

Experimental unicode mathematical typesetting: The unimath package

Will Robertson

2006/02/20 v0.01

Contents

1	Introduction	1	5.1	Enlarging the number of maths families	4
2	Current NFSS methods	1	5.2	<code>\DeclareMathSymbol</code> for unicode ranges	4
3	Specification	2	5.3	User interface to <code>\DeclareSymbolFont</code>	6
3.1	Dealing with real life	2	5.4	Big operators	7
I	The unimath package	3	5.5	Setting up the ASCII ranges	7
4	Things we need	3	6	fontspec feature hooks	8
4.1	Packages	3	6.1	Range processing	9
4.2	Counters and conditionals	3	II	STIX table data extraction	10
4.3	Programming macros	3	A	Documenting the NFSS	11
4.4	Overcoming <code>\@on-lypreamble</code>	3	A.1	Maths symbols	11
5	Fundamentals	4	A.2	Symbol fonts	13

1 Introduction

This document describes the unimath package, which is an *experimental* implementation of a macro to unicode glyph encoding for mathematical characters. Its intended use is for \XeTeX , although it is conjectured that small effect needs to be spent to create a cross-format package that would also work with \LaTeX .

As of \XeTeX v. 0.995, maths characters can be accessed in unicode ranges. Now, a proper method must be invented for real unicode maths support. Before any code is written, I'm writing a specification in order to work out what is required. Fairly significant pieces of the NFSS may have to be re-written, and I'm a little unsure where to start.

2 Current NFSS methods

In the following, $\langle NFSS\ decl. \rangle$ stands for something like $\{T1\}\{1mr\}\{m\}\{n\}$.

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

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

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

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

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

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

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

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

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

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

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

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

Delimiters, accents, and radicals are not dealt with yet.

3 Specification

In the ideal case, a single unicode font will contain all maths glyphs we need. Barbara Beeton’s STIX table provides the mapping between unicode maths glyphs and macro names (all 3298 of them!). A single command

`\setmathsfont[]\{\}`

would implement this for every every symbol and alphabetic variant. That means `x` to `x`, `\xi` to ξ , `\leq` to \leq , etc., `\mathcal{H}` to \mathcal{H} and so on, all for unicode glyphs within a single font.

Furthermore, this package should deal well with unicode characters for maths input, as well. This includes using literal Greek letters in formulae, resolving to upright or italic depending on preference.

Finally, maths versions must also be provided for. While I guess version selection in \LaTeX will remain the same, the specification for choosing the version fonts will probably be an optional argument:

`\setmathsfont[Version=Bold,]\{\}`

All instances of ‘maths’ in command names will be aliased to ‘math’ for our American (or abbreviatory-minded) friends. Instances above of `[{font features}]{{font name}}` follow from my `fontspec` package, and therefore any additional *{font features}* specific to maths fonts will hook into `fontspec`’s methods.

3.1 Dealing with real life

Let’s face it; there will probably be few cases where a single unicode maths font suffices. The upcoming STIX font comes to mind as a notable exception. It will therefore be necessary to delegate specific unicode ranges of glyphs to separate fonts. This syntax will also hook into the `fontspec` font feature processing:

```
\setmathsfont[Range={unicode range}],{font features}]{{font name}}
```

where *{unicode range}* is a comma-separated list of unicode slots and ranges such as `{27D0-27EB, 27FF, 295B-297F}`. Furthermore, preset names ranges could be used, such as `MiscMathSymbolsA`, with such ranges based on unicode chunks. The amount of optimisation required here to achieve acceptable performance has yet to be determined. Techniques such as saving out unicode subsets based on *{unicode range}* data to be `\input` in the next \LaTeX run are a possibility, but at this stage, performance without such measures seems acceptable.

File I

The unimath package

This is the package.

```
1 \ProvidesPackage{unimath}
2 [2006/02/20 v0.01 Unicode maths definitions]
```

4 Things we need

4.1 Packages

```
3 \RequirePackage{fontspec}
```

4.2 Counters and conditionals

```
4 \newcounter{um@fam}
5 \newif\if@um@fontspec@feature
```

4.3 Programming macros

```
\um@Loop See Kees van der Laan’s various articles on  $\text{\TeX}$  programming:
\um@Break
6 \def\um@Loop#1\um@Pool{#1\um@Loop#1\um@Pool}
7 \def\um@Break#1\um@Pool{}
```

`\um@FOR` A simple ‘for’ loop implemented with the above. Takes a (predefined) counter `\csname` and increments it between two integers, iterating as we go.

```

8 \long\def\um@FOR #1 = [#2:#3] #4{%
9   {\csname#1\endcsname =#2\relax
10  \um@Loop #4%
11   \expandafter\advance\csname#1\endcsname\@ne
12   \expandafter\ifnum\csname#1\endcsname>#3\relax
13     \expandafter\um@Break
14   \fi
15  \um@Pool}}

```

`g/h/i/j/k/l/m/`

`\newcount\@ii`
`\um@FOR @ii = [7:13] {\@alph\@ii/}`

4.4 Overcoming `\@onlypreamble`

This will be refined later!

```

16 \def\@preamblecmds{}

```

5 Fundamentals

5.1 Enlarging the number of maths families

To start with, we’ve got a power of two as many `\fams` as before. So (from `l1tfssbas.dtx`) we want to redefine

```

17 \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cc1vi}
18 \let\newfam\new@mathgroup

```

Up to math fam 25 of 255.

```

\um@FOR @tempcnta = [1:20]
{\expandafter\newfam
 \csname mt\@alph\@tempcnta\endcsname}
Up to math fam \the\mtt\ of 255.

```

This is sufficient for L^AT_EX’s `\DeclareSymbolFont`-type commands to be able to define 256 named maths fonts. Now we need a new `\DeclareMathSymbol`.

5.2 `\DeclareMathSymbol` for unicode ranges

This is mostly an adaptation from L^AT_EX’s definition.

```

\DeclareUnicodeMathSymbol #1 : Symbol, e.g., \alpha or a
#2 : Type, e.g., \mathalpha
#3 : Math font name, e.g., operators
#4 : Slot, e.g., "221E
19 \def\DeclareUnicodeMathSymbol#1#2#3#4{%

```

First ensure the math font (*e.g.*, operators) exists:

```
20 \expandafter\in@\csname sym#3\expandafter\endcsname
21 \expandafter{\group@list}%
22 \ifin@
```

No longer need here to perform the obfuscated hex conversion, since `\XeTeX-extmathchar` (and friends) has a more simplified input than `TEX's \mathchar`.

```
23 \begingroup
```

The symbol to be defined can be either a command (`\alpha`) or a character (`a`). Branch for the former:

```
24 \if\relax\noexpand#1% is command?
25 \edef\reserved@a{\noexpand\in@{\string\XeTeXextmathchar}{\meaning#1}}%
26 \reserved@a
```

If the symbol command definition contains `\XeTeXextmathchar`, then we can provide the info that a previous symbol definition is being overwritten:

```
27 \ifin@
28 \expandafter\um@set@mathsymbol
29 \csname sym#3\endcsname#1#2{#4}%
30 \@font@info{Redeclaring math symbol \string#1}%
```

Otherwise, overwrite it if the symbol command definition contains plain old `\mathchar`:

```
31 \else
32 %\edef\reserved@a{\noexpand\in@{\string\mathchar}{\meaning#1}}%
33 %\reserved@a
34 %\ifin@
35 % \expandafter\set@xmathsymbol
36 % \csname sym#3\endcsname#1#2{#4}%
```

Otherwise, throw an error if the command name is already taken by a non-symbol definition:

```
37 %\else
38 %\expandafter\ifx
39 %\csname\expandafter\@gobble\string#1\endcsname
40 %\relax
41 \expandafter\um@set@mathsymbol
42 \csname sym#3\endcsname#1#2{#4}%
43 %\else
44 % \@latex@error{Command '\string#1' already defined}\@eha
45 %\fi
46 %\fi
47 \fi
```

And if the symbol input is a character:

```
48 \else
49 \expandafter\um@set@mathchar
50 \csname sym#3\endcsname#1#2{#4}%
51 \fi
52 \endgroup
```

Everything previous was skipped if the maths font doesn't exist in the first place:

```
53 \else
54 \latex@error{Symbol font `#3' is not defined}\@eha
55 \fi}
```

The final macros that actually define the maths symbol with X_YTeX primitives.

`\um@set@mathsymbol` If the symbol definition is for a macro:

```
56 \def\um@set@mathsymbol#1#2#3#4{%
57 \global\XeTeXextmathchardef#2="\mathchar@type#3 #1 #4\relax}
```

`\um@set@mathchar` Or if it's for a character:

```
58 \def\um@set@mathchar#1#2#3#4{%
59 \global\XeTeXextmathcode`#2="\mathchar@type#3 #1 #4\relax}
```

∞

```
\zf@fontspec{}\{Cambria Math}
\let\glb@currsizel\relax
\DeclareSymbolFont{test}{EU1}{\zf@family}{m}{n}
\DeclareUnicodeMathSymbol{\infinity}{\mathord}{test}{"221E}
$\infinity$
```

`\DeclareUnicodeMathCode` [For later] or if it's for a character code: (just a wrapper around the primitive)

```
60 \def\DeclareUnicodeMathCode#1#2#3#4{%
61 \global\XeTeXextmathcode#1=
62 "\mathchar@type#2 \cename sym#3\endcename #4\relax}
```

A

```
\zf@fontspec{}\{Cambria Math}
\let\glb@currsizel\relax
\DeclareSymbolFont{test2}{EU1}{\zf@family}{m}{n}
\DeclareUnicodeMathCode{65}{\mathalpha}{test2}{119860}
$A$
```

5.3 User interface to `\DeclareSymbolFont`

`\setmathfont` [#1]: font features

#2 : font name

```
63 \newcommand\setmathfont[2][]{%
```

Erase any conception L^AT_EX has of previously defined math symbol fonts; this allows `\DeclareSymbolFont` at any point in the document.

To start with, assume we're defining every math symbol character.

```
64 \let\glb@currsizel\relax
65 \let\um@char@range\empty
```

Use `fontspec` to select a font to use:

```
66 \@um@fontspec@featuretrue
67 \zf@fontspec{#1}{#2}%
```

Probably want to check there that we're not creating multiple symbol fonts with the same NFSS declaration. On that note, fontspec doesn't seem to be keeping track of that, either : (

```
68 \stepcounter{um@fam}%
69 \DeclareSymbolFont{um@fam\theum@fam}
70 {EU1}{\zf@family}{\mddefault}{\updefault}%
```

Now when the list of unicode symbols is input, we want a suitable definition of its internal macro. By default, we want to define every single math char:

```
71 \ifx\um@char@range\empty
72 \um@text@input{um@fam\theum@fam}%
73 \PackageWarning{unimath}{Defining the default maths font as `#2'}
74 \def\unicode@math@symbol##1##2##3##4{%
75 \DeclareUnicodeMathSymbol
76 {##2}{##3}{um@fam\theum@fam}{##1}}%
77 \else
```

If the Range font feature has been used, then only a subset of the unicode glyphs are to be defined. See section 6.1 for the code that enables this.

```
78 \def\unicode@math@symbol##1##2##3##4{%
79 \um@parse@term{##1}{##2}{%
80 \PackageWarning{unimath}{Defining \string##2 as math-
char ##1 from font `#2'}
81 \DeclareUnicodeMathSymbol
82 {##2}{##3}{um@fam\theum@fam}{##1}}}%
83 \fi
```

And now we input every single maths char. See File II for the source to stix-tex.tex.

```
84 \input stix-tex.tex}
85 \let\setmathsfnt\setmathfont
```

Here's the simplest usage:

$Ax \stackrel{\text{def}}{=} \nabla \times Z$	<pre>\setmathfont{Cambria Math} \$Ax \eqdef \nabla \times \scrZ\$</pre>
---	---

And an example of the Range feature:

$(a, \alpha, \mathcal{M}, \aleph, \mathcal{H}, \mathbb{H})$	<pre>\setmathfont{Cambria Math} \$(a, \alpha, \scrM, \aleph, \scrH, \BbbH)\$</pre>
$(a, \alpha, \mathcal{M}, \aleph, \mathcal{H}, \mathbb{H})$	<pre>\setmathfont[Range={"2133-"2135, \scrH, \BbbH}]{Lucida Sans} \$(a, \alpha, \scrM, \aleph, \scrH, \BbbH)\$</pre>
$(a, \alpha, \mathcal{M}, \aleph, \mathcal{H}, \mathbb{H})$	<pre>\setmathfont[Range={"2133-"2135, \scrH, \BbbH}]{Apple Symbols} \$(a, \alpha, \scrM, \aleph, \scrH, \BbbH)\$</pre>

5.4 Big operators

$$\left(\frac{\left(\sum_{i=n}^N\left(\int_0^1(a\times b)\right)\right)}{A_{D_E}^{B^C}}\right)$$

```
\setmathfont{Cambria Math}
\left[\left(\frac{\left(\sum_{i=n}^N\left(\int_0^1(a\times b)\right)\right)}{A_{D_E}^{B^C}}\right)\right]
```

5.5 Setting up the ascii ranges

We want it to be convenient for users to actually type in maths. The ASCII Latin characters should be used for italic maths, and the text Greek characters should be used for upright/italic (depending on preference) Greek, if desired.

`\um@text@input` And here're the text input to maths output mappings, wrapped up in a macro.

```
86 \newcommand\um@text@input[1]{%
```

Latin alphabet, uppercase and lowercase respectively:

```
87 \um@FOR @tempcnta = [0:25] {%
88 \um@mathcode@offset{#1}{65}{119860}%
89 \um@mathcode@offset{#1}{97}{119886}%
90 }%
```

Filling a hole for 'h', which maps to U+210E: PLANCK CONSTANT instead of the expected U+1D455: MATHEMATICAL ITALIC SMALL H (which is not assigned):

```
91 \DeclareUnicodeMathCode{104}{\mathalpha}{#1}{8462}%
```

Greek alphabet, uppercase (note the hole after U+03A1: GREEK CAPITAL LETTER RHO):

```
92 \um@FOR @tempcnta = [0:23] {%
93 \DeclareUnicodeMathCode
94 {\ifnum\@tempcnta>16
95 \numexpr\the\@tempcnta+913\relax
96 \else
97 \numexpr\the\@tempcnta+913+1\relax
98 \fi}
99 {\mathalpha}{#1}
100 {\numexpr\the\@tempcnta+120546\relax}%
```

And Greek lowercase:

```
101 \um@mathcode@offset{#1}{945}{120572}%
102 }%
103 }
```

`\um@mathcode@offset` This is a wrapper macro to save space:

```
104 \newcommand\um@mathcode@offset[3]{%
105 \DeclareUnicodeMathCode
106 {\numexpr\the\@tempcnta+#2\relax}
107 {\mathalpha}{#1}}
```



```

108 {\numexpr\the\@tempcnta+#3\relax}%
109 }

```

<i>ABCDEFGHIJKLMNOPQRSTUVWXYZ</i>	<code>\setmathsf{Cambria Math}</code>
<i>abcdefghijklmnopqrstuvwxyz</i>	<code>\$ABCDEFGHIJKLMNOPQRSTUVWXYZ\$ \</code>
<i>ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΣΤΥΦΧΨ</i>	<code>\$abcdefghijklmnopqrstuvwxyz\$ \</code>
<i>αβγδεζηθικλμνξοπρστυφχψ</i>	<code>\$ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΣΤΥΦΧΨ\$ \</code>
	<code>αβγδεζηθικλμνξοπρστυφχψ\$ \</code>

6 fontspec feature hooks

`\um@zf@feature` Use the same method as fontspec for feature definition (*i.e.*, using `xkeyval`) but with a conditional to restrict the scope of these features to unimath commands.

```

110 \newcommand\um@zf@feature[2]{%
111   \define@key[zf]{options}{#1}{%
112     \if@um@fontspec@feature
113       #2
114     \else
115       \PackageError{fontspec/unimath}
116         {The `#1' font feature can only be used for maths fonts}
117         {The feature you tried to use can only be in commands
118          like \protect\setmathsf}
119     \fi}}

```

6.1 Range processing

```

120 \um@zf@feature{Range}{\xdef\um@char@range{\zap@space#1 \@empty}}

```

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

`\um@parse@term` This macro simply iterates over a comma separated list, passing each potential range value to `\um@parse@range`. Specified ranges r may be

<i>Input</i>	<i>Range</i>
x	$r = x$
$x-$	$r \geq x$
$-x$	$r \leq x$
$x-y$	$x \leq r \leq y$

```

121 \newcommand\um@parse@term[3]{%
122   \@for\@ii:=\um@char@range\do{%
123     \unless\ifx\@ii\@empty
124       \@tempswafalse
125       \expandafter\if\expandafter\relax\expandafter\noexpand\@ii
126         \expandafter\ifx\@ii#2
127           \@tempwattrue
128         \fi
129       \else
130         \expandafter\um@parse@range\@ii-\@marker-\@nil#1\@nil

```

```

131     \fi
132     \if@tempswa
133     #3
134     \fi
135     \fi}}

```

'1' or 'yuck' is included '3' or 'yum' is included '3' or 'yum' is included

```

\def\um@char@range{\yuck,2-4}
\um@parse@term{1}{\yuck}
  { '1' or '\string\yuck' is included}
\um@parse@term{3}{\yum}
  { '3' or '\string\yum' is included}

```

\um@parse@range As mentioned, this macro can be passed four different input types via \um@parse@term.

```

136 \def\um@parse@range#1-#2-#3\@nil#4\@nil{%
137   \def\@tempa{#1}%
138   \def\@tempb{#2}%

```

Range $r = x$
C-list input \@ii=X
Macro input \um@parse@range X-\@marker-\@nil#1\@nil
Arguments #1-#2-#3 = X-\@marker-{}

```

139   \ifx#2\@marker\relax
140     \ifnum#4=#1\relax
141       \@tempwattrue
142     \fi
143   \else

```

Range $r \geq x$
C-list input \@ii=X-
Macro input \um@parse@range X--\@marker-\@nil#1\@nil
Arguments #1-#2-#3 = X-{}-\@marker-

```

144   \ifx\@empty\@tempb
145     \ifnum#4>\numexpr#1-1\relax
146       \@tempwattrue
147     \fi
148   \else

```

Range $r \leq x$
C-list input \@ii=-Y
Macro input \um@parse@range -Y-\@marker-\@nil#1\@nil
Arguments #1-#2-#3 = {}-Y-\@marker-

```

149   \ifx\@empty\@tempa
150     \ifnum#4<\numexpr#2+1\relax
151       \@tempwattrue
152     \fi

```

Range $x \leq r \leq y$
C-list input \@ii=X-Y
Macro input \um@parse@range X-Y-\@marker-\@nil#1\@nil
Arguments #1-#2-#3 = X-Y-\@marker-

```

153     \else
154         \ifnum#4>\numexpr#1-1\relax
155             \ifnum#4<\numexpr#2+1\relax
156                 \@tempswatrue
157             \fi
158         \fi
159     \fi
160 \fi
161 \fi}

```

File II

STIX table data extraction

The source for the T_EX names for the very large number of mathematical glyphs are provided via Barbara Beeton's table file for the STIX project (ams.org/STIX). A version is located at `ftp://ftp.ams.org/pub/TeX/STIX/STIX-table.tex` but it's not currently up to date.

A single file is produced containing all 3298 symbols. Future optimisations might include generating various (possibly overlapping) subsets so not all definitions must be read just to redefine a small range of symbols..

```

1 #!/bin/sh
2
3 cat stix-tbl.asc |
4 awk '
5 BEGIN {OFS="|"}
6 {if (usv != substr($0,2,5) )
7     {if (substr($0,2,1) != " ")
8         {usv = substr($0,2,5);
9         texname = substr($0,84,25);
10        type = substr($0,57,1);
11        description = tolower(substr($0,233,350));
12        {if (texname ~ /[\\]/)
13            print usv, texname, type, description;}}}' - |
14 awk -F"|" '
15 ((($3 != " ") && ($3 != "F") && ($3 != "D"))) {
16     print "\\unicode@math@symbol{" "\"" $1 "\"}" $2 "{" $3 "{" $4 "}";
17 }' - |
18 sed -e ' s/{N}/{\\mathord}/ ' \
19     -e ' s/{A}/{\\mathalpha}/ ' \
20     -e ' s/{P}/{\\mathpunct}/ ' \
21     -e ' s/{B}/{\\mathbin}/ ' \
22     -e ' s/{R}/{\\mathrel}/ ' \
23     -e ' s/{L}/{\\mathop}/ ' \
24     -e ' s/{O}/{\\mathopen}/ ' \
25     -e ' s/{C}/{\\mathclose}/ ' > stix-tex.tex

```

A Documenting the NFSS

This section contains an abridged and documented version of (bits and pieces of) L^AT_EX's NFSS. Changes are mostly cosmetic and omission of irrelevant things.

A.1 Maths symbols

`\DeclareMathSymbol` #1 : Symbol, e.g., `\alpha` or 'a'
#2 : Type, e.g., `\mathalpha`
#3 : Math font name, e.g., operators
#4 : Slot, e.g., F1

```
26 \def\DeclareMathSymbol#1#2#3#4{%
```

First ensure the math font (e.g., operators) exists:

```
27 \expandafter\in@\csname sym#3\expandafter\endcsname
28 \expandafter{\group@list}%
29 \ifin@
```

Convert the slot number to two hex digits stored in `\count\z@` and `\count\tw@`, respectively:

```
30 \begingroup
31 \count\z@=#4\relax
32 \count\tw@\count\z@
33 \divide\count\z@\sixt@@n
34 \count@\count\z@
35 \multiply\count@\sixt@@n
36 \advance\count\tw@-\count@
```

The symbol to be defined can be either a command (`\alpha`) or a character (a). Branch for the former:

```
37 \if\relax\noexpand#1% is command?
38 \edef\reserved@a{\noexpand\in@{\string\mathchar}{\meaning#1}}%
39 \reserved@a
```

If the symbol command definition contains `\mathchar`, then we can provide the info that a previous symbol definition is being overwritten:

```
40 \ifin@
41 \expandafter\set@mathsymbol
42 \csname sym#3\endcsname#1#2%
43 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
44 \@font@info{Redeclaring math symbol \string#1}%
```

Otherwise, throw an error if the command name is already taken by a non-symbol definition:

```
45 \else
46 \expandafter\ifx
47 \csname\expandafter@gobble\string#1\endcsname
48 \relax
49 \expandafter\set@mathsymbol
50 \csname sym#3\endcsname#1#2%
51 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
```

```

52         \else
53         \@latex@error{Command '\string#1' already defined}\@eha
54         \fi
55     \fi

```

And if the symbol input is a character:

```

56     \else
57     \expandafter\set@mathchar
58     \csname sym#3\endcsname#1#2
59     {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}}%
60     \fi
61 \endgroup

```

Everything previous was skipped if the maths font doesn't exist in the first place:

```

62 \else
63 \@latex@error{Symbol font `#3' is not defined}\@eha
64 \fi}

```

The final macros that actually define the maths symbol with TeX primitives. If the symbol definition is for a macro:

```

65 \def\set@mathsymbol#1#2#3#4{%
66   \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}

```

Or if it's for a character:

```

67 \def\set@mathchar#1#2#3#4{%
68   \global\mathcode'#2="\mathchar@type#3\hexnumber@#1#4\relax}

```

Summary For symbols, something like:

```

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

```

For characters, something like:

```

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

```

A.2 Symbol fonts

```

\DeclareSymbolFont #1 : font name, e.g., letters
                    #2 : font encoding, e.g., OT1
                    #3 : font family, e.g., cmr
                    #4 : font series, e.g., m
                    #5 : font shape, e.g., n
69 \def\DeclareSymbolFont#1#2#3#4#5{%

```

First check that the font encoding is defined.

```

70 \@tempwafalse
71 \edef\reserved@b{#2}%
72 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
73     \ifx\reserved@b\reserved@c \@tempwattrue\fi}%
74 \cdp@list

```

So far so good. Now branch depending if this symbol font has been declared yet or not. If not, the symbol font is defined as the macro `\sym#1`; *i.e.*, for the letters symbol font, the associated command name is `\symletters`. (Funny it's not `\sym@#1`.)

```

75 \if@tempwa
76   \ifundefined{sym#1}{%
77     \expandafter\new@mathgroup\csname sym#1\endcsname
78     \expandafter\new@symbolfont\csname sym#1\endcsname{#2}{#3}{#4}{#5}%
79   }%

```

If the symbol font has been already declared:

```

80   {\@font@info{Redeclaring symbol font `#1'}%
      [Update the group list.]
81     \def\group@elt##1##2{%
82       \noexpand\group@elt\noexpand##1%
83       \expandafter\ifx\csname sym#1\endcsname##1%
84         \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
85       \else
86         \noexpand##2%
87       \fi}%
88     \xdef\group@list{\group@list}%

```

[Update the version list.]

```

89     \def\version@elt##1{%
90       \expandafter
91       \SetSymbolFont@\expandafter##1\csname#2/#3/#4/#5\expandafter
92       \endcsname \csname sym#1\endcsname
93     }%
94     \version@list
95   }%

```

If the font encoding wasn't defined, all of the above was skipped.

```

96 \else
97   \@latex@error{Encoding scheme `#2' unknown}\@eha
98 \fi}

```

`\new@symbolfont` #1 : internal symbol font name, *e.g.*, `\symletters`
 #2 : font encoding, *e.g.*, OT1
 #3 : font family, *e.g.*, cmr
 #4 : font series, *e.g.*, m
 #5 : font shape, *e.g.*, n

```

99 \def\new@symbolfont#1#2#3#4#5{%

```

Update the group list:

```

100 \toks@\expandafter{\group@list}%
101 \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
102 \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
103 \def\version@elt##1{\toks@\expandafter{##1}%
104 \edef##1{\the\toks@\noexpand\getanddefine@font%
105 #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
106 \global\advance\csname c@\expandafter
107 \gobble\string##1\endcsname\@ne
108 }%
109 \version@list}

```

\SetSymbolFont #1 : math font version, *e.g.*, normal
#2 : font name, *e.g.*, letters
#3 : font encoding, *e.g.*, OT1
#4 : font family, *e.g.*, cmr
#5 : font series, *e.g.*, m
#6 : font shape, *e.g.*, n

```

110 \def\SetSymbolFont#1#2#3#4#5#6{%
111 \@tempswafalse
112 \edef\reserved@b{#3}%
113 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
114 \ifx\reserved@b\reserved@c \@tempwattrue\fi}%
115 \cdp@list
116 \if@tempswa
117 \expandafter\SetSymbolFont@
118 \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
119 \endcsname \csname sym#1\endcsname
120 \else
121 \@latexerror{Encoding scheme `#3' unknown}\@eha
122 \fi
123 }

```

\SetSymbolFont@ #1 : internal math font version, *e.g.*, \mv@normal
#2 : NFSS font, *e.g.*, \OT1/cmr/m/n
#3 : internal symbol name, *e.g.*, \symletters

```

124 \def\SetSymbolFont@#1#2#3{%

```

If the maths version has been defined:

```

125 \expandafter\in@\expandafter#1\expandafter{\version@list}%
126 \ifin@

```

If the symbol font has been defined:

```

127 \expandafter\in@\expandafter#3\expandafter{\group@list}%
128 \ifin@
129 \begingroup
130 \expandafter\get@cdp\string#2\@nil\reserved@a
131 \toks@{}%
132 \def\install@mathalphabet##1##2{%
133 \addtohook\toks@{\install@mathalphabet##1{##2}}%

```

```

134     }%
135 \def\getanddefine@fonts##1##2{%
136   \ifnum##1=#3%
137     \addto@hook\toks@{\getanddefine@fonts#3#2}%
138     \expandafter\get@cdp\string##2\@nil\reserved@b
139     \ifx\reserved@a\reserved@b\else
140       \@font@warning{Encoding ``\reserved@b' has changed
141         to ``\reserved@a' for symbol font\MessageBreak
142         ``\expandafter\@gobblefour\string#3' in the
143         math version ``\expandafter
144         \@gobblefour\string#1'}%
145     \fi
146     \@font@info{%
147       Overwriting symbol font
148       ``\expandafter\@gobblefour\string#3' in
149       version ``\expandafter
150       \@gobblefour\string#1'\MessageBreak
151       \@spaces \expandafter\@gobble\string##2 -->
152       \expandafter\@gobble\string#2}%
153   \else
154     \addto@hook\toks@{\getanddefine@fonts##1##2}%
155   \fi}%
156 #1%
157 \xdef#1{\the\toks@}%
158 \endgroup

```

If the symbol font wasn't defined, all of the above was skipped:

```

159 \else
160   \@latex@error{Symbol font ``\expandafter\@gobblefour\string#3'
161     not defined}\@eha
162 \fi

```

If the maths version wasn't defined, all of the above was skipped:

```

163 \else
164   \@latex@error{Math version ``\expandafter\@gobblefour\string#1'
165     is not
166     defined}{You probably misspelled the name of the math
167     version.^^JOr you have to specify an additional package.}%
168 \fi}

```