##1} {#3}
255       \@@_map_char_single:nnn {##1} {#3} {#1}
256     }
257   }
258 }
```

`\cs_new:Nn \@@_set_normal_Latin:nn`

```

259 \cs_new:Nn \@@_set_normal_Latin:nn
260 {
261   \clist_map_inline:nn {#1}
262   {
263     \@@_set_mathalphabet_Latin:nnn {normal} {##1} {#2}
264     \@@_map_chars_range:nnnn {26} {##1} {#2} {Latin}

```

```

265 }
266 }
267 \cs_new:Nn \@@_set_normal_latin:nn
268 {
269   \clist_map_inline:nn {#1}
270   {
271     \@@_set_mathalphabet_latin:nnn {normal} {##1} {#2}
272     \@@_map_chars_range:nnnn {26} {##1} {#2} {latin}
273   }
274 }
275 \cs_new:Nn \@@_set_normal_greek:nn
276 {
277   \clist_map_inline:nn {#1}
278   {
279     \@@_set_mathalphabet_greek:nnn {normal} {##1} {#2}
280     \@@_map_chars_range:nnnn {25} {##1} {#2} {greek}
281     \@@_map_char_single:nnn {##1} {#2} {epsilon}
282     \@@_map_char_single:nnn {##1} {#2} {vartheta}
283     \@@_map_char_single:nnn {##1} {#2} {varkappa}
284     \@@_map_char_single:nnn {##1} {#2} {phi}
285     \@@_map_char_single:nnn {##1} {#2} {varrho}
286     \@@_map_char_single:nnn {##1} {#2} {varpi}
287     \@@_set_mathalphabet_pos:nnnn {normal} {epsilon} {##1} {#2}
288     \@@_set_mathalphabet_pos:nnnn {normal} {vartheta} {##1} {#2}
289     \@@_set_mathalphabet_pos:nnnn {normal} {varkappa} {##1} {#2}
290     \@@_set_mathalphabet_pos:nnnn {normal} {phi} {##1} {#2}
291     \@@_set_mathalphabet_pos:nnnn {normal} {varrho} {##1} {#2}
292     \@@_set_mathalphabet_pos:nnnn {normal} {varpi} {##1} {#2}
293   }
294 }
295 \cs_new:Nn \@@_set_normal_Greek:nn
296 {
297   \clist_map_inline:nn {#1}
298   {
299     \@@_set_mathalphabet_Greek:nnn {normal} {##1} {#2}
300     \@@_map_chars_range:nnnn {25} {##1} {#2} {Greek}
301     \@@_map_char_single:nnn {##1} {#2} {varTheta}
302     \@@_set_mathalphabet_pos:nnnn {normal} {varTheta} {##1} {#2}
303   }
304 }
305 \cs_new:Nn \@@_set_normal_numbers:nn
306 {
307   \@@_set_mathalphabet_numbers:nnn {normal} {#1} {#2}
308   \@@_map_chars_range:nnnn {10} {#1} {#2} {num}
309 }

```

## 15.7 Mapping chars inside a math style

### 15.7.1 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.

```
310 \cs_new:Nn \@@_mathmap_noparse:nnn
311 {
312   \clist_map_inline:nn {#2}
313   {
314     \tl_put_right:cx {@@_switchto_#1:}
315     {
316       \@@_set_mathcode:nnnn {##1} {\mathalpha} {\l_@@_symfont_label_tl} {#3}
317     }
318   }
319 }
```

`\@@_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:nnn` declarations to the maths alphabet definition.

```
320 \cs_new:Nn \@@_mathmap_parse:nnn
321 {
322   \clist_if_in:NnT \l_@@_char_nrange_clist {#3}
323   {
324     \@@_mathmap_noparse:nnn {#1}{#2}{#3}
325   }
326 }
```

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

```
327 \cs_new:Nn \@@_set_mathalphabet_char:nnnn
328 {
329   \@@_set_mathalphabet_char:nnn {#1} { \@@_to_usv:nn {#2} {#4} }
330   { \@@_to_usv:nn {#3} {#4} }
331 }
```

`\@@_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:

```
332 \cs_new:Nn \@@_set_mathalph_range:nnnn
333 {
334   \int_step_inline:nnnn {0} {1} {#1-1}
335     { \@@_set_mathalphabet_char:nnn {#2} { ##1 + #3 } { ##1 + #4 } }
336 }
```

Then the wrapper version that uses names:

```
337 \cs_new:Nn \@@_set_mathalph_range:nnnn
338 {
339   \@@_set_mathalph_range:nnnn {#1} {#2} { \@@_to_usv:nn {#3} {#5} }
340                                     { \@@_to_usv:nn {#4} {#5} }
341 }
```

### 15.7.2 *Individual mapping functions for different alphabets*

```
342 \cs_new:Nn \@@_set_mathalphabet_pos:nnnn
343 {
344   \@@_usv_if_exist:nnT {#4} {#2}
345   {
346     \clist_map_inline:nn {#3}
347       { \@@_set_mathalphabet_char:nnnn {#1} {##1} {#4} {#2} }
348   }
349 }

350 \cs_new:Nn \@@_set_mathalphabet_numbers:nnn
351 {
352   \clist_map_inline:nn {#2}
353     { \@@_set_mathalph_range:nnnn {10} {#1} {##1} {#3} {num} }
354 }

355 \cs_new:Nn \@@_set_mathalphabet_Latin:nnn
356 {
357   \clist_map_inline:nn {#2}
358     { \@@_set_mathalph_range:nnnn {26} {#1} {##1} {#3} {Latin} }
359 }

360 \cs_new:Nn \@@_set_mathalphabet_latin:nnn
361 {
362   \clist_map_inline:nn {#2}
363     {
364       \@@_set_mathalph_range:nnnn {26} {#1} {##1} {#3} {latin}
365       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {h}
366     }
367 }

368 \cs_new:Nn \@@_set_mathalphabet_Greek:nnn
```

```

369 {
370   \clist_map_inline:nn {#2}
371   {
372     \@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {Greek}
373     \@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varTheta}
374   }
375 }

376 \cs_new:Nn \@_set_mathalphabet_greek:nnn
377 {
378   \clist_map_inline:nn {#2}
379   {
380     \@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {greek}
381     \@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {epsilon}
382     \@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {vartheta}
383     \@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varkappa}
384     \@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {phi}
385     \@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varrho}
386     \@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varpi}
387   }
388 }

389 </package>

```

## File XVI

# um-code-epilogue.dtx

## 16 Epilogue

1 *(\*package)*

Lots of little things to tidy up.

### 16.1 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.

```
2 \AtBeginDocument{\@@_resolve_greek:}
3 \cs_new:Npn \@@_resolve_greek:
4 {
5   \clist_map_inline:nn
6   {
7     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
8     alpha,beta,gamma,delta,epsilon,zeta,eta,theta,iota,kappa,lambda,
9     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
10    mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,phi,chi,psi,omega,
11    varTheta,varsigma,vartheta,varkappa,varrho,varphi,varepsilon,varphi
12  }
13  {
14    \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
15    \tl_set:cx {up ##1} { \exp_not:N \symup \exp_not:c { ##1 } }
16    \tl_set:cx {it ##1} { \exp_not:N \symit \exp_not:c { ##1 } }
17  }
18 }
```

### 16.2 Unicode radicals

Make sure `\Uroot` is defined in the case where the  $\LaTeX$  kernel doesn't make it available with its native name.

```
19 (*LU)
20 \cs_if_exist:NF \Uroot
21 { \cs_new_eq:NN \Uroot \luatexUroot }
22 (/LU)
23 \AtBeginDocument{\@@_redefine_radical:}
24 \cs_new:Nn \@@_redefine_radical:
25 (*XE)
26 {
27   \ifpackageloaded { amsmath } { }
28   {
```

`\r@@t` #1 : A mathstyle (for `\mathpalette`)  
#2 : Leading superscript for the sqrt sign  
A re-implementation of L<sup>A</sup>T<sub>E</sub>X's hard-coded n-root sign using the appropriate `\fontdimens`.

```

29   \cs_set_nopar:Npn \r@@@t ##1 ##2
30   {
31     \hbox_set:Nn \l_tmpa_box
32     {
33       \c_math_toggle_token
34       \m@th
35       ##1
36       \sqrtsign { ##2 }
37       \c_math_toggle_token
38     }
39     \@@_mathstyle_scale:Nnn ##1 { \kern }
40     { \fontdimen 63 \l_@@_font }
41     \box_move_up:nn
42     {
43       (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
44       * \number \fontdimen 65 \l_@@_font / 100
45     }
46     { \box_use:N \rootbox }
47     \@@_mathstyle_scale:Nnn ##1 { \kern }
48     { \fontdimen 64 \l_@@_font }
49     \box_use_clear:N \l_tmpa_box
50   }
51 }
52 }
53 </XE>
54 <*LU>
55 {
56   \ifpackageloaded { amsmath } { }
57   {

```

`\root` Redefine this macro for LuaT<sub>E</sub>X, which provides us a nice primitive to use.

```

58   \cs_set:Npn \root ##1 \of ##2
59   {
60     \Uroot \l_@@_radical_sqrt_tl { ##1 } { ##2 }
61   }
62 }
63 }
64 </LU>

```

### 16.2.1 Active fractions

Active fractions can be setup independently of any maths font definition; all it requires is a mapping from the Unicode input chars to the relevant L<sup>A</sup>T<sub>E</sub>X fraction declaration.

```

65 \cs_new:Npn \@@_define_active_frac:Nw #1 #2/#3
66 {
67   \char_set_catcode_active:N #1
68   \@@_char_gmake_mathactive:N #1
69   \tl_rescan:nn
70   {
71     \catcode`\_ =11\relax
72     \catcode`\: =11\relax
73   }
74   {
75     \cs_gset:Npx #1
76     {
77       \bool_if:NTF \l_@@_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
78       {#2} {#3}
79     }
80   }
81 }

```

These are redefined for each math font selection in case the active-frac feature changes.

```

82 \cs_new:Npn \@@_setup_active_frac:
83 {
84   \group_begin:
85   \@@_define_active_frac:Nw ^^2189 0/3
86   \@@_define_active_frac:Nw ^^2152 1/{10}
87   \@@_define_active_frac:Nw ^^2151 1/9
88   \@@_define_active_frac:Nw ^^215b 1/8
89   \@@_define_active_frac:Nw ^^2150 1/7
90   \@@_define_active_frac:Nw ^^2159 1/6
91   \@@_define_active_frac:Nw ^^2155 1/5
92   \@@_define_active_frac:Nw ^^00bc 1/4
93   \@@_define_active_frac:Nw ^^2153 1/3
94   \@@_define_active_frac:Nw ^^215c 3/8
95   \@@_define_active_frac:Nw ^^2156 2/5
96   \@@_define_active_frac:Nw ^^00bd 1/2
97   \@@_define_active_frac:Nw ^^2157 3/5
98   \@@_define_active_frac:Nw ^^215d 5/8
99   \@@_define_active_frac:Nw ^^2154 2/3
100  \@@_define_active_frac:Nw ^^00be 3/4
101  \@@_define_active_frac:Nw ^^2158 4/5
102  \@@_define_active_frac:Nw ^^215a 5/6
103  \@@_define_active_frac:Nw ^^215e 7/8
104  \group_end:
105 }
106 \@@_setup_active_frac:

```

### 16.3 *Synonyms and all the rest*

These are symbols with multiple names. Eventually to be taken care of automatically by the maths characters database.

```
107 \protected\def\to{\rightarrow}
108 \protected\def\le{\leq}
109 \protected\def\ge{\geq}
110 \protected\def\neq{\neq}
111 \protected\def\triangle{\mathord{\bigtriangleup}}
112 \protected\def\bigcirc{\mdlgwhtcircle}
113 \protected\def\circ{\vysmwhtcircle}
114 \protected\def\bullet{\smbkcircle}
115 \protected\def\mathyen{\yen}
116 \protected\def\mathsterling{\sterling}
117 \protected\def\diamond{\smwhtdiamond}
118 \protected\def\emptyset{\varnothing}
119 \protected\def\hbar{\hslash}
120 \protected\def\land{\wedge}
121 \protected\def\lor{\vee}
122 \protected\def\owns{\ni}
123 \protected\def\gets{\leftarrow}
124 \protected\def\mathring{\ocirc}
125 \protected\def\not{\neg}
126 \protected\def\longdivision{\longdivisionsign}
```

These are somewhat odd: (and their usual Unicode uprightness does not match their amssymb glyphs)

```
127 \protected\def\backepsilon{\upbackepsilon}
128 \protected\def\eth{\matheth}
```

These are names that are ‘frozen’ in HTML but have dumb names:

```
129 \protected\def\dbkarow {\dbkarrow}
130 \protected\def\drbkarow{\drbkarrow}
131 \protected\def\hksearrow{\hksearrow}
132 \protected\def\hkswarrow{\hkswarrow}
```

Due to the magic of OpenType math, big operators are automatically enlarged when necessary. Since there isn’t a separate unicode glyph for ‘small integral’, I’m not sure if there is a better way to do this:

```
133 \protected\def\smallint{\mathop{\textstyle\int}\limits}
```

`\underbar`

```
134 \cs_set_eq:NN \latex_underbar:n \underbar
135 \renewcommand\underbar
136 {
137   \mode_if_math:TF \mathunderbar \latex_underbar:n
138 }
```

`\colon` Define `\colon` as a mathpunct ‘:’. This is wrong: it should be U+003A colon instead! We hope no-one will notice.

```

139 \ifpackageloaded{amsmath}
140 {
141 % define their own colon, perhaps I should just steal it. (It does look much bet-
ter.)
142 }
143 {
144 \cs_set_protected:Npn \colon
145 {
146 \bool_if:NTF \g_@@_literal_colon_bool {;} { \mathpunct{:} }
147 }
148 }

```

`\digamma` I might end up just changing these in the table.

```

\Digamma
149 \protected\def\digamma{\updigamma}
150 \protected\def\Digamma{\upDigamma}

```

### *Symbols*

```

151 \cs_set_protected:Npn \l {\Vert}
\mathinner items:
152 \cs_set_protected:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
153 \cs_set_protected:Npn \cdots {\mathinner{\unicodcdots}}
154 \cs_set_eq:NN \@@_text_slash: \slash
155 \cs_set_protected:Npn \slash
156 {
157 \mode_if_math:TF {\mathslash} {\@@_text_slash:}
158 }

```

#### 16.3.1 `\not`

The situation of `\not` symbol is currently messy, in Unicode it is defined as a combining mark so naturally it should be treated as a math accent, however neither Lua $\TeX$  nor Xe $\TeX$  correctly place it as it needs special treatment compared to other accents, furthermore a math accent changes the spacing of its nucleus, so `\not=` will be spaced as an ordinary not relational symbol, which is undesired.

Here modify `\not` to a macro that tries to use predefined negated symbols, which would give better results in most cases, until there is more robust solution in the engines.

This code is based on an answer to a TeX – Stack Exchange question by Enrico Gregorio<sup>3</sup>.

```

159 \cs_new:Npn \@@_newnot:N #1
160 {
161 \tl_set:Nx \l_not_token_name_tl { \token_to_str:N #1 }
162 \exp_args:Nx \tl_if_empty:nF { \tl_tail:V \l_not_token_name_tl }
163 {
164 \tl_set:Nx \l_not_token_name_tl { \tl_tail:V \l_not_token_name_tl }

```

<sup>3</sup><http://tex.stackexchange.com/a/47260/729>

```

165 }
166 \cs_if_exist:cTF { n \l_not_token_name_tl }
167 {
168   \use:c { n \l_not_token_name_tl }
169 }
170 {
171   \cs_if_exist:cTF { not \l_not_token_name_tl }
172   {
173     \use:c { not \l_not_token_name_tl }
174   }
175   {
176     \@@_oldnot: #1
177   }
178 }
179 }

180 \cs_set_eq:NN \@@_oldnot: \not
181 \AtBeginDocument{\cs_set_eq:NN \not \@@_newnot:N}

182 \cs_new_protected_nopar:Nn \@@_setup_negations:
183 {
184   \cs_gset:cpn { not= } { \neq }
185   \cs_gset:cpn { not< } { \nless }
186   \cs_gset:cpn { not> } { \ngtr }
187   \cs_gset:Npn \ngets { \leftarrow }
188   \cs_gset:Npn \nsimeq { \simeq }
189   \cs_gset:Npn \nequal { \ne }
190   \cs_gset:Npn \nle { \leq }
191   \cs_gset:Npn \nge { \geq }
192   \cs_gset:Npn \ngreater { \ngtr }
193   \cs_gset:Npn \nforksnot { \forks }
194 }
195 </package>

```

## File XVII

# um-code-primes.dtx

## 17 Primes

1 *(\*package)*

We need a new ‘prime’ algorithm. Unicode math has four pre-drawn prime glyphs.

U+2032 prime (`\prime`):  $x'$

U+2033 double prime (`\dprime`):  $x''$

U+2034 triple prime (`\trprime`):  $x'''$

U+2057 quadruple prime (`\qprime`):  $x''''$

As you can see, they’re all drawn at the correct height without being superscripted. However, in a correctly behaving OpenType font, we also see different behaviour after the `ssty` feature is applied:

$x'$   $x''$   $x'''$   $x''''$

The glyphs are now ‘full size’ so that when placed inside a superscript, their shape will match the originally sized ones. Many thanks to Ross Mills of Tiro Typeworks for originally pointing out this behaviour.

In regular  $\text{\LaTeX}$ , primes can be entered with the straight quote character `'`, and multiple straight quotes chain together to produce multiple primes. Better results can be achieved in `unicode-math` by chaining multiple single primes into a pre-drawn multi-prime glyph; consider  $x''''$  vs.  $x'''$ .

For Unicode maths, we wish to conserve this behaviour and augment it with the possibility of adding any combination of Unicode prime or any of the  $n$ -prime characters. E.g., the user might copy-paste a double prime from another source and then later type another single prime after it; the output should be the triple prime.

Our algorithm is:

- Prime encountered; `pcount=1`.
- Scan ahead; if prime: `pcount:=pcount+1`; repeat.
- If not prime, stop scanning.
- If `pcount=1`, `\prime`, end.
- If `pcount=2`, check `\dprime`; if it exists, use it, end; if not, goto last step.
- Ditto `pcount=3` & `\trprime`.
- Ditto `pcount=4` & `\qprime`.
- If `pcount>4` or the glyph doesn’t exist, insert `pcount \primes` with `\primekern` between each.

This is a wrapper to insert a superscript; if there is a subsequent trailing superscript, then it is included within the insertion.

2 `\cs_new:Nn \@@_arg_i_before_egroup:n {#1\egroup}`

```

3 \cs_new:Nn \@@_superscript:n
4 {
5   ^\bgroup #1
6   \peek_meaning_remove:NTF ^ \@@_arg_i_before_egroup:n \egroup
7 }

8 \cs_new:Nn \@@_nprimes:Nn
9 {
10  \@@_superscript:n
11  {
12    #1
13    \prg_replicate:nn {#2-1} { \mskip \g_@@_primekern_muskip #1 }
14  }
15 }

16 \cs_new:Nn \@@_nprimes_select:nn
17 {
18  \int_case:nnF {#2}
19  {
20    {1} { \@@_superscript:n {#1} }
21    {2} {
22      \@@_glyph_if_exist:NnTF \l_@@_font {"2033}
23      { \@@_superscript:n {\@@_prime_double_mchar} }
24      { \@@_nprimes:Nn #1 {#2} }
25    }
26    {3} {
27      \@@_glyph_if_exist:NnTF \l_@@_font {"2034}
28      { \@@_superscript:n {\@@_prime_triple_mchar} }
29      { \@@_nprimes:Nn #1 {#2} }
30    }
31    {4} {
32      \@@_glyph_if_exist:NnTF \l_@@_font {"2057}
33      { \@@_superscript:n {\@@_prime_quad_mchar} }
34      { \@@_nprimes:Nn #1 {#2} }
35    }
36  }
37  {
38    \@@_nprimes:Nn #1 {#2}
39  }
40 }

41 \cs_new:Nn \@@_nbackprimes_select:nn
42 {
43  \int_case:nnF {#2}
44  {
45    {1} { \@@_superscript:n {#1} }
46    {2} {
47      \@@_glyph_if_exist:NnTF \l_@@_font {"2036}
48      { \@@_superscript:n {\@@_backprime_double_mchar} }
49      { \@@_nprimes:Nn #1 {#2} }
50    }

```

```

51 {3} {
52   \@@_glyph_if_exist:NnTF \l_@@_font {"2037}
53   { \@@_superscript:n { \@@_backprime_triple_mchar } }
54   { \@@_nprimes:Nn #1 {#2} }
55 }
56 }
57 {
58   \@@_nprimes:Nn #1 {#2}
59 }
60 }

```

Scanning is annoying because I'm too lazy to do it for the general case.

```

61 \cs_new:Npn \@@_scan_prime:
62 {
63   \cs_set_eq:NN \@@_superscript:n \use:n
64   \int_zero:N \l_@@_primecount_int
65   \@@_scanprime_collect:N \@@_prime_single_mchar
66 }
67 \cs_new:Npn \@@_scan_dprime:
68 {
69   \cs_set_eq:NN \@@_superscript:n \use:n
70   \int_set:Nn \l_@@_primecount_int {1}
71   \@@_scanprime_collect:N \@@_prime_single_mchar
72 }
73 \cs_new:Npn \@@_scan_trprime:
74 {
75   \cs_set_eq:NN \@@_superscript:n \use:n
76   \int_set:Nn \l_@@_primecount_int {2}
77   \@@_scanprime_collect:N \@@_prime_single_mchar
78 }
79 \cs_new:Npn \@@_scan_qprime:
80 {
81   \cs_set_eq:NN \@@_superscript:n \use:n
82   \int_set:Nn \l_@@_primecount_int {3}
83   \@@_scanprime_collect:N \@@_prime_single_mchar
84 }
85 \cs_new:Npn \@@_scan_sup_prime:
86 {
87   \int_zero:N \l_@@_primecount_int
88   \@@_scanprime_collect:N \@@_prime_single_mchar
89 }
90 \cs_new:Npn \@@_scan_sup_dprime:
91 {
92   \int_set:Nn \l_@@_primecount_int {1}
93   \@@_scanprime_collect:N \@@_prime_single_mchar
94 }
95 \cs_new:Npn \@@_scan_sup_trprime:
96 {
97   \int_set:Nn \l_@@_primecount_int {2}
98   \@@_scanprime_collect:N \@@_prime_single_mchar

```

```

99 }
100 \cs_new:Npn \@@_scan_sup_qprime:
101 {
102   \int_set:Nn \l_@@_primecount_int {3}
103   \@@_scanprime_collect:N \@@_prime_single_mchar
104 }
105 \cs_new:Nn \@@_scanprime_collect:N
106 {
107   \int_incr:N \l_@@_primecount_int
108   \peek_meaning_remove:NTF '
109   { \@@_scanprime_collect:N #1 }
110   {
111     \peek_meaning_remove:NTF \@@_scan_prime:
112     { \@@_scanprime_collect:N #1 }
113     {
114       \peek_meaning_remove:NTF ^^^^2032
115       { \@@_scanprime_collect:N #1 }
116       {
117         \peek_meaning_remove:NTF \@@_scan_dprime:
118         {
119           \int_incr:N \l_@@_primecount_int
120           \@@_scanprime_collect:N #1
121         }
122         {
123           \peek_meaning_remove:NTF ^^^^2033
124           {
125             \int_incr:N \l_@@_primecount_int
126             \@@_scanprime_collect:N #1
127           }
128           {
129             \peek_meaning_remove:NTF \@@_scan_trprime:
130             {
131               \int_add:Nn \l_@@_primecount_int {2}
132               \@@_scanprime_collect:N #1
133             }
134             {
135               \peek_meaning_remove:NTF ^^^^2034
136               {
137                 \int_add:Nn \l_@@_primecount_int {2}
138                 \@@_scanprime_collect:N #1
139               }
140               {
141                 \peek_meaning_remove:NTF \@@_scan_qprime:
142                 {
143                   \int_add:Nn \l_@@_primecount_int {3}
144                   \@@_scanprime_collect:N #1
145                 }
146                 {
147                   \peek_meaning_remove:NTF ^^^^2057

```

```

148         {
149             \int_add:Nn \l_@@_primecount_int {3}
150             \@@_scanprime_collect:N #1
151         }
152         {
153             \@@_nprimes_select:nn {#1} {\l_@@_primecount_int}
154         }
155     }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }

164 \cs_new:Npn \@@_scan_backprime:
165 {
166     \cs_set_eq:NN \@@_superscript:n \use:n
167     \int_zero:N \l_@@_primecount_int
168     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
169 }
170 \cs_new:Npn \@@_scan_backdprime:
171 {
172     \cs_set_eq:NN \@@_superscript:n \use:n
173     \int_set:Nn \l_@@_primecount_int {1}
174     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
175 }
176 \cs_new:Npn \@@_scan_backtrprime:
177 {
178     \cs_set_eq:NN \@@_superscript:n \use:n
179     \int_set:Nn \l_@@_primecount_int {2}
180     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
181 }
182 \cs_new:Npn \@@_scan_sup_backprime:
183 {
184     \int_zero:N \l_@@_primecount_int
185     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
186 }
187 \cs_new:Npn \@@_scan_sup_backdprime:
188 {
189     \int_set:Nn \l_@@_primecount_int {1}
190     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
191 }
192 \cs_new:Npn \@@_scan_sup_backtrprime:
193 {
194     \int_set:Nn \l_@@_primecount_int {2}
195     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
196 }

```

```

197 \cs_new:Nn \@@_scanbackprime_collect:N
198 {
199   \int_incr:N \l_@@_primecount_int
200   \peek_meaning_remove:NTF `
201   {
202     \@@_scanbackprime_collect:N #1
203   }
204   {
205     \peek_meaning_remove:NTF \@@_scan_backprime:
206     {
207       \@@_scanbackprime_collect:N #1
208     }
209     {
210       \peek_meaning_remove:NTF ^^^^2035
211       {
212         \@@_scanbackprime_collect:N #1
213       }
214       {
215         \peek_meaning_remove:NTF \@@_scan_backdprime:
216         {
217           \int_incr:N \l_@@_primecount_int
218           \@@_scanbackprime_collect:N #1
219         }
220         {
221           \peek_meaning_remove:NTF ^^^^2036
222           {
223             \int_incr:N \l_@@_primecount_int
224             \@@_scanbackprime_collect:N #1
225           }
226           {
227             \peek_meaning_remove:NTF \@@_scan_backtrprime:
228             {
229               \int_add:Nn \l_@@_primecount_int {2}
230               \@@_scanbackprime_collect:N #1
231             }
232             {
233               \peek_meaning_remove:NTF ^^^^2037
234               {
235                 \int_add:Nn \l_@@_primecount_int {2}
236                 \@@_scanbackprime_collect:N #1
237               }
238               {
239                 \@@_nbackprimes_select:nn {#1} {\l_@@_primecount_int}
240               }
241             }
242           }
243         }
244       }
245     }

```

```

246     }
247 }
248 \AtBeginDocument { \@@_define_prime_commands: \@@_define_prime_chars: }
249 \cs_new:Nn \@@_define_prime_commands:
250 {
251   \cs_set_eq:NN \prime      \@@_prime_single_mchar
252   \cs_set_eq:NN \dprime    \@@_prime_double_mchar
253   \cs_set_eq:NN \trprime   \@@_prime_triple_mchar
254   \cs_set_eq:NN \qprime    \@@_prime_quad_mchar
255   \cs_set_eq:NN \backprime \@@_backprime_single_mchar
256   \cs_set_eq:NN \backdprime \@@_backprime_double_mchar
257   \cs_set_eq:NN \backtrprime \@@_backprime_triple_mchar
258 }
259 \group_begin:
260   \char_set_catcode_active:N \'
261   \char_set_catcode_active:N `
262   \char_set_catcode_active:n {"2032}
263   \char_set_catcode_active:n {"2033}
264   \char_set_catcode_active:n {"2034}
265   \char_set_catcode_active:n {"2057}
266   \char_set_catcode_active:n {"2035}
267   \char_set_catcode_active:n {"2036}
268   \char_set_catcode_active:n {"2037}
269   \cs_gset:Nn \@@_define_prime_chars:
270     {
271       \cs_set_eq:NN '      \@@_scan_sup_prime:
272       \cs_set_eq:NN ^^^^2032 \@@_scan_sup_prime:
273       \cs_set_eq:NN ^^^^2033 \@@_scan_sup_dprime:
274       \cs_set_eq:NN ^^^^2034 \@@_scan_sup_trprime:
275       \cs_set_eq:NN ^^^^2057 \@@_scan_sup_qprime:
276       \cs_set_eq:NN `     \@@_scan_sup_backprime:
277       \cs_set_eq:NN ^^^^2035 \@@_scan_sup_backprime:
278       \cs_set_eq:NN ^^^^2036 \@@_scan_sup_backdprime:
279       \cs_set_eq:NN ^^^^2037 \@@_scan_sup_backtrprime:
280     }
281 \group_end:
282 </package>

```

## File XVIII

# um-code-sscript.dtx

### 18 Unicode sub- and super-scripts

1 *(\*package)*

The idea here is to enter a scanning state after a superscript or subscript is encountered. If subsequent superscripts or subscripts (resp.) are found, they are lumped together. Each sub/super has a corresponding regular size glyph which is used by X<sub>Y</sub>T<sub>E</sub>X to typeset the results; this means that the actual subscript/superscript glyphs are never seen in the output document — they are only used as input characters.

Open question: should the superscript-like ‘modifiers’ (U+1D2C modifier capital letter a and on) be included here?

2 `\group_begin:`

*Superscripts* Populate a property list with superscript characters; themselves as their key, and their replacement as each key’s value. Then make the superscript active and bind it to the scanning function.

`\scantokens` makes this process much simpler since we can activate the char and assign its meaning in one step.

```
3 \cs_new:Nn \@@_setup_active_superscript:nn
4 {
5   \prop_gput:Nnn \g_@@_supers_prop {#1} {#2}
6   \char_set_catcode_active:N #1
7   \@@_char_gmake_mathactive:N #1
8   \scantokens
9   {
10    \cs_gset:Npn #1
11    {
12      \tl_set:Nn \l_@@_ss_chain_tl {#2}
13      \cs_set_eq:NN \@@_sub_or_super:n \sp
14      \tl_set:Nn \l_@@_tmpa_tl {supers}
15      \@@_scan_sscript:
16    }
17  }
18 }
```

Bam:

```
19 \@@_setup_active_superscript:nn {^^^2070} {0}
20 \@@_setup_active_superscript:nn {^^^00b9} {1}
21 \@@_setup_active_superscript:nn {^^^00b2} {2}
22 \@@_setup_active_superscript:nn {^^^00b3} {3}
23 \@@_setup_active_superscript:nn {^^^2074} {4}
24 \@@_setup_active_superscript:nn {^^^2075} {5}
25 \@@_setup_active_superscript:nn {^^^2076} {6}
26 \@@_setup_active_superscript:nn {^^^2077} {7}
```

```

27 \@@_setup_active_superscript:nn {^^^^2078} {8}
28 \@@_setup_active_superscript:nn {^^^^2079} {9}
29 \@@_setup_active_superscript:nn {^^^^207a} {+}
30 \@@_setup_active_superscript:nn {^^^^207b} {-}
31 \@@_setup_active_superscript:nn {^^^^207c} {=}
32 \@@_setup_active_superscript:nn {^^^^207d} {(}
33 \@@_setup_active_superscript:nn {^^^^207e} {)}
34 \@@_setup_active_superscript:nn {^^^^2071} {i}
35 \@@_setup_active_superscript:nn {^^^^207f} {n}
36 \@@_setup_active_superscript:nn {^^^^02b0} {h}
37 \@@_setup_active_superscript:nn {^^^^02b2} {j}
38 \@@_setup_active_superscript:nn {^^^^02b3} {r}
39 \@@_setup_active_superscript:nn {^^^^02b7} {w}
40 \@@_setup_active_superscript:nn {^^^^02b8} {y}

```

*Subscripts* Ditto above.

```

41 \cs_new:Nn \@@_setup_active_subscript:nn
42 {
43   \prop_gput:Nnn \g_@@_subs_prop {#1} {#2}
44   \char_set_catcode_active:N #1
45   \@@_char_gmake_mathactive:N #1
46   \scantokens
47   {
48     \cs_gset:Npn #1
49     {
50       \tl_set:Nn \l_@@_ss_chain_tl {#2}
51       \cs_set_eq:NN \@@_sub_or_super:n \sb
52       \tl_set:Nn \l_@@_tmpa_tl {subs}
53       \@@_scan_sscript:
54     }
55   }
56 }

```

A few more subscripts than superscripts:

```

57 \@@_setup_active_subscript:nn {^^^^2080} {0}
58 \@@_setup_active_subscript:nn {^^^^2081} {1}
59 \@@_setup_active_subscript:nn {^^^^2082} {2}
60 \@@_setup_active_subscript:nn {^^^^2083} {3}
61 \@@_setup_active_subscript:nn {^^^^2084} {4}
62 \@@_setup_active_subscript:nn {^^^^2085} {5}
63 \@@_setup_active_subscript:nn {^^^^2086} {6}
64 \@@_setup_active_subscript:nn {^^^^2087} {7}
65 \@@_setup_active_subscript:nn {^^^^2088} {8}
66 \@@_setup_active_subscript:nn {^^^^2089} {9}
67 \@@_setup_active_subscript:nn {^^^^208a} {+}
68 \@@_setup_active_subscript:nn {^^^^208b} {-}
69 \@@_setup_active_subscript:nn {^^^^208c} {=}
70 \@@_setup_active_subscript:nn {^^^^208d} {(}
71 \@@_setup_active_subscript:nn {^^^^208e} {)}

```

```

72 \@@_setup_active_subscript:nn {^^^^2090} {a}
73 \@@_setup_active_subscript:nn {^^^^2091} {e}
74 \@@_setup_active_subscript:nn {^^^^2095} {h}
75 \@@_setup_active_subscript:nn {^^^^1d62} {i}
76 \@@_setup_active_subscript:nn {^^^^2c7c} {j}
77 \@@_setup_active_subscript:nn {^^^^2096} {k}
78 \@@_setup_active_subscript:nn {^^^^2097} {l}
79 \@@_setup_active_subscript:nn {^^^^2098} {m}
80 \@@_setup_active_subscript:nn {^^^^2099} {n}
81 \@@_setup_active_subscript:nn {^^^^2092} {o}
82 \@@_setup_active_subscript:nn {^^^^209a} {p}
83 \@@_setup_active_subscript:nn {^^^^1d63} {r}
84 \@@_setup_active_subscript:nn {^^^^209b} {s}
85 \@@_setup_active_subscript:nn {^^^^209c} {t}
86 \@@_setup_active_subscript:nn {^^^^1d64} {u}
87 \@@_setup_active_subscript:nn {^^^^1d65} {v}
88 \@@_setup_active_subscript:nn {^^^^2093} {x}
89 \@@_setup_active_subscript:nn {^^^^1d66} {\beta}
90 \@@_setup_active_subscript:nn {^^^^1d67} {\gamma}
91 \@@_setup_active_subscript:nn {^^^^1d68} {\rho}
92 \@@_setup_active_subscript:nn {^^^^1d69} {\phi}
93 \@@_setup_active_subscript:nn {^^^^1d6a} {\chi}

94 \group_end:

```

The scanning command, which collects a chain of subscripts or a chain of superscripts and then typesets what it has collected.

```

95 \cs_new:Npn \@@_scan_sscript:
96 {
97   \@@_scan_sscript:TF
98   {
99     \@@_scan_sscript:
100   }
101   {
102     \@@_sub_or_super:n {\l_@@_ss_chain_tl}
103   }
104 }

```

We do not skip spaces when scanning ahead, and we explicitly wish to bail out on encountering a space or a brace. These cases are filtered using `\peek_N_type:TF`. Otherwise the token can be taken as an N-type argument. Then we search for it in the appropriate property list (`\l_@@_tmpa_tl` is `subs` or `supers`). If found, add the value to the current chain of sub/superscripts. Remember to put the character back in the input otherwise. The `\group_align_safe_begin:` and `\group_align_safe_end:` are needed in case #3 is `&`.

```

105 \cs_new:Npn \@@_scan_sscript:TF #1#2
106 {
107   \peek_N_type:TF
108   {
109     \group_align_safe_begin:

```

```
110 \@@_scan_sscript_aux:nnN {#1} {#2}
111 }
112 {#2}
113 }
114 \cs_new:Npn \@@_scan_sscript_aux:nnN #1#2#3
115 {
116 \prop_get:cnTF {g_@_l_@_tmpa_tl _prop} {#3} \l_@_tmpb_tl
117 {
118 \tl_put_right:NV \l_@_ss_chain_tl \l_@_tmpb_tl
119 \group_align_safe_end:
120 #1
121 }
122 { \group_align_safe_end: #2 #3 }
123 }
124 </package>
```

## File XIX

# um-code-compat.dtx

## 19 Compatibility

1 *(\*package)*

```
\@@_check_and_fix:NNnnn #1 : command
                          #2 : factory command
                          #3 : parameter text
                          #4 : expected replacement text
                          #5 : new replacement text for LuaTeX
                          #6 : new replacement text for XeTeX
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 over-
written using the <parameter text> and the <new replacement text for LuaTeX> or the
<new replacement text for XeTeX>, depending on the engine. Otherwise issue a warn-
ing and don't overwrite.
2 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnn
3 {
4   \cs_if_exist:NT #1
5   {
6     \token_if_macro:NTF #1
7     {
8       \group_begin:
9       #2 \@@_tmpa:w #3 { #4 }
10      \cs_if_eq:NNTF #1 \@@_tmpa:w
11      {
12        \msg_info:nxx { unicode-math } { patch-macro }
13        { \token_to_str:N #1 }
14        \group_end:
15        #2 #1 #3
16      (XE)      { #6 }
17      (LU)      { #5 }
18      }
19      {
20        \msg_warning:nxxx { unicode-math } { wrong-meaning }
21        { \token_to_str:N #1 } { \token_to_meaning:N #1 }
22        { \token_to_meaning:N \@@_tmpa:w }
23        \group_end:
24      }
25    }
26    {
27      \msg_warning:nxx { unicode-math } { macro-expected }
28      { \token_to_str:N #1 }
29    }
```

```

30 }
31 }

```

```

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

```

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  $\langle new replacement text \rangle$ . Otherwise issue a warning and don't overwrite.

```

32 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnn
33 {
34   \@@_check_and_fix:NNnnnn #1 #2 { #3 } { #4 } { #5 } { #5 }
35 }

```

```

\@@_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 Xe<sub>La</sub>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.

```

36 \cs_new_protected_nopar:Nn \@@_check_and_fix_luatex:NNnnn
37 {
38   (LU) \@@_check_and_fix:NNnnn #1 #2 { #3 } { #4 } { #5 }
39 }
40 \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.

```

41 \AtEndOfPackageFile * {url}
42 {
43   \tl_put_left:Nn \url@FormatString { \@@_switchto_up: }
44   \tl_put_right:Nn \url@Specials
45   {
46     \do\{\mathchar``\}
47     \do\'\{\mathchar``\}

```

```

48   \do\${\mathchar`\$}
49   \do\&{\mathchar`\&}
50   }
51 }

```

*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:

```

52 \AtEndOfPackageFile * {amsmath}
53 {
54 (*XE)
55   \tl_remove_once:Nn \@begindocumenthook
56   {
57     \mathchardef\std@minus\mathcode`\-\relax
58     \mathchardef\std@equal\mathcode`\=\relax
59   }
60   \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
61   \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
62 (/XE)
63   \cs_set:Npn \@cdots {\mathinner{\cdots}}
64   \cs_set_eq:NN \dotsb@ \cdots

```

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

```

65 (*XE)
66   \def \resetMathstrut@
67   {%
68     \setbox\z@\hbox{$(%)%}
69     \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
70   }

```

The subarray environment uses inappropriate font dimensions.

```

71   \@_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 }
72   {
73     \vcenter
74     \bgroup
75     \Let@
76     \restore@math@cr
77     \default@tag
78     \baselineskip \fontdimen 10~ \scriptfont \tw@
79     \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
80     \lineskip \thr@@@ \fontdimen 8~ \scriptfont \thr@@@
81     \lineskiplimit \lineskip
82     \ialign
83     \bgroup
84     \ifx c #1 \hfil \fi
85     $ \m@th \scriptstyle ## $
86     \hfil
87     \crrc
88   }

```

```

89 {
90   \vcenter
91   \c_group_begin_token
92   \Let@
93   \restore@math@cr
94   \default@tag
95   \skip_set:Nn \baselineskip
96   {

```

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

```

97     \@@_stack_num_up:N \scriptstyle
98     + \@@_stack_denom_down:N \scriptstyle
99   }

```

Here we use the minimum stack gap.

```

100   \lineskip \@@_stack_vgap:N \scriptstyle
101   \lineskiplimit \lineskip
102   \ialign
103   \c_group_begin_token
104   \token_if_eq_meaning:NNT c #1 { \hfil }
105   \c_math_toggle_token
106   \m@th
107   \scriptstyle
108   \c_parameter_token \c_parameter_token
109   \c_math_toggle_token
110   \hfil
111   \crcr
112   }
113 </XE>

```

The roots need a complete rework.

```

114 \@@_check_and_fix luatex:NNnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 }
115 {
116   \setbox \rootbox \hbox
117   {
118     $ \m@th \scriptscriptstyle { #1 } $
119   }
120   \mathchoice
121     { \r@@@t \displaystyle { #2 } }
122     { \r@@@t \textstyle { #2 } }~
123     { \r@@@t \scriptstyle { #2 } }
124     { \r@@@t \scriptscriptstyle { #2 } }
125   \egroup
126 }
127 {
128   \bool_if:nTF
129     {
130       \int_compare_p:nNn { \uproot@ } = { \c_zero }
131       && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
132     }
133   {

```

```

134     \Uroot \l_@@_radical_sqrt_tl { #1 } { #2 }
135   }
136   {
137     \hbox_set:Nn \rootbox
138     {
139       \c_math_toggle_token
140       \m@th
141       \scriptscriptstyle { #1 }
142       \c_math_toggle_token
143     }
144     \mathchoice
145       { \r@@@t \displaystyle { #2 } }
146       { \r@@@t \textstyle { #2 } }
147       { \r@@@t \scriptstyle { #2 } }
148       { \r@@@t \scriptscriptstyle { #2 } }
149   }
150   \c_group_end_token
151 }
152 \@@_check_and_fix:NNnnn \r@@@t \cs_set_nopar:Npn { #1 #2 }
153 {
154   \setboxz@h { $ \m@th #1 \sqrt{sign { #2 } } $ }
155   \dimen@ \ht\z@
156   \advance \dimen@ -\dp\z@
157   \setbox\@ne \hbox { $ \m@th #1 \mskip \uproot@ mu $ }
158   \advance \dimen@ by 1.667 \wd\@ne
159   \mkern -\leftroot@ mu
160   \mkern 5mu
161   \raise .6\dimen@ \copy\rootbox
162   \mkern -10mu
163   \mkern \leftroot@ mu
164   \boxz@
165 }
166 {
167   \hbox_set:Nn \l_tmpa_box
168   {
169     \c_math_toggle_token
170     \m@th
171     #1
172     \mskip \uproot@ mu
173     \c_math_toggle_token
174   }
175   \Uroot \l_@@_radical_sqrt_tl
176   {
177     \box_move_up:nn { \box_wd:N \l_tmpa_box }
178     {
179       \hbox:n
180       {
181         \c_math_toggle_token
182         \m@th

```

```

183     \mkern -\leftroot@ mu
184     \box_use:N \rootbox
185     \mkern \leftroot@ mu
186     \c_math_toggle_token
187   }
188 }
189 }
190 { #2 }
191 }
192 {
193   \hbox_set:Nn \l_tmpa_box
194   {
195     \c_math_toggle_token
196     \m@th
197     #1
198     \sqrtsign { #2 }
199     \c_math_toggle_token
200   }
201   \hbox_set:Nn \l_tmpb_box
202   {
203     \c_math_toggle_token
204     \m@th
205     #1
206     \mskip \uproot@ mu
207     \c_math_toggle_token
208   }
209   \mkern -\leftroot@ mu
210   \@_mathstyle_scale:Nnn #1 { \kern }
211   {
212     \fontdimen 63 \l_@@_font
213   }
214   \box_move_up:nn
215   {
216     \box_wd:N \l_tmpb_box
217     + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
218     * \number \fontdimen 65 \l_@@_font / 100
219   }
220   {
221     \box_use:N \rootbox
222   }
223   \@_mathstyle_scale:Nnn #1 { \kern }
224   {
225     \fontdimen 64 \l_@@_font
226   }
227   \mkern \leftroot@ mu
228   \box_use_clear:N \l_tmpa_box
229 }
230 }

```

*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:

```

231 (*XE)
232 \AtEndOfPackageFile * {amsopn}
233 {
234   \cs_set:Npn \newmcodes@
235   {
236     \mathcode`\'39\scan_stop:
237     \mathcode`\*42\scan_stop:
238     \mathcode`\."613A\scan_stop:
239     %% \ifnum\mathcode`\-=45 \else
240     %%   \mathchardef\std@minus\mathcode`\-\relax
241     %% \fi
242     \mathcode`\-45\scan_stop:
243     \mathcode`\-/47\scan_stop:
244     \mathcode`\:"603A\scan_stop:
245   }
246 }
247 (/XE)

```

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

```

248 (*XE)
249 \AtEndOfPackageFile * { mathtools }
250 {
251   \@@_check_and_fix:NNnnn
252   \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }
253   {
254     \sbox \z@
255     {
256       $
257       \m@th
258       #1
259       \nulldelimiterspace = \z@
260       \radical \z@ { #2 }
261       $
262     }
263     \ifx #1 \displaystyle
264       \dimen@ = \fontdimen 8 \textfont 3
265       \advance \dimen@ .25 \fontdimen 5 \textfont 2
266     \else
267       \dimen@ = 1.25 \fontdimen 8
268       \ifx #1 \textstyle
269         \textfont
270       \else
271         \ifx #1 \scriptstyle
272           \scriptfont
273         \else

```

```

274         \scriptscriptfont
275         \fi
276         \fi
277         3
278         \fi
279         \advance \dimen@ -\ht\z@
280         \ht\z@ = -\dimen@
281         \box\z@
282     }

```

The Xe<sub>La</sub>TeX 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.

```

283     {
284     \hbox_set:Nn \l_tmpa_box
285     {
286     \color@setgroup
287     \c_math_toggle_token
288     \m@th
289     #1
290     \dim_zero:N \nulldelimiterspace
291     \XeTeXradical \c_two_hundred_fifty_five \c_zero { #2 }
292     \c_math_toggle_token
293     \color@endgroup
294     }
295     \box_set_ht:Nn \l_tmpa_box
296     {
297     \box_ht:N \l_tmpa_box

```

Here we use the radical vertical gap.

```

298     - \@@_radical_vgap:N #1
299     }
300     \box_use_clear:N \l_tmpa_box
301     }
302 }
303 </XE>

```

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

```

304 \AtEndOfPackageFile * { mathtools }
305 {
306     \cs_set_eq:NN \MToverbracket \overbracket
307     \cs_set_eq:NN \MTunderbracket \underbracket
308
309     \AtBeginDocument
310     {
311     \msg_warning:nn { unicode-math } { mathtools-overbracket }
312

```

```

313 \def\downbracketfill#1#2
314 {%
Original definition used the height of \braced which is not available with Uni-
code fonts, so we are hard coding the 5/18ex suggested by mathtools's documen-
tation.
315         \edef\l_MT_bracketheight_fdim{.27ex}%
316         \downbracketend{#1}{#2}
317         \leaders \vrule \@height #1 \@depth \z@ \hfill
318         \downbracketend{#1}{#2}%
319     }
320 \def\upbracketfill#1#2
321 {%
322         \edef\l_MT_bracketheight_fdim{.27ex}%
323         \upbracketend{#1}{#2}
324         \leaders \vrule \@height \z@ \@depth #1 \hfill
325         \upbracketend{#1}{#2}%
326     }
327 \let\Uoverbracket =\overbracket
328 \let\Uunderbracket=\underbracket
329     \let\overbracket =\MToverbracket
330     \let\underbracket =\MTunderbracket
331 }% 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` Because mathtools uses `\providecommand` `\AtBeginDocument`, we can just define the offending commands here.

```

332 \msg_warning:nn { unicode-math } { mathtools-colon }
333 \NewDocumentCommand \dblcolon { } { \Colon }
334 \NewDocumentCommand \coloneqq { } { \coloneq }
335 \NewDocumentCommand \Coloneqq { } { \Coloneq }
336 \NewDocumentCommand \eqqcolon { } { \eqcolon }
337 }

```

### *colonequals*

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

```

\minuscolon 338 \AtEndOfPackageFile * { colonequals }
\colonequals 339 {
\equalscolon 340 \msg_warning:nn { unicode-math } { colonequals }
\coloncolonequals 341 \RenewDocumentCommand \ratio { } { \mathratio }
342 \RenewDocumentCommand \coloncolon { } { \Colon }
343 \RenewDocumentCommand \minuscolon { } { \dashcolon }
344 \RenewDocumentCommand \colonequals { } { \coloneq }
345 \RenewDocumentCommand \equalscolon { } { \eqcolon }
346 \RenewDocumentCommand \coloncolonequals { } { \Coloneq }
347 }

```

348 </package>

## File XX

# um-code-alphabets.dtx

## 20 *Setting up alphabets*

1 *(\*package)*

### 20.1 *Upright: up*

```
2 \@@_new_alphabet_config:nnn {up} {num}
3 {
4   \@@_set_normal_numbers:nn {up} {#1}
5   \@@_set_mathalphabet_numbers:nnn {up} {up} {#1}
6 }
7
8 \@@_new_alphabet_config:nnn {up} {Latin}
9 {
10  \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {up} {#1} }
11  {
12    \bool_if:NT \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
13  }
14  \@@_set_mathalphabet_Latin:nnn {up} {up,it} {#1}
15  \@@_set_mathalphabet_Latin:nnn {literal} {up} {up}
16  \@@_set_mathalphabet_Latin:nnn {literal} {it} {it}
17 }
18
19 \@@_new_alphabet_config:nnn {up} {latin}
20 {
21  \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_latin:nn {up} {#1} }
22  {
23    \bool_if:NT \g_@@_uplatin_bool
24    {
25      \@@_set_normal_latin:nn      {up,it} {#1}
26      \@@_set_normal_char:nnn      {h} {up,it} {#1}
27      \@@_set_normal_char:nnn {dotlessi} {up,it} {#1}
28      \@@_set_normal_char:nnn {dotlessj} {up,it} {#1}
29    }
30  }
31  \@@_set_mathalphabet_latin:nnn {up} {up,it}{#1}
32  \@@_set_mathalphabet_latin:nnn {literal} {up} {up}
33  \@@_set_mathalphabet_latin:nnn {literal} {it} {it}
34 }
35
36 \@@_new_alphabet_config:nnn {up} {Greek}
37 {
38  \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Greek:nn {up}{#1} }
39  {
40    \bool_if:NT \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
```

```

41 }
42 \@@_set_mathalphabet_Greek:nnn {up} {up,it}{#1}
43 \@@_set_mathalphabet_Greek:nnn {literal} {up} {up}
44 \@@_set_mathalphabet_Greek:nnn {literal} {it} {it}
45 }
46
47 \@@_new_alphabet_config:nnn {up} {greek}
48 {
49   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_greek:nn {up} {#1} }
50   {
51     \bool_if:NT \g_@@_upgreek_bool
52     {
53       \@@_set_normal_greek:nn {up,it} {#1}
54     }
55   }
56   \@@_set_mathalphabet_greek:nnn {up} {up,it} {#1}
57   \@@_set_mathalphabet_greek:nnn {literal} {up} {up}
58   \@@_set_mathalphabet_greek:nnn {literal} {it} {it}
59 }
60
61 \@@_new_alphabet_config:nnn {up} {misc}
62 {
63   \bool_if:NTF \g_@@_literal_Nabla_bool
64   {
65     \@@_set_normal_char:nnn {Nabla}{up}{up}
66   }
67   {
68     \bool_if:NT \g_@@_upNabla_bool
69     {
70       \@@_set_normal_char:nnn {Nabla}{up,it}{up}
71     }
72   }
73   \bool_if:NTF \g_@@_literal_partial_bool
74   {
75     \@@_set_normal_char:nnn {partial}{up}{up}
76   }
77   {
78     \bool_if:NT \g_@@_uppartial_bool
79     {
80       \@@_set_normal_char:nnn {partial}{up,it}{up}
81     }
82   }
83   \@@_set_mathalphabet_pos:nnnn {up} {partial} {up,it} {#1}
84   \@@_set_mathalphabet_pos:nnnn {up} {Nabla} {up,it} {#1}
85   \@@_set_mathalphabet_pos:nnnn {up} {dotlessi} {up,it} {#1}
86   \@@_set_mathalphabet_pos:nnnn {up} {dotlessj} {up,it} {#1}
87 }

```

## 20.2 *Italic: it*

```

88 \@@_new_alphabet_config:nnn {it} {Latin}
89 {
90   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {it} {#1} }
91   {
92     \bool_if:NF \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
93   }
94   \@@_set_mathalphabet_Latin:nnn {it}{up,it}{#1}
95 }
96
97 \@@_new_alphabet_config:nnn {it} {latin}
98 {
99   \bool_if:NTF \g_@@_literal_bool
100   {
101     \@@_set_normal_latin:nn {it} {#1}
102     \@@_set_normal_char:nnn {h}{it}{#1}
103   }
104   {
105     \bool_if:NF \g_@@_uplatin_bool
106     {
107       \@@_set_normal_latin:nn {up,it} {#1}
108       \@@_set_normal_char:nnn {h}{up,it}{#1}
109       \@@_set_normal_char:nnn {dotlessi}{up,it}{#1}
110       \@@_set_normal_char:nnn {dotlessj}{up,it}{#1}
111     }
112   }
113   \@@_set_mathalphabet_latin:nnn {it} {up,it} {#1}
114   \@@_set_mathalphabet_pos:nnnn {it} {dotlessi} {up,it} {#1}
115   \@@_set_mathalphabet_pos:nnnn {it} {dotlessj} {up,it} {#1}
116 }
117
118 \@@_new_alphabet_config:nnn {it} {Greek}
119 {
120   \bool_if:NTF \g_@@_literal_bool
121   {
122     \@@_set_normal_Greek:nn {it}{#1}
123   }
124   {
125     \bool_if:NF \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
126   }
127   \@@_set_mathalphabet_Greek:nnn {it} {up,it}{#1}
128 }
129
130 \@@_new_alphabet_config:nnn {it} {greek}
131 {
132   \bool_if:NTF \g_@@_literal_bool
133   {
134     \@@_set_normal_greek:nn {it} {#1}
135   }
136   {

```

```

137   \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_greek:nn {it,up} {#1} }
138   }
139   \@@_set_mathalphabet_greek:nnn {it} {up,it} {#1}
140 }
141
142 \@@_new_alphabet_config:nnn {it} {misc}
143 {
144   \bool_if:NTF \g_@@_literal_Nabla_bool
145   {
146     \@@_set_normal_char:nnn {Nabla}{it}{it}
147   }
148   {
149     \bool_if:NF \g_@@_upNabla_bool
150     {
151       \@@_set_normal_char:nnn {Nabla}{up,it}{it}
152     }
153   }
154   \bool_if:NTF \g_@@_literal_partial_bool
155   {
156     \@@_set_normal_char:nnn {partial}{it}{it}
157   }
158   {
159     \bool_if:NF \g_@@_uppartial_bool
160     {
161       \@@_set_normal_char:nnn {partial}{up,it}{it}
162     }
163   }
164   \@@_set_mathalphabet_pos:nnnn {it} {partial} {up,it}{#1}
165   \@@_set_mathalphabet_pos:nnnn {it} {Nabla} {up,it}{#1}
166 }

```

### 20.3 Blackboard or double-struck: *bb* and *bbit*

```

167 \@@_new_alphabet_config:nnn {bb} {latin}
168 {
169   \@@_set_mathalphabet_latin:nnn {bb} {up,it}{#1}
170 }
171
172 \@@_new_alphabet_config:nnn {bb} {Latin}
173 {
174   \@@_set_mathalphabet_Latin:nnn {bb} {up,it}{#1}
175   \@@_set_mathalphabet_pos:nnnn {bb} {C} {up,it} {#1}
176   \@@_set_mathalphabet_pos:nnnn {bb} {H} {up,it} {#1}
177   \@@_set_mathalphabet_pos:nnnn {bb} {N} {up,it} {#1}
178   \@@_set_mathalphabet_pos:nnnn {bb} {P} {up,it} {#1}
179   \@@_set_mathalphabet_pos:nnnn {bb} {Q} {up,it} {#1}
180   \@@_set_mathalphabet_pos:nnnn {bb} {R} {up,it} {#1}
181   \@@_set_mathalphabet_pos:nnnn {bb} {Z} {up,it} {#1}
182 }
183

```

```

184 \@@_new_alphabet_config:nnn {bb} {num}
185 {
186   \@@_set_mathalphabet_numbers:nnn {bb} {up}{#1}
187 }
188
189 \@@_new_alphabet_config:nnn {bb} {misc}
190 {
191   \@@_set_mathalphabet_pos:nnnn {bb}      {Pi} {up,it} {#1}
192   \@@_set_mathalphabet_pos:nnnn {bb}      {pi} {up,it} {#1}
193   \@@_set_mathalphabet_pos:nnnn {bb}      {Gamma} {up,it} {#1}
194   \@@_set_mathalphabet_pos:nnnn {bb}      {gamma} {up,it} {#1}
195   \@@_set_mathalphabet_pos:nnnn {bb} {summation} {up} {#1}
196 }
197
198 \@@_new_alphabet_config:nnn {bbit} {misc}
199 {
200   \@@_set_mathalphabet_pos:nnnn {bbit} {D} {up,it} {#1}
201   \@@_set_mathalphabet_pos:nnnn {bbit} {d} {up,it} {#1}
202   \@@_set_mathalphabet_pos:nnnn {bbit} {e} {up,it} {#1}
203   \@@_set_mathalphabet_pos:nnnn {bbit} {i} {up,it} {#1}
204   \@@_set_mathalphabet_pos:nnnn {bbit} {j} {up,it} {#1}
205 }

```

#### 20.4 *Script and caligraphic: scr and cal*

```

206 \@@_new_alphabet_config:nnn {scr} {Latin}
207 {
208   \@@_set_mathalphabet_Latin:nnn {scr} {up,it}{#1}
209   \@@_set_mathalphabet_pos:nnnn {scr} {B}{up,it}{#1}
210   \@@_set_mathalphabet_pos:nnnn {scr} {E}{up,it}{#1}
211   \@@_set_mathalphabet_pos:nnnn {scr} {F}{up,it}{#1}
212   \@@_set_mathalphabet_pos:nnnn {scr} {H}{up,it}{#1}
213   \@@_set_mathalphabet_pos:nnnn {scr} {I}{up,it}{#1}
214   \@@_set_mathalphabet_pos:nnnn {scr} {L}{up,it}{#1}
215   \@@_set_mathalphabet_pos:nnnn {scr} {M}{up,it}{#1}
216   \@@_set_mathalphabet_pos:nnnn {scr} {R}{up,it}{#1}
217 }
218
219 \@@_new_alphabet_config:nnn {scr} {latin}
220 {
221   \@@_set_mathalphabet_latin:nnn {scr} {up,it}{#1}
222   \@@_set_mathalphabet_pos:nnnn {scr} {e}{up,it}{#1}
223   \@@_set_mathalphabet_pos:nnnn {scr} {g}{up,it}{#1}
224   \@@_set_mathalphabet_pos:nnnn {scr} {o}{up,it}{#1}
225 }

```

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

```

226 \@@_new_alphabet_config:nnn {cal} {Latin}
227 {
228   \@@_set_mathalphabet_Latin:nnn {cal} {up,it}{#1}

```

```

229 \@@_set_mathalphabet_pos:nnnn {cal} {B}{up,it}{#1}
230 \@@_set_mathalphabet_pos:nnnn {cal} {E}{up,it}{#1}
231 \@@_set_mathalphabet_pos:nnnn {cal} {F}{up,it}{#1}
232 \@@_set_mathalphabet_pos:nnnn {cal} {H}{up,it}{#1}
233 \@@_set_mathalphabet_pos:nnnn {cal} {I}{up,it}{#1}
234 \@@_set_mathalphabet_pos:nnnn {cal} {L}{up,it}{#1}
235 \@@_set_mathalphabet_pos:nnnn {cal} {M}{up,it}{#1}
236 \@@_set_mathalphabet_pos:nnnn {cal} {R}{up,it}{#1}
237 }

```

## 20.5 *Fraktur or fraktur or blackletter: frak*

```

238 \@@_new_alphabet_config:nnn {frak} {Latin}
239 {
240 \@@_set_mathalphabet_Latin:nnn {frak} {up,it}{#1}
241 \@@_set_mathalphabet_pos:nnnn {frak} {C}{up,it}{#1}
242 \@@_set_mathalphabet_pos:nnnn {frak} {H}{up,it}{#1}
243 \@@_set_mathalphabet_pos:nnnn {frak} {I}{up,it}{#1}
244 \@@_set_mathalphabet_pos:nnnn {frak} {R}{up,it}{#1}
245 \@@_set_mathalphabet_pos:nnnn {frak} {Z}{up,it}{#1}
246 }
247 \@@_new_alphabet_config:nnn {frak} {latin}
248 {
249 \@@_set_mathalphabet_latin:nnn {frak} {up,it}{#1}
250 }

```

## 20.6 *Sans serif upright: sfup*

```

251 \@@_new_alphabet_config:nnn {sfup} {num}
252 {
253 \@@_set_mathalphabet_numbers:nnn {sf} {up}{#1}
254 \@@_set_mathalphabet_numbers:nnn {sfup} {up}{#1}
255 }
256 \@@_new_alphabet_config:nnn {sfup} {Latin}
257 {
258 \bool_if:NTF \g_@@_sfliteral_bool
259 {
260 \@@_set_normal_Latin:nn {sfup} {#1}
261 \@@_set_mathalphabet_Latin:nnn {sf} {up}{#1}
262 }
263 {
264 \bool_if:NT \g_@@_upsans_bool
265 {
266 \@@_set_normal_Latin:nn {sfup,sfit} {#1}
267 \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
268 }
269 }
270 \@@_set_mathalphabet_Latin:nnn {sfup} {up,it}{#1}
271 }
272 \@@_new_alphabet_config:nnn {sfup} {latin}
273 {

```

```

274 \bool_if:NTF \g_@@_sfliteral_bool
275 {
276   \@@_set_normal_latin:nn {sfup} {#1}
277   \@@_set_mathalphabet_latin:nnn {sf} {up}{#1}
278 }
279 {
280   \bool_if:NT \g_@@_upsans_bool
281   {
282     \@@_set_normal_latin:nn {sfup,sfit} {#1}
283     \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
284   }
285 }
286 \@@_set_mathalphabet_latin:nnn {sfup} {up,it}{#1}
287 }

```

## 20.7 *Sans serif italic: sfit*

```

288 \@@_new_alphabet_config:nnn {sfit} {Latin}
289 {
290   \bool_if:NTF \g_@@_sfliteral_bool
291   {
292     \@@_set_normal_Latin:nn {sfit} {#1}
293     \@@_set_mathalphabet_Latin:nnn {sf} {it}{#1}
294   }
295   {
296     \bool_if:NF \g_@@_upsans_bool
297     {
298       \@@_set_normal_Latin:nn {sfup,sfit} {#1}
299       \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
300     }
301   }
302   \@@_set_mathalphabet_Latin:nnn {sfit} {up,it}{#1}
303 }
304 \@@_new_alphabet_config:nnn {sfit} {latin}
305 {
306   \bool_if:NTF \g_@@_sfliteral_bool
307   {
308     \@@_set_normal_latin:nn {sfit} {#1}
309     \@@_set_mathalphabet_latin:nnn {sf} {it}{#1}
310   }
311   {
312     \bool_if:NF \g_@@_upsans_bool
313     {
314       \@@_set_normal_latin:nn {sfup,sfit} {#1}
315       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
316     }
317   }
318   \@@_set_mathalphabet_latin:nnn {sfit} {up,it}{#1}
319 }

```

## 20.8 *Typewriter or monospaced: tt*

```
320 \@@_new_alphabet_config:nnn {tt} {num}
321 {
322   \@@_set_mathalphabet_numbers:nnn {tt} {up}{#1}
323 }
324 \@@_new_alphabet_config:nnn {tt} {Latin}
325 {
326   \@@_set_mathalphabet_Latin:nnn {tt} {up,it}{#1}
327 }
328 \@@_new_alphabet_config:nnn {tt} {latin}
329 {
330   \@@_set_mathalphabet_latin:nnn {tt} {up,it}{#1}
331 }
```

## 20.9 *Bold Italic: bfit*

```
332 \@@_new_alphabet_config:nnn {bfit} {Latin}
333 {
334   \bool_if:NF \g_@@_bfupLatin_bool
335   {
336     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
337   }
338   \@@_set_mathalphabet_Latin:nnn {bfit} {up,it}{#1}
339   \bool_if:NTF \g_@@_bfliteral_bool
340   {
341     \@@_set_normal_Latin:nn {bfit} {#1}
342     \@@_set_mathalphabet_Latin:nnn {bf} {it}{#1}
343   }
344   {
345     \bool_if:NF \g_@@_bfupLatin_bool
346     {
347       \@@_set_normal_Latin:nn {bfup,bfit} {#1}
348       \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
349     }
350   }
351 }
352
353 \@@_new_alphabet_config:nnn {bfit} {latin}
354 {
355   \bool_if:NF \g_@@_bfuplatin_bool
356   {
357     \@@_set_normal_latin:nn {bfup,bfit} {#1}
358   }
359   \@@_set_mathalphabet_latin:nnn {bfit} {up,it}{#1}
360   \bool_if:NTF \g_@@_bfliteral_bool
361   {
362     \@@_set_normal_latin:nn {bfit} {#1}
363     \@@_set_mathalphabet_latin:nnn {bf} {it}{#1}
364   }
```

```

365 {
366   \bool_if:NF \g_@@_bfuplatin_bool
367   {
368     \@@_set_normal_latin:nn {bfup,bfit} {#1}
369     \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
370   }
371 }
372 }
373
374 \@@_new_alphabet_config:nnn {bfit} {Greek}
375 {
376   \@@_set_mathalphabet_Greek:nnn {bfit} {up,it}{#1}
377   \bool_if:NTF \g_@@_bfliteral_bool
378   {
379     \@@_set_normal_Greek:nn {bfit}{#1}
380     \@@_set_mathalphabet_Greek:nnn {bf} {it}{#1}
381   }
382   {
383     \bool_if:NF \g_@@_bfupGreek_bool
384     {
385       \@@_set_normal_Greek:nn {bfup,bfit}{#1}
386       \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
387     }
388   }
389 }
390
391 \@@_new_alphabet_config:nnn {bfit} {greek}
392 {
393   \@@_set_mathalphabet_greek:nnn {bfit} {up,it} {#1}
394   \bool_if:NTF \g_@@_bfliteral_bool
395   {
396     \@@_set_normal_greek:nn {bfit} {#1}
397     \@@_set_mathalphabet_greek:nnn {bf} {it} {#1}
398   }
399   {
400     \bool_if:NF \g_@@_bfupgreek_bool
401     {
402       \@@_set_normal_greek:nn {bfit,bfup} {#1}
403       \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
404     }
405   }
406 }
407
408 \@@_new_alphabet_config:nnn {bfit} {misc}
409 {
410   \bool_if:NTF \g_@@_literal_Nabla_bool
411   { \@@_set_normal_char:nnn {Nabla}{bfit}{#1} }
412   {
413     \bool_if:NF \g_@@_upNabla_bool

```

```

414     { \@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1} }
415   }
416   \bool_if:NTF \g_@@_literal_partial_bool
417   { \@_set_normal_char:nnn {partial}{bfit}{#1} }
418   {
419     \bool_if:NF \g_@@_uppartial_bool
420     { \@_set_normal_char:nnn {partial}{bfup,bfit}{#1} }
421   }
422   \@_set_mathalphabet_pos:nnnn {bfit} {partial} {up,it}{#1}
423   \@_set_mathalphabet_pos:nnnn {bfit} {Nabla} {up,it}{#1}
424   \bool_if:NTF \g_@@_literal_partial_bool
425   {
426     \@_set_mathalphabet_pos:nnnn {bf} {partial} {it}{#1}
427   }
428   {
429     \bool_if:NF \g_@@_uppartial_bool
430     {
431       \@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
432     }
433   }
434   \bool_if:NTF \g_@@_literal_Nabla_bool
435   {
436     \@_set_mathalphabet_pos:nnnn {bf} {Nabla} {it}{#1}
437   }
438   {
439     \bool_if:NF \g_@@_upNabla_bool
440     {
441       \@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
442     }
443   }
444 }

```

## 20.10 Bold Upright: *bfup*

```

445 \@_new_alphabet_config:nnn {bfup} {num}
446 {
447   \@_set_mathalphabet_numbers:nnn {bf} {up}{#1}
448   \@_set_mathalphabet_numbers:nnn {bfup} {up}{#1}
449 }
450
451 \@_new_alphabet_config:nnn {bfup} {Latin}
452 {
453   \bool_if:NT \g_@@_bfupLatin_bool
454   {
455     \@_set_normal_Latin:nn {bfup,bfit} {#1}
456   }
457   \@_set_mathalphabet_Latin:nnn {bfup} {up,it}{#1}
458   \bool_if:NTF \g_@@_bfliteral_bool
459   {
460     \@_set_normal_Latin:nn {bfup} {#1}

```

```

461 \@@_set_mathalphabet_Latin:nnn {bf} {up}{#1}
462 }
463 {
464 \bool_if:NT \g_@@_bfupLatin_bool
465 {
466 \@@_set_normal_Latin:nn {bfup,bfit} {#1}
467 \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
468 }
469 }
470 }
471
472 \@@_new_alphabet_config:nnn {bfup} {latin}
473 {
474 \bool_if:NT \g_@@_bfuplatin_bool
475 {
476 \@@_set_normal_latin:nn {bfup,bfit} {#1}
477 }
478 \@@_set_mathalphabet_latin:nnn {bfup} {up,it}{#1}
479 \bool_if:NTF \g_@@_bfliteral_bool
480 {
481 \@@_set_normal_latin:nn {bfup} {#1}
482 \@@_set_mathalphabet_latin:nnn {bf} {up}{#1}
483 }
484 {
485 \bool_if:NT \g_@@_bfuplatin_bool
486 {
487 \@@_set_normal_latin:nn {bfup,bfit} {#1}
488 \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
489 }
490 }
491 }
492 \@@_new_alphabet_config:nnn {bfup} {Greek}
493 {
494 \@@_set_mathalphabet_Greek:nnn {bfup} {up,it}{#1}
495 \bool_if:NTF \g_@@_bfliteral_bool
496 {
497 \@@_set_normal_Greek:nn {bfup}{#1}
498 \@@_set_mathalphabet_Greek:nnn {bf} {up}{#1}
499 }
500 {
501 \bool_if:NT \g_@@_bfupGreek_bool
502 {
503 \@@_set_normal_Greek:nn {bfup,bfit}{#1}
504 \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
505 }
506 }
507 }
508
509 \@@_new_alphabet_config:nnn {bfup} {greek}

```

```

510 {
511   \@@_set_mathalphabet_greek:nnn {bfup} {up,it} {#1}
512   \bool_if:NTF \g_@@_bfliteral_bool
513   {
514     \@@_set_normal_greek:nn {bfup} {#1}
515     \@@_set_mathalphabet_greek:nnn {bf} {up} {#1}
516   }
517   {
518     \bool_if:NT \g_@@_bfupgreek_bool
519     {
520       \@@_set_normal_greek:nn {bfup,bfit} {#1}
521       \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
522     }
523   }
524 }
525
526 \@@_new_alphabet_config:nnn {bfup} {misc}
527 {
528   \bool_if:NTF \g_@@_literal_Nabla_bool
529   {
530     \@@_set_normal_char:nnn {Nabla}{bfup}{#1}
531   }
532   {
533     \bool_if:NT \g_@@_upNabla_bool
534     {
535       \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1}
536     }
537   }
538   \bool_if:NTF \g_@@_literal_partial_bool
539   {
540     \@@_set_normal_char:nnn {partial}{bfup}{#1}
541   }
542   {
543     \bool_if:NT \g_@@_uppartial_bool
544     {
545       \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1}
546     }
547   }
548   \@@_set_mathalphabet_pos:nnnn {bfup} {partial} {up,it}{#1}
549   \@@_set_mathalphabet_pos:nnnn {bfup} {Nabla} {up,it}{#1}
550   \@@_set_mathalphabet_pos:nnnn {bfup} {digamma} {up}{#1}
551   \@@_set_mathalphabet_pos:nnnn {bfup} {Digamma} {up}{#1}
552   \@@_set_mathalphabet_pos:nnnn {bf} {digamma} {up}{#1}
553   \@@_set_mathalphabet_pos:nnnn {bf} {Digamma} {up}{#1}
554   \bool_if:NTF \g_@@_literal_partial_bool
555   {
556     \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up}{#1}
557   }
558   {

```

```

559 \bool_if:NT \g_@@_uppartial_bool
560 {
561   \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
562 }
563 }
564 \bool_if:NTF \g_@@_literal_Nabla_bool
565 {
566   \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up}{#1}
567 }
568 {
569   \bool_if:NT \g_@@_upNabla_bool
570   {
571     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
572   }
573 }
574 }

```

### 20.11 *Bold fractur or fraktur or blackletter: bffrak*

```

575 \@@_new_alphabet_config:nnn {bffrak} {Latin}
576 {
577   \@@_set_mathalphabet_Latin:nnn {bffrak} {up,it}{#1}
578 }
579
580 \@@_new_alphabet_config:nnn {bffrak} {latin}
581 {
582   \@@_set_mathalphabet_latin:nnn {bffrak} {up,it}{#1}
583 }

```

### 20.12 *Bold script or calligraphic: bfscr*

```

584 \@@_new_alphabet_config:nnn {bfscr} {Latin}
585 {
586   \@@_set_mathalphabet_Latin:nnn {bfscr} {up,it}{#1}
587 }
588 \@@_new_alphabet_config:nnn {bfscr} {latin}
589 {
590   \@@_set_mathalphabet_latin:nnn {bfscr} {up,it}{#1}
591 }
592 \@@_new_alphabet_config:nnn {bfcal} {Latin}
593 {
594   \@@_set_mathalphabet_Latin:nnn {bfcal} {up,it}{#1}
595 }

```

### 20.13 *Bold upright sans serif: bfsfup*

```

596 \@@_new_alphabet_config:nnn {bfsfup} {num}
597 {
598   \@@_set_mathalphabet_numbers:nnn {bfsf} {up}{#1}
599   \@@_set_mathalphabet_numbers:nnn {bfsfup} {up}{#1}
600 }
601 \@@_new_alphabet_config:nnn {bfsfup} {Latin}

```

```

602 {
603   \bool_if:NTF \g_@@_sfliteral_bool
604   {
605     \@@_set_normal_Latin:nn {bfsfup} {#1}
606     \@@_set_mathalphabet_Latin:nnn {bfsf} {up}{#1}
607   }
608   {
609     \bool_if:NT \g_@@_upsans_bool
610     {
611       \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
612       \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
613     }
614   }
615   \@@_set_mathalphabet_Latin:nnn {bfsfup} {up,it}{#1}
616 }
617
618 \@@_new_alphabet_config:nnn {bfsfup} {latin}
619 {
620   \bool_if:NTF \g_@@_sfliteral_bool
621   {
622     \@@_set_normal_latin:nn {bfsfup} {#1}
623     \@@_set_mathalphabet_latin:nnn {bfsf} {up}{#1}
624   }
625   {
626     \bool_if:NT \g_@@_upsans_bool
627     {
628       \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
629       \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
630     }
631   }
632   \@@_set_mathalphabet_latin:nnn {bfsfup} {up,it}{#1}
633 }
634
635 \@@_new_alphabet_config:nnn {bfsfup} {Greek}
636 {
637   \bool_if:NTF \g_@@_sfliteral_bool
638   {
639     \@@_set_normal_Greek:nn {bfsfup}{#1}
640     \@@_set_mathalphabet_Greek:nnn {bfsf} {up}{#1}
641   }
642   {
643     \bool_if:NT \g_@@_upsans_bool
644     {
645       \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
646       \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
647     }
648   }
649   \@@_set_mathalphabet_Greek:nnn {bfsfup} {up,it}{#1}
650 }

```

```

651
652 \@@_new_alphabet_config:nnn {bfsfup} {greek}
653 {
654   \bool_if:NTF \g_@@_sfliteral_bool
655   {
656     \@@_set_normal_greek:nn {bfsfup} {#1}
657     \@@_set_mathalphabet_greek:nnn {bfsf} {up} {#1}
658   }
659   {
660     \bool_if:NT \g_@@_upsans_bool
661     {
662       \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
663       \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
664     }
665   }
666   \@@_set_mathalphabet_greek:nnn {bfsfup} {up,it} {#1}
667 }
668 \@@_new_alphabet_config:nnn {bfsfup} {misc}
669 {
670   \bool_if:NTF \g_@@_literal_Nabla_bool
671   {
672     \@@_set_normal_char:nnn {Nabla}{bfsfup}{#1}
673   }
674   {
675     \bool_if:NT \g_@@_upNabla_bool
676     {
677       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
678     }
679   }
680   \bool_if:NTF \g_@@_literal_partial_bool
681   {
682     \@@_set_normal_char:nnn {partial}{bfsfup}{#1}
683   }
684   {
685     \bool_if:NT \g_@@_uppartial_bool
686     {
687       \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
688     }
689   }
690   \@@_set_mathalphabet_pos:nnnn {bfsfup} {partial} {up,it}{#1}
691   \@@_set_mathalphabet_pos:nnnn {bfsfup} {Nabla} {up,it}{#1}
692   \bool_if:NTF \g_@@_literal_partial_bool
693   {
694     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up}{#1}
695   }
696   {
697     \bool_if:NT \g_@@_uppartial_bool
698     {
699       \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}

```

```

700     }
701   }
702   \bool_if:NTF \g_@@_literal_Nabla_bool
703   {
704     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla}  {up}{#1}
705   }
706   {
707     \bool_if:NT \g_@@_upNabla_bool
708     {
709       \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla}  {up,it}{#1}
710     }
711   }
712 }

```

## 20.14 *Bold italic sans serif: bfsfit*

```

713 \@@_new_alphabet_config:nnn {bfsfit} {Latin}
714 {
715   \bool_if:NTF \g_@@_sfliteral_bool
716   {
717     \@@_set_normal_Latin:nn {bfsfit} {#1}
718     \@@_set_mathalphabet_Latin:nnn {bfsf} {it}{#1}
719   }
720   {
721     \bool_if:NF \g_@@_upsans_bool
722     {
723       \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
724       \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
725     }
726   }
727   \@@_set_mathalphabet_Latin:nnn {bfsfit} {up,it}{#1}
728 }
729
730 \@@_new_alphabet_config:nnn {bfsfit} {latin}
731 {
732   \bool_if:NTF \g_@@_sfliteral_bool
733   {
734     \@@_set_normal_latin:nn {bfsfit} {#1}
735     \@@_set_mathalphabet_latin:nnn {bfsf} {it}{#1}
736   }
737   {
738     \bool_if:NF \g_@@_upsans_bool
739     {
740       \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
741       \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
742     }
743   }
744   \@@_set_mathalphabet_latin:nnn {bfsfit} {up,it}{#1}
745 }
746

```

```

747 \@@_new_alphabet_config:nnn {bfsfit} {Greek}
748 {
749   \bool_if:NTF \g_@@_sfliteral_bool
750   {
751     \@@_set_normal_Greek:nn {bfsfit}{#1}
752     \@@_set_mathalphabet_Greek:nnn {bfsf} {it}{#1}
753   }
754   {
755     \bool_if:NF \g_@@_upsans_bool
756     {
757       \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
758       \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
759     }
760   }
761   \@@_set_mathalphabet_Greek:nnn {bfsfit} {up,it}{#1}
762 }
763
764 \@@_new_alphabet_config:nnn {bfsfit} {greek}
765 {
766   \bool_if:NTF \g_@@_sfliteral_bool
767   {
768     \@@_set_normal_greek:nn {bfsfit} {#1}
769     \@@_set_mathalphabet_greek:nnn {bfsf} {it} {#1}
770   }
771   {
772     \bool_if:NF \g_@@_upsans_bool
773     {
774       \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
775       \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
776     }
777   }
778   \@@_set_mathalphabet_greek:nnn {bfsfit} {up,it} {#1}
779 }
780
781 \@@_new_alphabet_config:nnn {bfsfit} {misc}
782 {
783   \bool_if:NTF \g_@@_literal_Nabla_bool
784   {
785     \@@_set_normal_char:nnn {Nabla}{bfsfit}{#1}
786   }
787   {
788     \bool_if:NF \g_@@_upNabla_bool
789     {
790       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
791     }
792   }
793   \bool_if:NTF \g_@@_literal_partial_bool
794   {
795     \@@_set_normal_char:nnn {partial}{bfsfit}{#1}

```

```

796 }
797 {
798   \bool_if:NF \g_@@_uppartial_bool
799   {
800     \@@_set_mathalphabet_pos:nnn {partial}{bfsfup,bfsfit}{#1}
801   }
802 }
803 \@@_set_mathalphabet_pos:nxxx {bfsfit} {partial} {up,it}{#1}
804 \@@_set_mathalphabet_pos:nxxx {bfsfit} {Nabla} {up,it}{#1}
805 \bool_if:NTF \g_@@_literal_partial_bool
806 {
807   \@@_set_mathalphabet_pos:nxxx {bfsf} {partial} {it}{#1}
808 }
809 {
810   \bool_if:NF \g_@@_uppartial_bool
811   {
812     \@@_set_mathalphabet_pos:nxxx {bfsf} {partial} {up,it}{#1}
813   }
814 }
815 \bool_if:NTF \g_@@_literal_Nabla_bool
816 {
817   \@@_set_mathalphabet_pos:nxxx {bfsf} {Nabla} {it}{#1}
818 }
819 {
820   \bool_if:NF \g_@@_upNabla_bool
821   {
822     \@@_set_mathalphabet_pos:nxxx {bfsf} {Nabla} {up,it}{#1}
823   }
824 }
825 }
826 </package>

```

## Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
$\backslash$ \$	48
$\backslash$ &	49
$\backslash$ '	47, 236, 242, 260
$\backslash$ *	217, 237
$\backslash$ -	57, 60, 216, 239, 240, 242
$\backslash$ .	238, 270
$\backslash$ /	243, 273
$\backslash$ :	72, 220, 244
$\backslash$ :::	7
$\backslash$ ::N	7
$\backslash$ ::x_unbraced	7
$\backslash$ <	277
$\backslash$ =	58, 61
$\backslash$ >	278
$\backslash$ @_accent:nnn	46, 117, 124
$\backslash$ @_alphabet_config:nnn	69, 200, 206, 218
$\backslash$ @_arg_i_before_egroup:n	2, 6
$\backslash$ @_assign_delcode:n	266, 276, 279–305
$\backslash$ @_assign_delcode:nn	29, 77, 255, 266, 273–275, 277, 278
$\backslash$ @_assign_delcode_noparse:nn	77, 255, 263
$\backslash$ @_assign_delcode_parse:nn	29, 259
$\backslash$ @_backprime_double_mchar	48, 240, 256
$\backslash$ @_backprime_single_mchar	168, 174, 180, 185, 190, 195, 239, 255
$\backslash$ @_backprime_triple_mchar	53, 241, 257
$\backslash$ @_char_gmake_mathactive:N	7, 45, 50, 68
$\backslash$ @_char_gmake_mathactive:n	50, 71, 253
$\backslash$ @_check_and_fix:NNnnn	32, 38, 71, 251
$\backslash$ @_check_and_fix:NNnnnn	2, 34, 152
$\backslash$ @_check_and_fix_luatex:NNnnn	36, 114
$\backslash$ @_check_and_fix_luatex:cNnnn	36
$\backslash$ @_copy_fontparam:nnn	58, 119–130, 133, 153–157
$\backslash$ @_declare_math_sizes:	16, 80
$\backslash$ @_default_mathalph:nnn	78, 84–100
$\backslash$ @_define_active_frac:Nw	65, 85–103
$\backslash$ @_define_math_chars:	11, 12
$\backslash$ @_define_prime_chars:	248, 269
$\backslash$ @_define_prime_commands:	248, 249
$\backslash$ @_delimiter:Nnn	42, 95, 103, 110, 112
$\backslash$ @_error:n	2
$\backslash$ @_font_dimen:Nnnnn	12, 78
$\backslash$ @_font_param:n	70, 103, 110, 113, 122, 135, 145
$\backslash$ @_font_param:nn	66, 101, 102, 108, 109, 111, 112, 114–121, 123, 124, 126–129, 133, 134, 136–139, 141–144
$\backslash$ @_font_param:nnn	62, 104–107, 125, 130–132
$\backslash$ @_font_param:nnnnn	47, 64, 68, 72, 140
$\backslash$ @_font_param_aux:NNN	73
$\backslash$ @_font_param_aux:NNnnnn	73
$\backslash$ @_font_param_aux:ccc	58
$\backslash$ @_font_param_aux:ccnnnn	50
$\backslash$ @_fontdimen_to_percent:nn	146, 157, 160
$\backslash$ @_fontdimen_to_scale:nn	94, 95, 146
$\backslash$ @_fontspec_select_font:	12, 17, 171
$\backslash$ @_get_fontparam:nn	60, 66
$\backslash$ @_glyph_if_exist:Nn	10
$\backslash$ @_glyph_if_exist:NnT	182
$\backslash$ @_glyph_if_exist:NnTF	10, 22, 27, 32, 47, 52, 203
$\backslash$ @_if_alphabet_exists:nn	73
$\backslash$ @_if_alphabet_exists:nnTF	174
$\backslash$ @_if_char_spec:nNNT	92, 209, 225, 230, 247, 261
$\backslash$ @_if_mathalph_decl:n	59
$\backslash$ @_if_mathalph_decl:nTF	38, 59
$\backslash$ @_init:	6, 58
$\backslash$ @_init_alphabet:n	21, 39, 75, 160
$\backslash$ @_input_math_symbol_table:	10, 24, 38
$\backslash$ @_int_if_slot_in_range:nnT	105, 118
$\backslash$ @_keys_choices:nn	2, 26, 31, 36, 41, 46, 51, 56, 90, 113, 140, 146, 158, 170, 175, 181, 208
$\backslash$ @_keys_choices_aux:nnn	4, 14
$\backslash$ @_keys_choices_fn:nn	4, 10
$\backslash$ @_load_lm_if_necessary:	48, 49
$\backslash$ @_log:n	5, 10, 136, 144, 167
$\backslash$ @_log:nx	6, 41, 199, 205
$\backslash$ @_make_mathactive:nNN	30, 78, 235–243, 245

$\backslash$ @@_make_mathactive_noparse:nNN . . .	$\backslash$ @@_remap_symbol_noparse:nnn . . .
. . . . . 78, 248, 250	. . . . . 74, <u>214</u>
$\backslash$ @@_make_mathactive_parse:nNN . . .	$\backslash$ @@_remap_symbol_parse:nnn . . .
. . . . . 30, 245	. . . . . 26, <u>214</u> , 223
$\backslash$ @@_map_char_noparse:nn . . .	$\backslash$ @@_remap_symbols: . . . . .
. . . . . 76, 147, <u>226</u>	. . . . . 40, <u>214</u>
$\backslash$ @@_map_char_parse:nn . . . . .	$\backslash$ @@_resolve_greek: . . . . .
. . . . . 28, <u>226</u>	. . . . . <u>2</u>
$\backslash$ @@_map_char_single:nn . . . . .	$\backslash$ @@_scan_backdprime: . . . . .
. . . . . 28, 76, 147, <u>226</u> , 235, 241	. . . . . 170, 215
$\backslash$ @@_map_char_single:nnn . . . . .	$\backslash$ @@_scan_backprime: . . . . .
. . . . . <u>233</u> , 255, 281–286, 301	. . . . . 164, 205
$\backslash$ @@_map_chars_range:nnn . . . . .	$\backslash$ @@_scan_backtrprime: . . . . .
. . . . . 238, 245	. . . . . 176, 227
$\backslash$ @@_map_chars_range:nnnn . . . . .	$\backslash$ @@_scan_dprime: . . . . .
. . . . . <u>238</u> , 264, 272, 280, 300, 308	. . . . . 67, 117
$\backslash$ @@_mathmap_noparse:nnn . . . . .	$\backslash$ @@_scan_prime: . . . . .
. . . . . 73, 146, <u>310</u> , 324	. . . . . 61, 111
$\backslash$ @@_mathmap_parse:Nnn . . . . .	$\backslash$ @@_scan_qprime: . . . . .
. . . . . 25	. . . . . 79, 141
$\backslash$ @@_mathmap_parse:nnn . . . . .	$\backslash$ @@_scan_ssctipt: . . . . .
. . . . . <u>320</u>	. . . . . 15, 53, 95, 99
$\backslash$ @@_mathstyle_scale:Nnn . . . . .	$\backslash$ @@_scan_ssctipt:TF . . . . .
. . . . . 39, 47, <u>154</u> , 210, 223	. . . . . 97, 105
$\backslash$ @@_maybe_init_alphabet:n . . . . .	$\backslash$ @@_scan_ssctipt_aux:nnN . . . . .
. . . . . 27, 75, 139–141, 160, 178, 184	. . . . . 110, 114
$\backslash$ @@_nbackprimes_select:nn . . . . .	$\backslash$ @@_scan_sup_backdprime: . . . . .
. . . . . 41, 239	. . . . . 187, 278
$\backslash$ @@_new_alphabet_config:nnn . . . . .	$\backslash$ @@_scan_sup_backprime: . . . . .
. . . . . 2, 8, 19, 36, 47, 55,	. . . . . 182, 276, 277
61, 88, 97, 118, 130, 142, 167, 172,	$\backslash$ @@_scan_sup_backtrprime: . . . . .
184, 189, 198, 206, 219, 226, 238,	. . . . . 192, 279
247, 251, 256, 272, 288, 304, 320,	$\backslash$ @@_scan_sup_dprime: . . . . .
324, 328, 332, 353, 374, 391, 408,	. . . . . 90, 273
445, 451, 472, 492, 509, 526, 575,	$\backslash$ @@_scan_sup_prime: . . . . .
580, 584, 588, 592, 596, 601, 618,	. . . . . 85, 271, 272
635, 652, 668, 713, 730, 747, 764, 781	$\backslash$ @@_scan_sup_qprime: . . . . .
$\backslash$ @@_new_cramped_style:N . . . . .	. . . . . 100, 275
. . . . . <u>2</u> , 8–11	$\backslash$ @@_scan_sup_trprime: . . . . .
$\backslash$ @@_new_named_range:n . . . . .	. . . . . 95, 274
. . . . . 44, 54	$\backslash$ @@_scan_trprime: . . . . .
$\backslash$ @@_newnot:N . . . . .	. . . . . 73, 129
. . . . . 159, 181	$\backslash$ @@_scanbackprime_collect:N . . . . .
$\backslash$ @@_nprimes:Nn . . . . .	. . . . . 168, 174, 180, 185, 190, 195,
. . . . . 8, 24, 29, 34, 38, 49, 54, 58	. . . . . 197, 202, 207, 212, 218, 224, 230, 236
$\backslash$ @@_nprimes_select:nn . . . . .	$\backslash$ @@_scanprime_collect:N . . . . .
. . . . . 16, 153	. . . . . 65,
$\backslash$ @@_numrange_parse:nwT . . . . .	. . . . . 71, 77, 83, 88, 93, 98, 103, 105, 109,
. . . . . 119, 120	. . . . . 112, 115, 120, 126, 132, 138, 144, 150
$\backslash$ @@_oldnot: . . . . .	$\backslash$ @@_script_style_size:n . . . . .
. . . . . 176, 180	. . . . . 83, 86, 87, <u>99</u>
$\backslash$ @@_prepare_mathstyle:n . . . . .	$\backslash$ @@_set_big_operator:nnn . . . . .
. . . . . <u>18</u> , 107	. . . . . 45, <u>69</u>
$\backslash$ @@_prime_double_mchar . . . . .	$\backslash$ @@_set_delcode:nnn . . . . .
. . . . . 23, 236, 252	. . . . . <u>34</u> , 92, 100, 108, 257, 270
$\backslash$ @@_prime_quad_mchar . . . . .	$\backslash$ @@_set_math_accent:Nnnn . . . . .
. . . . . 33, 238, 254	. . . . . 50, 52, 54, 56, <u>114</u>
$\backslash$ @@_prime_single_mchar . . . . .	$\backslash$ @@_set_math_close:nnn . . . . .
. . . . . 65,	. . . . . 47, <u>98</u>
71, 77, 83, 88, 93, 98, 103, 235, 251	$\backslash$ @@_set_math_fence:nnnn . . . . .
$\backslash$ @@_prime_triple_mchar . . . . .	. . . . . 48, <u>105</u>
. . . . . 28, 237, 253	$\backslash$ @@_set_math_open:nnn . . . . .
$\backslash$ @@_print_indent:n . . . . .	. . . . . 46, <u>83</u>
. . . . . 44, 46	$\backslash$ @@_set_math_overunder:Nnnn . . . . .
$\backslash$ @@_process_symbol_noparse:nnn . . . . .	. . . . . 58, 60, <u>119</u>
. . . . . 72, <u>203</u>	$\backslash$ @@_set_mathalph_range:nnnn . . . . .
$\backslash$ @@_process_symbol_parse:nnn . . . . .	. . . . . <u>332</u>
. . . . . 24, <u>203</u>	$\backslash$ @@_set_mathalph_range:nnnnn . . . . .
$\backslash$ @@_radical:nn . . . . .	. . . . . 337, 353, 358, 364, 372, 380
. . . . . <u>38</u> , 88	$\backslash$ @@_set_mathalphabet_Greek:nnn . . . . .
$\backslash$ @@_radical_vgap:N . . . . .	. . . . . 42–44,
. . . . . 298	. . . . . 127, 299, 368, 376, 380, 386, 494,
$\backslash$ @@_redefine_radical: . . . . .	. . . . . 498, 504, 640, 646, 649, 752, 758, 761
. . . . . 23, 24	$\backslash$ @@_set_mathalphabet_Latin:nnn . . . . .
$\backslash$ @@_remap_symbol:nnn . . . . .	. . . . . 14–
. . . . . 26, 74, 216, 217, 220	. . . . . 16, 94, 174, 208, 228, 240, 261, 263,
	. . . . . 267, 270, 293, 299, 302, 326, 338,
	. . . . . 342, 348, 355, 457, 461, 467, 577,
	. . . . . 586, 594, 606, 612, 615, 718, 724, 727

<code>\@@_set_mathalphabet_char:Nnn</code>	. 25, <a href="#">310</a>	<code>\@@_set_normal_numbers:nn</code>	..... 4, 305
<code>\@@_set_mathalphabet_char:nnn</code>	....	<code>\@@_setmathfont:nn</code>	..... <a href="#">2</a> , 5
.....	73, 146, 329, 335	<code>\@@_setmathfontface:Nnn</code>	..... <a href="#">2</a> , 9
<code>\@@_set_mathalphabet_char:nnnn</code>	....	<code>\@@_setup_active_frac:</code>	..... 82, 106
.....	<a href="#">327</a> , 347, 365, 373, 381–386	<code>\@@_setup_active_subscript:nn</code>	41, 57–93
<code>\@@_set_mathalphabet_greek:nnn</code>	....	<code>\@@_setup_active_superscript:nn</code>	...
.....	56–58,	.....	3, 19–40
139, 279, 376, 393, 397, 403, 511,		<code>\@@_setup_alphabets:</code>	..... 44, <a href="#">132</a>
515, 521, 657, 663, 666, 769, 775, 778		<code>\@@_setup_delcodes:</code>	..... 42, <a href="#">267</a>
<code>\@@_set_mathalphabet_latin:nnn</code>	....	<code>\@@_setup_legacy_fam_three:</code>	... 35, <a href="#">146</a>
. 31–33, 113, 169, 221, 249, 271,		<code>\@@_setup_legacy_fam_two:</code>	.. 34, 108, 111
277, 283, 286, 309, 315, 318, 330,		<code>\@@_setup_math_alphabet:</code>	.... 165, <a href="#">169</a>
359, 360, 363, 369, 478, 482, 488,		<code>\@@_setup_mathactives:</code>	..... 41, <a href="#">233</a>
582, 590, 623, 629, 632, 735, 741, 744		<code>\@@_setup_negations:</code>	..... 46, 182
<code>\@@_set_mathalphabet_numbers:nnn</code>	..	<code>\@@_split_arrow:w</code>	..... 66, 82
.....	5, 186, 253,	<code>\@@_split_slash:w</code>	..... 69, 87
254, 307, 322, 350, 447, 448, 598, 599		<code>\@@_stack_denom_down:N</code>	..... 98
<code>\@@_set_mathalphabet_pos:nnnn</code>	....	<code>\@@_stack_num_up:N</code>	..... 97
....	83–86, 114, 115, 164, 165,	<code>\@@_stack_vgap:N</code>	..... 100
175–181, 191–195, 200–204, 209–		<code>\@@_sub_or_super:n</code>	..... 13, 51, 102
216, 222–224, 229–236, 241–245,		<code>\@@_superscript:n</code>	3, 10, 20, 23, 28, 33,
254, 287–292, 302, 342, 422, 423,		45, 48, 53, 63, 69, 75, 81, 166, 172, 178	
426, 431, 436, 441, 548–553, 556,		<code>\@@_switchto_literal:</code>	..... 8, 15
561, 566, 571, 690, 691, 694, 699,		<code>\@@_switchto_up:</code>	..... 43
704, 709, 803, 804, 807, 812, 817, 822		<code>\@@_symbol_setup:</code>	..... 2, 9
<code>\@@_set_mathchar:NNnn</code>	..... <a href="#">28</a> , 33, 252	<code>\@@_text_slash:</code>	..... 154, 157
<code>\@@_set_mathchar:cNnn</code>	..... <a href="#">28</a> , 75	<code>\@@_tl_map_dbl:nN</code>	..... 10, 15
<code>\@@_set_mathcode:nnn</code>	.....	<code>\@@_tmpa:</code>	..... 72, 73
.....	<a href="#">18</a> , 40–44, 93, 101, 107	<code>\@@_tmpa:w</code>	..... 9, 10, 22
<code>\@@_set_mathcode:nnnn</code>	. <a href="#">18</a> , 227, 231, 316	<code>\@@_to_usv:nn</code>	..... 182, 203,
<code>\@@_set_mathsymbol:nNNn</code>	..... <a href="#">27</a> , 205	235, 236, 245, 246, 329, 330, 339, 340	
<code>\@@_set_normal_Greek:nn</code>	.....	<code>\@@_usv_if_exist:nn</code>	..... 73
.....	38, 40, 122, 125, 295,	<code>\@@_usv_if_exist:nnT</code>	..... 250, 344
379, 385, 497, 503, 639, 645, 751, 757		<code>\@@_warning:n</code>	..... 3, 188
<code>\@@_set_normal_Latin:nn</code>	10, 12, 90, 92,	<code>\@@_warning:nnn</code>	..... 4, 58
259, 260, 266, 292, 298, 336, 341,		<code>\@@_zero_fontparam:n</code>	.. <a href="#">62</a> , 131, 132, 158
347, 455, 460, 466, 605, 611, 717, 723		<code>\@DeclareMathDelimiter</code>	..... 90
<code>\@@_set_normal_char:nnn</code>	.....	<code>\@DeclareMathSizes</code>	..... 81
. 26–28, 65, 70, 75, 80, 102, 108–		<code>\@backslashchar</code>	..... 26, 213
110, 146, 151, 156, 161, <a href="#">248</a> , 411,		<code>\@begindocumenthook</code>	..... 55
414, 417, 420, 530, 535, 540, 545,		<code>\@cdots</code>	..... 63
672, 677, 682, 687, 785, 790, 795, 800		<code>\@depth</code>	..... 317, 324
<code>\@@_set_normal_greek:nn</code>	.....	<code>\@ehd</code>	..... 7, 14, 20
.....	49, 53, 134, 137, 275,	<code>\@fontswitch</code>	..... 16
396, 402, 514, 520, 656, 662, 768, 774		<code>\@height</code>	..... 317, 324
<code>\@@_set_normal_latin:nn</code>	.....	<code>\@ifpackageloaded</code>	..... 27, 56, 139
.....	21, 25, 101, 107,	<code>\@ne</code>	..... 157, 158
267, 276, 282, 308, 314, 357, 362,		<code>\@nil</code>	..... 225, 230, 261
368, 476, 481, 487, 622, 628, 734, 740		<code>\@onlypreamble</code>	..... 11

<code>\@preamblecmds</code> .....	95	366, 383, 400, 413, 419, 429, 439,
<code>\@xDeclareMathDelimiter</code> .....	90	721, 738, 755, 772, 788, 798, 810, 820
<code>\@xxDeclareMathDelimiter</code> .....	89	
<code>\[</code> .....	73	
<code>\</code> .....	9, 43, 46, 62, 63, 70, 74, 80, 83, 86	
<code>\_</code> .....	71	609, 626, 643, 660, 675, 685, 697, 707
<code>\_@_sym:nnn</code> .....	6, 15, 24, 72	
<code>\_@_t1_map_dbl:Nnn</code> .....	17, 19, 24	
<code>\_fontspec_setboldmathrm_hook:nn</code> .	72	
<code>\_fontspec_setmainfont_hook:nn</code> ...	33	
<code>\_fontspec_setmathrm_hook:nn</code> ....	66	
<code>\_fontspec_setmathsf_hook:nn</code> ....	78	
<code>\_fontspec_setmathhtt_hook:nn</code> ....	83	
<code>\_fontspec_setmonofont_hook:nn</code> ...	56	
<code>\_fontspec_setsansfont_hook:nn</code> ...	45	
<code>\`</code> .....	46, 243, 261	
<code>\ </code> .....	151	
<code>\_</code> .....	9, 60, 61, 64, 65, 71, 81, 84, 109	
<b>A</b>		
<code>\addnolimits</code> .....	17	
<code>\addtoversion</code> .....	82	
<code>\advance</code> .....	79, 156, 158, 265, 279	
<code>\alpha@elt</code> .....	83	
<code>\alpha@list</code> .....	83	
<code>\AtBeginDocument</code> .....		
.....	2, 11, 23, 48, 119, 181, 248, 309	
<code>\AtEndOfPackageFile</code> .....		
.....	41, 52, 232, 249, 304, 338	
<code>\awint</code> .....	311	
<b>B</b>		
<code>\B</code> .....	139	
<code>\backprime</code> .....	256	
<code>\backepsilon</code> .....	127	
<code>\backprime</code> .....	255	
<code>\backtrprime</code> .....	257	
<code>\baselineskip</code> .....	78, 79, 95	
<code>\begin</code> .....	108, 110	
<code>\beta</code> .....	89	
<code>\bfdefault</code> .....		
.....	29, 41, 52, 63, 70, 75, 81, 86, 143, 168	
<code>\bgroup</code> .....	5, 74, 83	
<code>\bigcirc</code> .....	112	
<code>\bigtriangleup</code> .....	111	
<code>\bool_gset_false:N</code> .....	200	
<code>\bool_if:N</code> ...	92, 105, 125, 137, 149,	
.....	159, 218, 296, 312, 334, 345, 355,	
<code>\bool_if:NT</code> .....	10, 12, 23, 40,	
.....	51, 68, 78, 264, 280, 453, 464, 474,	
.....	485, 501, 518, 533, 543, 559, 569,	
.....	609, 626, 643, 660, 675, 685, 697, 707	
<code>\bool_if:nT</code> .....	14, 31	
<code>\bool_if:N</code> .....		
.....	10, 21, 38, 49, 63, 73–80, 90, 99,	
.....	120, 125, 132, 144, 146, 154, 209,	
.....	258, 274, 290, 306, 339, 360, 377,	
.....	394, 410, 416, 424, 434, 458, 479,	
.....	495, 512, 528, 538, 554, 564, 603,	
.....	620, 637, 654, 670, 680, 692, 702,	
.....	715, 732, 749, 766, 783, 793, 805, 815	
<code>\bool_if:nTF</code> .....	128	
<code>\bool_lazy_and:nnT</code> .....	29	
<code>\bool_lazy_and:nnTF</code> .....	51	
<code>\bool_new:N</code> .....	2–28	
<code>\bool_set_false:N</code> .....		
.....	21, 28, 33, 38, 43, 48, 53, 55,	
.....	59–63, 66, 68–70, 73, 77, 80, 116–	
.....	120, 123, 125, 130, 142, 145, 149,	
.....	153, 154, 161, 165, 166, 173, 189, 196	
<code>\bool_set_true:N</code> .....	29, 33, 34, 39,	
.....	44, 49, 54, 60, 67, 74–76, 81–84, 87,	
.....	124, 126, 127, 131–134, 137, 138,	
.....	143, 144, 150, 156, 162, 168, 172, 186	
<code>\box</code> .....	281	
<code>\box_dp:N</code> .....	43, 217	
<code>\box_ht:N</code> .....	43, 217, 297	
<code>\box_move_up:nn</code> .....	41, 177, 214	
<code>\box_set_ht:Nn</code> .....	295	
<code>\box_use:N</code> .....	46, 184, 221	
<code>\box_use_clear:N</code> .....	49, 228, 300	
<code>\box_wd:N</code> .....	177, 216	
<code>\boxze</code> .....	164	
<code>\bullet</code> .....	114	
<b>C</b>		
<code>\C</code> .....	140	
<code>\c@@_math_alphabet_name_Greek_t1</code> .	33	
<code>\c@@_math_alphabet_name_greek_t1</code> .	32	
<code>\c@@_math_alphabet_name_Latin_t1</code> .	31	
<code>\c@@_math_alphabet_name_latin_t1</code> .	30	
<code>\c@@_math_alphabet_name_misc_t1</code> ..	35	
<code>\c@@_math_alphabet_name_num_t1</code> ...	34	
<code>\c_group_begin_token</code> .....	91, 103	
<code>\c_group_end_token</code> .....	150	

<code>\c_math_toggle_token</code> .....	33,	<code>\crampedscriptscriptstyle</code> .....	8
	37, 105, 109, 139, 142, 169, 173,	<code>\crampedscriptstyle</code> .....	8, 30
	181, 186, 195, 199, 203, 207, 287, 292	<code>\crampedtextstyle</code> .....	8, 24
<code>\c_parameter_token</code> .....	108	<code>\crcr</code> .....	87, 111
<code>\c_space_tl</code> .....	89, 90	<code>\cs_generate_variant:Nn</code> .....	
<code>\c_two</code> .....	44		8, 9, 32, 33, 40, 85, 99
<code>\c_two_hundred_fifty_five</code> .....	291	<code>\cs_gset:cpn</code> .....	184–186
<code>\c_zero</code> .....	130, 131, 270, 291	<code>\cs_gset:cpx</code> .....	77
<code>\calculate@math@sizes</code> .....	7	<code>\cs_gset:Nn</code> .....	269
<code>\catcode</code> .....	71, 72	<code>\cs_gset:Npn</code> .....	10, 48, 187–193
<code>\cdots</code> .....	63, 64, 153	<code>\cs_gset:Npx</code> .....	75
<code>\cdp@elt</code> .....	80	<code>\cs_gset_eq:NN</code> .....	21
<code>\cdp@list</code> .....	80	<code>\cs_gset_protected_nopar:cpx</code>	87, 109, 111
<code>\char_gset_active_eq:nN</code> .....	73	<code>\cs_gset_protected_nopar:Npx</code> .....	94, 102, 116, 121
<code>\char_set_catcode_active:N</code> .....	6, 44, 67, 260, 261	<code>\cs_if_eq:cNF</code> .....	7
<code>\char_set_catcode_active:n</code> ...	262–268	<code>\cs_if_eq:NNTF</code>	10, 15, 18, 21, 24, 27, 30, 33
<code>\char_set_catcode_other:n</code> .....	36	<code>\cs_if_exist:cF</code> .....	7, 19
<code>\char_value_catcode:n</code> .....	34	<code>\cs_if_exist:cT</code> .....	195
<code>\chi</code> .....	93	<code>\cs_if_exist:cTF</code> .....	75, 166, 171
<code>\circ</code> .....	113	<code>\cs_if_exist:Nf</code> .....	5, 20, 51
<code>\cirfnint</code> .....	311	<code>\cs_if_exist:NT</code> .....	4, 218
<code>\clist_clear:N</code> .....	69	<code>\cs_if_exist:NTF</code> .....	185
<code>\clist_if_empty:NT</code> .....	158	<code>\cs_new:cn</code> .....	67
<code>\clist_if_in:NnT</code> .....	322	<code>\cs_new:Nn</code> .....	2, 3, 8, 10,
<code>\clist_map_break:</code> .....	179, 185		12, 14–16, 18, 19, 24, 34, 38, 41, 42,
<code>\clist_map_inline:Nn</code> .....	53, 171, 192		44, 46, 49, 50, 54, 55, 58, 62, 66, 69,
<code>\clist_map_inline:nn</code> .....	5, 36, 101, 108, 111,		71, 72, 78, 80, 83, 92, 98, 100, 105,
	120, 204, 230, 252, 261, 269, 277,		111, 114, 118, 119, 146, 150, 154,
	297, 312, 346, 352, 357, 362, 370, 378		169, 171, 197, 223, 226, 228, 233,
<code>\clist_new:N</code> .....	45, 47–50		243, 245, 249, 250, 255, 259, 266,
<code>\clist_put_right:Nx</code> .....	114		267, 275, 295, 305, 310, 320, 327,
<code>\clist_set:Nn</code> .....	46, 48		332, 337, 342, 350, 355, 360, 368, 376
<code>\clist_set:No</code> .....	154	<code>\cs_new:Npn</code>	2–6, 61, 65, 67, 73, 79, 82, 85,
<code>\Colon</code> .....	333, 342		90, 95, 100, 105, 114, 132, 159, 164,
<code>\colon</code> .....	139		170, 176, 182, 187, 192, 214, 233, 267
<code>\coloncolon</code> .....	82, 338	<code>\cs_new_eq:Nc</code> .....	6
<code>\coloncolonequals</code> .....	86, 338	<code>\cs_new_eq:NN</code> .....	21
<code>\Coloneq</code> .....	335, 346	<code>\cs_new_nopar:Npn</code> .....	13, 76, 90
<code>\coloneq</code> .....	334, 344	<code>\cs_new_protected_nopar:Nn</code> .....	2, 32, 36, 47, 62, 66, 70, 74, 88, 182
<code>\Coloneqq</code> .....	73, 332	<code>\cs_new_protected_nopar:Npn</code> ...	80, 94
<code>\coloneqq</code> .....	72, 332	<code>\cs_set:cpn</code> .....	22
<code>\colonequals</code> .....	84, 338	<code>\cs_set:cpx</code> .....	117
<code>\colonsep</code> .....	89	<code>\cs_set:Nn</code> .....	2, 7, 27, 33, 39, 45,
<code>\color@endgroup</code> .....	293		56, 66, 72, 78, 83, 203, 207, 238, 248
<code>\color@setgroup</code> .....	286	<code>\cs_set:Npn</code> .....	2, 4, 13, 15,
<code>\copy</code> .....	161		18, 23, 28, 58, 63, 71, 82, 87, 120, 234
<code>\crampeddisplaystyle</code> .....	8, 18	<code>\cs_set_eq:cc</code> .....	110



<code>\fi</code> .....	8, 15, 21, 27, 84, 163, 164, 241, 275, 276, 278	<code>\g_@_bfupLatin_bool</code> .....	12, 77, 119, 126, 133, 334, 345, 453, 464
<code>\fi:</code> .....	16	<code>\g_@_bfuplatin_bool</code> .....	13, 78, 120, 127, 134, 355, 366, 474, 485
<code>\fint</code> .....	311	<code>\g_@_char_nrange_clist</code> .....	48
<code>\fmtname</code> .....	27	<code>\g_@_default_mathalph_seq</code> ..	<u>60</u> , 81, 137
<code>\font</code> .....	60, 64	<code>\g_@_fam_int</code> .....	22, 23, 29
<code>\fontdimen</code> 14, 40, 44, 48, 60, 64, 68, 78–	80, 91, 148, 212, 218, 225, 264, 265, 267	<code>\g_@_family_tl</code> .....	75
<code>\fontfamily</code> .....	199	<code>\g_@_font_keyval_tl</code> .....	74
<code>\fontname</code> .....	43	<code>\g_@_fontname_tl</code> .....	71
<code>\fontspec_if_script:nF</code> .....	200	<code>\g_@_literal_bool</code> ...	6, 10, 21, 38, 49, 59, 66, 73, 80, 87, 90, 99, 120, 132
<code>\fontspec_set_family:Nnn</code> .....	8, 37, 38, 49, 50, 60, 61	<code>\g_@_literal_colon_bool</code> .....	23, 146, 172, 173, 218
<code>\fontspec_set_family:Nxn</code> ..	13, 113, 148	<code>\g_@_literal_Nabla_bool</code> .....	20, 63, 144, 149, 153, 156, 410, 434, 528, 564, 670, 702, 783, 815
<code>\fontspec_set_fontface:NNnn</code> .....	9	<code>\g_@_literal_partial_bool</code> .....	21, 73, 154, 161, 165, 168, 416, 424, 538, 554, 680, 692, 793, 805
<code>\fontspec_set_fontface:NNxn</code> .....	196	<code>\g_@_mainfont_already_set_bool</code> ...	5, 14, 31, 33, 55
<code>\forks</code> .....	193	<code>\g_@_mathbf_text_bool</code> .....	26, 43, 44
<code>\fp_compare:nF</code> .....	83	<code>\g_@_mathclasses_seq</code> .....	48, <u>51</u>
<code>\fp_eval:n</code> .....	102, 148, 152	<code>\g_@_mathit_text_bool</code> .....	25, 38, 39
<code>\frac</code> .....	77	<code>\g_@_mathrm_text_bool</code> .....	24, 28, 29, 33, 34
<b>G</b>			
<code>\g</code> .....	169	<code>\g_@_mathsf_text_bool</code> .....	27, 48, 49
<code>\g_@_alphabet_clist</code> .....	50	<code>\g_@_mathstyles_seq</code> .....	20, <u>61</u>
<code>\g_@_alphabets_seq</code> .....	<u>45</u>	<code>\g_@_mathtable_tl</code> .....	9, 10, 70
<code>\g_@_bfit_Greek_usv</code> .....	79	<code>\g_@_mathtt_text_bool</code> .....	28, 53, 54
<code>\g_@_bfit_greek_usv</code> .....	80	<code>\g_@_mversion_tl</code> .....	72
<code>\g_@_bfit_Latin_usv</code> .....	77	<code>\g_@_named_ranges_clist</code> ...	47, 48, 53
<code>\g_@_bfit_latin_usv</code> .....	78	<code>\g_@_named_ranges_seq</code> .....	61, 79, 80
<code>\g_@_bfliteral_bool</code> .....	11, 116, 123, 130, 137, 339, 360, 377, 394, 458, 479, 495, 512	<code>\g_@_operator_mathfont_tl</code> ..	14, 16, 39
<code>\g_@_bfsfit_Greek_usv</code> .....	75	<code>\g_@_primekern_muskip</code> .....	13, 63, 64
<code>\g_@_bfsfit_greek_usv</code> .....	76	<code>\g_@_remap_style_tl</code> .....	77
<code>\g_@_bfsfit_Latin_usv</code> .....	73	<code>\g_@_sfliteral_bool</code> .....	17, 144, 258, 274, 290, 306, 603, 620, 637, 654, 715, 732, 749, 766
<code>\g_@_bfsfit_latin_usv</code> .....	74	<code>\g_@_slash_delimiter_usv</code> .....	69, 177–179, 273–275
<code>\g_@_bfsfup_Greek_usv</code> .....	75	<code>\g_@_style_tl</code> .....	76
<code>\g_@_bfsfup_greek_usv</code> .....	76	<code>\g_@_subs_prop</code> .....	43, 67
<code>\g_@_bfsfup_Latin_usv</code> .....	73	<code>\g_@_supers_prop</code> .....	5, 66
<code>\g_@_bfsfup_latin_usv</code> .....	74	<code>\g_@_symfont_tl</code> .....	73
<code>\g_@_bfup_Greek_usv</code> .....	79	<code>\g_@_unknown_keys_clist</code> .....	49
<code>\g_@_bfup_greek_usv</code> .....	80	<code>\g_@_upGreek_bool</code> .....	9, 40, 60, 67, 74, 75, 81, 125
<code>\g_@_bfup_Latin_usv</code> .....	77		
<code>\g_@_bfup_latin_usv</code> .....	78		
<code>\g_@_bfupGreek_bool</code> .....	14, 79, 117, 124, 131, 383, 501		
<code>\g_@_bfupgreek_bool</code> .....	15, 80, 118, 125, 132, 400, 518		



<b>L</b>	
<code>\L</code> .....	146
<code>\l_@@_alphabet_clist</code> .....	154, 158, 162, 171, 192
<code>\l_@@_alphabet_tl</code> .....	173, 174, 176, 182, 189, 194, 195, 197, 199, 200, 203, 205, 206, 214, 218
<code>\l_@@_char_nrange_clist</code> ...	69, 114, 322
<code>\l_@@_char_range_seq</code> ..	31, 42, 53, 68, 103
<code>\l_@@_cmd_range_seq</code> .....	33, 44, 52, 99
<code>\l_@@_family_tl</code> .....	22, 25, 29, 113, 138, 143, 148, 163, 168, 196, 199
<code>\l_@@_font</code> .....	22, 27, 32, 40, 43, 44, 47, 48, 52, 68, 91, 94, 95, 157, 160, 182, 196, 203, 212, 218, 225
<code>\l_@@_font_keyval_tl</code> ..	115, 150, 173, 197
<code>\l_@@_fontname_tl</code> .....	4, 14, 51, 65, 66, 135, 160, 197
<code>\l_@@_implicit_alph_bool</code>	4, 138, 145, 209
<code>\l_@@_init_bool</code> .....	3, 10, 21, 67
<code>\l_@@_keyval_clist</code> .....	11, 14
<code>\l_@@_mathalph_seq</code> .....	34, 40, 41, 70, 134, 137, 151
<code>\l_@@_mathstyle_tl</code> .....	4, 34, 36
<code>\l_@@_mclass_range_seq</code> ...	32, 43, 49, 95
<code>\l_@@_missing_alph_seq</code>	40, 44, 71, 167, 211
<code>\l_@@_mversion_tl</code> .....	5–7, 9, 16, 18, 22, 24, 26, 61, 137, 140, 162, 165
<code>\l_@@_nolimits_tl</code> ....	19, 23, 38, 80, 307
<code>\l_@@_ot_math_bool</code> ...	2, 14, 31, 60, 200
<code>\l_@@_primecount_int</code> .....	64, 65, 70, 76, 82, 87, 92, 97, 102, 107, 119, 125, 131, 137, 143, 149, 153, 167, 173, 179, 184, 189, 194, 199, 217, 223, 229, 235, 239
<code>\l_@@_radical_sqrt_tl</code> ....	60, 134, 175
<code>\l_@@_radicals_tl</code> .....	37, 85, 315
<code>\l_@@_remap_style_tl</code> ..	155, 200, 203, 206
<code>\l_@@_script_features_tl</code> ...	12, 63, 185
<code>\l_@@_script_font_tl</code> .....	14, 65, 184
<code>\l_@@_smallfrac_bool</code>	22, 77, 186, 189, 196
<code>\l_@@_ss_chain_tl</code> ....	12, 50, 102, 118
<code>\l_@@_ssscript_features_tl</code> ...	13, 64, 190
<code>\l_@@_ssscript_font_tl</code> .....	15, 66, 189
<code>\l_@@_style_tl</code> .....	153, 161, 174, 178, 182, 184, 189, 195, 199, 200, 205, 206, 213, 218
<code>\l_@@_symfont_label_tl</code>	19, 21, 23, 24, 28, 62, 205, 227, 231, 252, 257, 270, 316
<code>\l_@@_tmpa_tl</code> .....	13, 14, 19, 22, 27–29, 42, 52, 56, 57, 59, 61, 65, 66, 68, 69, 71–74, 77, 79, 84, 89, 116
<code>\l_@@_tmpb_tl</code> .....	43, 62, 90, 116, 118
<code>\l_@@_tmpc_tl</code> .....	44, 63, 76, 77, 85
<code>\l_@@_unknown_keys_clist</code> .....	9, 193
<code>\l_MT_bracketheight_fdim</code> ....	315, 322
<code>\l_not_token_name_tl</code> .....	68, 161, 162, 164, 166, 168, 171, 173
<code>\l_tmpa_box</code> .....	31, 43, 49, 167, 177, 193, 217, 228, 284, 295, 297, 300
<code>\l_tmpb_box</code> .....	201, 216
<code>\land</code> .....	120
<code>\latex_underbar:n</code> .....	134, 137
<code>\le</code> .....	108
<code>\leaders</code> .....	317, 324
<code>\left</code> .....	269
<code>\leftarrow</code> .....	123
<code>\leftroot@</code>	131, 159, 163, 183, 185, 209, 227
<code>\leq</code> .....	108
<code>\let</code> .....	327–330
<code>\Let@</code> .....	75, 92
<code>\limits</code> .....	125, 133
<code>\lineskip</code> .....	80, 81, 100, 101
<code>\lineskiplimit</code> .....	81, 101
<code>\lnot</code> .....	125
<code>\longdivision</code> .....	126, 315
<code>\longdivisionsign</code> .....	126
<code>\lor</code> .....	121
<code>\lowint</code> .....	313
<code>\luatexUroot</code> .....	21
<code>\luatexversion</code> .....	9, 10
<b>M</b>	
<code>\M</code> .....	147
<code>\m@th</code> .....	34, 85, 106, 118, 140, 154, 157, 170, 182, 196, 204, 257, 288
<code>\math</code> .....	72
<code>\math@bgroup</code> .....	6, 28
<code>\math@egroup</code> .....	10, 23
<code>\mathaccent</code> .....	48, 49, 58
<code>\mathaccentwide</code> .....	53, 58, 67
<code>\mathalpha</code> ....	18, 41, 54, 227, 230, 316
<code>\mathbacktick</code> .....	243
<code>\mathbf</code> .....	41, 70, 75
<code>\mathbin</code> .....	18, 42, 54, 216, 217
<code>\mathbotaccent</code> .....	51, 58, 66
<code>\mathbotaccentwide</code> .....	55, 58, 68
<code>\mathchar</code> .....	46–49



<code>\oint</code> .....	309	<code>\r@@@et</code> .....	29, 121–124, 145–148, 152
<code>\oint</code> .....	309	<code>\r@@t</code> .....	29
<code>\ointctrlockwise</code> .....	310	<code>\radical</code> .....	260
<code>\operator@font</code> .....	13	<code>\raise</code> .....	161
<code>\overbracket</code> .....	60, 304	<code>\ratio</code> .....	81, 338
<code>\owns</code> .....	122	<code>\relax</code> .....	9, 57, 58, 60, 61, 64, 68, 71, 72, 119–130, 133, 153–157, 240
<b>P</b>			
<code>\P</code> .....	149	<code>\removenolimits</code> .....	21
<code>\PackageError</code> .....	4, 11, 17	<code>\renewcommand</code> .....	135
<code>\peek_meaning_remove:NTF</code> ...	6, 108, 111, 114, 117, 123, 129, 135, 141, 147, 200, 205, 210, 215, 221, 227, 233	<code>\RenewDocumentCommand</code> .....	341–346
<code>\peek_N_type:TF</code> .....	107	<code>\RequirePackage</code> .....	4, 22–26
<code>\phi</code> .....	92	<code>\RequirePackageWithOptions</code> .....	28, 29
<code>\plainroot@</code> .....	114	<code>\resetMathstrut@</code> .....	66
<code>\pointint</code> .....	312	<code>\restore@math@cr</code> .....	76, 93
<code>\prg_do_nothing:</code> .....	42	<code>\restore@mathversion</code> .....	84
<code>\prg_new_conditional:Nnn</code> ...	10, 59, 73	<code>\rho</code> .....	91
<code>\prg_replicate:nn</code> .....	13	<code>\right</code> .....	269
<code>\prg_return_false:</code> .....	15, 76, 80	<code>\rightarrow</code> .....	107
<code>\prg_return_true:</code> .....	13, 76, 80	<code>\rmdefault</code> .....	35
<code>\prime</code> .....	251	<code>\root</code> .....	58
<code>\process@table</code> .....	84	<code>\rootbox</code> .....	46, 116, 137, 161, 184, 221
<code>\ProcessKeysOptions</code> .....	219	<code>\rppolint</code> .....	311
<code>\prop_get:cnN</code> .....	161	<b>S</b>	
<code>\prop_get:cnNTF</code> .....	116	<code>\sb</code> .....	51
<code>\prop_gput:cn</code> .....	60, 82	<code>\sbox</code> .....	254
<code>\prop_gput:Nnn</code> .....	5, 43	<code>\scan_stop:</code> ...	5, 12, 21, 26, 31, 36, 40, 44, 48, 52, 56, 236–238, 242–244
<code>\prop_if_exist:cF</code> .....	57	<code>\scantokens</code> .....	8, 46
<code>\prop_item:cn</code> .....	62	<code>\scpolint</code> .....	312
<code>\prop_new:c</code> .....	46	<code>\scriptfont</code> .....	28, 31, 78–80, 272
<code>\prop_new:N</code> .....	66, 67	<code>\scriptscriptfont</code> .....	34, 36, 274
<code>\protect</code> .....	9	<code>\scriptscriptstyle</code> .....	33, 118, 124, 141, 148, 159
<code>\protected</code> .....	107–133, 149, 150	<code>\scriptstyle</code> .....	27, 85, 97, 98, 100, 107, 123, 147, 156, 271
<code>\ProvidesFile</code> .....	33	<code>\selectfont</code> .....	199
<code>\ProvidesPackage</code> .....	29–31	<code>\seq_clear:N</code> .....	31–34, 68, 70, 71
<b>Q</b>			
<code>\Q</code> .....	150	<code>\seq_if_empty:N</code> .....	167
<code>\q_nil</code> .....	66, 69, 82, 87, 111, 119	<code>\seq_if_empty:NTF</code> .....	134
<code>\q_recursion_stop</code> .....	17	<code>\seq_if_in:NnT</code> .....	95, 99
<code>\q_recursion_tail</code> .....	17	<code>\seq_if_in:NnTF</code> .....	48
<code>\q_stop</code> .....	119, 120	<code>\seq_if_in:NVTF</code> .....	79
<code>\qprime</code> .....	254	<code>\seq_map_break:n</code> .....	106
<code>\quark_if_recursion_tail_stop:n</code>	21, 22	<code>\seq_map_function:NN</code> .....	44
<code>\quark_new:N</code> .....	3	<code>\seq_map_inline:Nn</code> .....	103, 151
<b>R</b>			
<code>\R</code> .....	151	<code>\seq_new:N</code> .....	40–44, 51, 60–62
		<code>\seq_put_right:Nn</code> ...	20, 49, 52, 53, 81
		<code>\seq_put_right:Nx</code> .....	40, 80, 211

<code>\seq_set_eq:NN</code> .....	137	<code>\tf@size</code> .....	180, 183	
<code>\seq_set_from_clist:Nn</code> .....	52	<code>\tfrac</code> .....	9, 77, 185, 218	
<code>\set@mathdelimiter</code> .....	91	<code>\the</code> .....	68	
<code>\set@mathaccent</code> .....	88	<code>\thinmuskip</code> .....	64	
<code>\set@mathchar</code> .....	88	<code>\thr@@@</code> .....	80	
<code>\set@mathdelimiter</code> .....	90	<code>\tl_case:Nn</code> .....	25, 38	
<code>\set@mathsymbol</code> .....	89	<code>\tl_clear:N</code> .....	9, 62, 63	
<code>\setbox</code> .....	68, 116, 157	<code>\tl_const:cn</code> .....	71	
<code>\setboxz@h</code> .....	154	<code>\tl_const:Nn</code> .....	30–35	
<code>\SetMathAlphabet</code> .. 22, 39–41, 51, 52, 62, 63, 68–70, 74–76, 80, 81, 85–87		<code>\tl_if_empty:nF</code> .....	162	
<code>\SetMathAlphabet@</code> .....	87	<code>\tl_if_empty:NT</code> .....	16, 76	
<code>\setmathfont</code> .....	3, 54	<code>\tl_if_empty:nTF</code> .....	122, 124, 126	
<code>\setmathfontface</code> .....	7	<code>\tl_if_eq:nnT</code> .....	32	
<code>\setmathrm</code> .....	24	<code>\tl_if_eq:nnTF</code> .....	197	
<code>\setoperatorfont</code> .....	12	<code>\tl_if_eq:onT</code> .....	35, 47, 58	
<code>\SetSymbolFont</code> 24, 28, 86, 137, 142, 162, 167		<code>\tl_if_in:NnT</code> .....	65, 68, 80	
<code>\SetSymbolFont@</code> .....	86	<code>\tl_if_in:nnT</code> .....	17	
<code>\sf@size</code> .....	183, 188	<code>\tl_if_in:NnTF</code> .....	85	
<code>\sfdefault</code> .....	47	<code>\tl_if_single_p:n</code> .....	51	
<code>\skip_set:Nn</code> .....	95	<code>\tl_map_inline:nn</code> .....	78	
<code>\slash</code> .....	154, 155	<code>\tl_new:N</code> .....	36–39, 68–77	
<code>\smallint</code> .....	133	<code>\tl_put_left:Nn</code> .....	43	
<code>\smblkcircle</code> .....	114	<code>\tl_put_right:cx</code> .....	314	
<code>\smwhtdiamond</code> .....	117	<code>\tl_put_right:Nn</code> .....	19, 44	
<code>\sp</code> .....	13	<code>\tl_put_right:NV</code> .....	118	
<code>\space</code> .....	32, 46, 213	<code>\tl_remove_all:Nn</code> .....	23, 57, 72, 73	
<code>\sqint</code> .....	312	<code>\tl_remove_once:Nn</code> .....	55, 95	
<code>\sqrt</code> .....	315	<code>\tl_rescan:nn</code> .....	69	
<code>\sqrtsign</code> .....	36, 154, 198	<code>\tl_set:cn</code> .....	89	
<code>\std=equal</code> .....	58, 61	<code>\tl_set:cx</code> .....	14–16	
<code>\std=minus</code> .....	57, 60, 240	<code>\tl_set:Nn</code> .... 4–6, 12, 14, 18, 34, 50, 52, 56, 61–64, 173, 177–179, 307, 315		
<code>\sterling</code> .....	116	<code>\tl_set:No</code> .....	153, 155	
<code>\str_if_eq_x:nnT</code> .....	26, 140, 165	<code>\tl_set:Nx</code> .....	23, 71, 84, 85, 89, 90, 161, 164, 173, 194	
<code>\str_if_eq_x:nnTF</code> .....	176	<code>\tl_set_eq:NN</code> .....	65, 66, 77	
<code>\string</code> .....	63–68	<code>\tl_set_from_file_x:Nnn</code> .....	9	
<code>\subarray</code> .....	71	<code>\tl_tail:V</code> .....	162, 164	
<code>\sumint</code> .....	310	<code>\tl_to_str:N</code> .....	71	
<code>\sym</code> .....	73	<code>\tl_to_str:n</code> .....	80	
<code>\symit</code> .....	16	<code>\tl_trim_spaces:N</code> .....	74	
<code>\symrm</code> .....	131	<code>\tl_trim_spaces:n</code> .... 84, 85, 89, 90, 194		
<code>\symup</code> .....	15, 131	<code>\tl_use:c</code> .....	214	
<code>\sys_if_engine luatex:T</code> .....	28	<code>\tl_use:N</code> .....	4	
<code>\sys_if_engine xetex:T</code> .....	29	<code>\tmpa</code> .....	26, 27	
<b>T</b>				
<code>\TeX</code> .....	109	<code>\to</code> .....	107	
<code>\textfont</code> .... 16, 19, 22, 25, 264, 265, 269		<code>\token_if_cs_p:N</code> .....	51	
<code>\textstyle</code> .....	21, 122, 133, 146, 268	<code>\token_if_eq_meaning:NNT</code> .....	104	
		<code>\token_if_macro:NTF</code> .....	6	

