The `unicodefonttable` package*

Frank Mittelbach

Abstract

A package for typesetting font tables for larger fonts, e.g., TrueType or OpenType Unicode fonts. To produce a one-off table, a standalone version is available as well.

Contents

1 Introduction

When I started to write a new chapter for the third edition of *The LaTeX Companion* on modern fonts available for different LaTeX engines, I was a bit surprised that I couldn't find a way to easily typeset tables showing the glyphs available in TrueType or OpenType fonts. The nfssfont package available with LaTeX only supports fonts from the 8-bit world, but modern fonts that can be used with XeTeX or LuaTeX can contain thousands of glyphs and having a method to display what is available in them was important for me.

I therefore set out to write my own little package and what started as an afternoon exercise ended up being this package, offering plenty of bells and whistles for typesetting such font tables.

* This is version 1.0k of the package, dated 2025/07/11; the license is LPPL.

As there can be many glyphs in such fonts a tabular representation of them might run for several pages, so the package internally uses the longtable package to handle that.

In most cases the glyphs inside the fonts are indexed by their Unicode numbers so it is natural to display them sorted by their position in the Unicode character set.

Unicode is organized in named blocks such as "Basic Latin", "Latin-1 Supplement", etc., typically consisting of 265 characters each.[1] It is therefore helpful to use these block names as subtitles within the table, to more easily find the information one is looking for.

A common way to represent the number of a single Unicode character is `U+` followed by four (or more) hexadecimal digits. For example, `U+0041` represents the letter "A" and `U+20AC` the Euro currency symbol "€". We use this convention by showing a Unicode range of sixteen characters at the left of each table row, e.g., `U+0040 - 004F`, followed by the sixteen glyphs in the range. Thus that particular table row from the "Basic Latin" block would show something like

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0040 - 004F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

If a Unicode character has no glyph representation in a given font then this is indicated by a special symbol (by default a colored hyphen). By default some color is used, but we've grayscaled the output for *TUGboat*.

In order to easily locate any Unicode character the table shows by default sixteen hex digits as a column heading. For example, to find Euro currency symbol (`U+20AC`) one first finds the right row, which is the range `U+20A0 - 20AF`, and then the `C` column in that row, and the glyph is there (or an indication that the font is missing that glyph; the line shows that for some of the other slots).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+20A0 - 20AF	-	₡	-	-	£	-	₦	-	-	₩	-	₫	€	-	-	-

It can be useful to compare two fonts with each other by filling the table with glyphs from a secondary font if the primary font is missing them. For example, the next display shows two rows of Latin Modern Math (black glyphs) and instead of showing a missing glyph symbol in most slots, we use the glyphs from New Computer Modern Math, which has a much larger glyph set (normally red glyphs with gray background but again, grayscaled for *TUGboat*).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+2A00 - 2A0F	⨀	⨁	⨂	⨃	⨄	⨅	⨆	⨇	⨈	⨉	⨊	⨋	⨌	⨍	⨎	⨏
U+2A10 - 2A1F	⨐	⨑	⨒	⨓	⨔	⨕	⨖	⨗	⨘	⨙	⨚	⨛	⨜	⨝	⨞	⨟

2 The user interface

The package offers one command to typeset a font table. The appearance of the table can be customized by specifying key/value pairs.

[1] Some blocks are smaller, while those containing the Asian ideographs are much larger.

\displayfonttable `\displayfonttable * [⟨key/value-list⟩] {⟨font-name⟩} [⟨font-features⟩]`

The ⟨*font-name*⟩ is the font to be displayed. This and the ⟨*font-features*⟩ argument are passed to fontspec, thus they should follow the conventions of that package for specifying a font. The ⟨*key/value-list*⟩ offers customization possibilities discussed below.

The `\displayfonttable*` is a variant of the command, intended for use with 8-bit legacy fonts. It presets some keys, but otherwise behaves identically. The preset values are:

```
nostatistics, display-block=none, hex-digits=head, range-end=FF
```

For details see the next section.

\fonttablesetup `\fonttablesetup {⟨key/value-list⟩}`

Instead of or in addition to specifying key/values to `\displayfonttable` it is possible to set them up as defaults. Inside `\displayfonttable` the defaults are applied first, so one can still overwrite their values for an individual table.

\fonttableglyphcount `\fonttableglyphcount`

While typesetting a font table the package keeps track of the number of glyphs it finds in the font. After the table has finished, this value is available in `\fonttableglyphcount` and it is, for example, used when statistics are produced. At the start of the next table it is reset to zero.

2.1 Keys and their values

Several of the available keys are booleans accepting `true` or `false`. They usually exist in pairs so that one can specify the desired behavior without needing to provide a value, e.g., specifying `header` is equivalent to specifying `header=true` or `noheader=false`, etc. In the lists below the default settings are indicated by an underline.

`header`
`noheader`
`title-format`
`title-format-cont`

The first set of keys is concerned with the overall look and feel of the generated table.

`header`, `noheader` These keys determine whether a header to the table is produced.

`title-format`, `title-format-cont` These keys define what is provided as a header title or continuation title if the table consists of several pages. They expect code as their value. This code can contain `#1` and `#2` to denote the ⟨*font-name*⟩ and ⟨*font-features*⟩ arguments, respectively.

By default a title using the `\caption` command is produced; on continuation titles, the ⟨*font-features*⟩ are not shown. This is typeset as a `longtable` header row, so you either need to use `\multicolumn` or a `\caption` command — otherwise everything ends up in the first column.

display-block
hex-digits
hex-digits-font
hex-digits-row-format
color

These keys handle the inner parts of the table.

display-block The Unicode dataset is organized in named blocks that are typically 128 or 256 characters, though some are noticeably larger and a few are smaller. With the **display-block** key it is possible to specify if and how such blocks should be made visible. The following values are supported:

- **titles** Above each display block that contains glyphs the Unicode title of the block is displayed.
- **rules** Display blocks are indicated only by a `\midrule`.
- **none** Display blocks are not indicated at all.

hex-digits To ease reading the table, rows of hex digits are added to it. Where or if this happens is controlled by this key. Allowed values for it are the following:

- **block** A row of hex digits is placed at the beginning of each Unicode block containing glyphs in the displayed font.
- **foot** A row is added to the foot of each table page.
- **head** A row is added to the top of each table page.
- **head+foot** A row is added to the top and the foot of each table page.
- **none** All hex digit rows are suppressed.

hex-digits-font The font to use for the hex digits, by default `\ttfamily\scriptsize`.

hex-digits-row-format This key defines the format for the hex digits shown on the left of each row. It accepts one argument hold the hex values for the row except for the last digit, e.g, `0A3` for the values from `0A30` to `0A3F`. The default formatting is `U+#10\,-\,#1F` and without further adjustments it is automaticaly set in `\ttfamily\footnotesize` and in the color specified by the `color` key. A suitable value that takes up less space would be `U+#1x`.

color This key determines the color for parts of the table (hex digits and Unicode ranges). It can be either `none` or a color specification as understood by the `\color` command. The default is `blue`.

statistics
nostatistics
statistics-font
statistics-format

The next set of keys allows altering the statistics that are produced.

statistics, nostatistics These keys determine whether some statistics are listed at the end of the table.

statistics-font The font used to typeset the statistics; the default is `\normalfont\small`.

statistics-format Code (text) to specify what should be typeset in the statistics. One can use `#1` for the ⟨*font-name*⟩ and `#2` for the glyph count. The material is typeset on a single line at the end of the table. If several lines are needed you need to use `\parbox` or a similar construct.

`glyph-width`
`missing-glyph`
`missing-glyph-font`
`missing-glyph-color`

Another set of keys deals with customization on the glyph level.

`glyph-width` All glyphs are typeset in a box with the same width, the default value is `6pt` which is suitable for most 10pt fonts and make the table fit comfortably into the text width of a typical document.

`missing-glyph` If a slot in a row doesn't have a glyph in the font you may still want display something to indicate this state. By giving the key a value any arbitrary glyph or material can be typeset. The default is to typeset a `-` (hyphen) in a special color.

Rows that contain no glyph whatsoever are not displayed at all. Instead a small vertical space is added to indicate the one or more rows are omitted.

`missing-glyph-font` The font used for the missing glyphs (the default value is `\ttfamily\scriptsize`).

`missing-glyph-color` If not specified it uses the value specified with the `color` key. If you want a different color, e.g., `red`, you can use a color value or you can specify `none` to use no coloring.

`compare-with`
`compare-color`
`compare-bgcolor`
`statistics-compare-format`

You can make comparisons between two fonts, which is useful, for example when dealing with incomplete math fonts and you need to see how well the symbols from one font blend with the supplementary symbols from another font.

`compare-with` If given, the value is a ⟨*comparison-font-name*⟩ that is used to supply missing glyphs. This means that if the ⟨*font-name*⟩ to be displayed is missing a glyph in a slot, then the ⟨*comparison-font-name*⟩ is checked, and if that font has the glyph in question, it will be displayed instead of showing a missing glyph indicator.

`compare-color`, `compare-bgcolor` To distinguish real glyphs from missing but substituted glyphs, they can be colored specially (default `red`) and/or you can have their background colored (default is `black!10`, i.e., a light gray).

`statistics-compare-format` Code (text) to specify what should be typeset in the statistics when comparing two fonts. One can use `#1` for the ⟨*font-name*⟩ and `#2` for its glyph count, `#3` is the name of the comparison font, `#4` its glyph count, `#5` for the number of glyphs missing in this font and `#6` the number of extra glyphs in it. This code is used instead of `statistics-format` when comparisons are made.

The material is typeset on a single line at the end of the table. If several lines are needed you need to use `\parbox` or a similar construct.

range-start
range-end

Finally there are two keys for restricting the display range.

range-start, range-end The full Unicode set of characters is huge and checking every slot to see if the current font contains a glyph in the slot takes a long time. If you know that font contains only a certain subset then you can speed up the table generation considerably by limiting the search (and consequently the output generation). The `range-start` specifies where to start with the search (default `0000`) and `range-end` gives the last slot that is tested (default `FFFF`).

Thus, by default we restrict the display to slots below `10000`, because text fonts seldom contain glyphs in the higher planes. But if you want to see everything of the font (as far as supported by this package) and are prepared to wait for the higher planes to be scanned, you can go up to a value of `FFFFF`.

However, please note that the LuaTEX fontloader uses the "Supplementary Private Use Area-A", which starts at `F0000`, as its own playground and places remapping into it, so by default you see random data instead of font data there. You either have to use the XeTEX engine or load the font with `Renderer=HarfBuzz` in LuaTEX.

These keys are also quite useful in combination with the previous `compare-with` key, to display only, for example, the Greek letters and see how glyphs from two fonts blend with each other.

2.2 A standalone interactive version

If you want to quickly display a single font, you can run `unicodefont.tex` through LaTeX using LuaTEX (or XeTEX) as the engine. Similar to `nfssfont.tex` (which is for 8-bit fonts with pdfTEX) it asks you a few questions and then generates the font table for you. There are fewer configuration options available, but this workflow saves you writing a document to get a one-off table.

Most font tables need several runs due to the use of longtable, which has to find the right width for the columns across several pages. The `unicodefont` file therefore remembers your selection from the previous run and asks you if you want to reapply it to speed up the process.

3 Notes on the table data

If you look at some parts of a Unicode font table you see a number of slots that do not show a "missing glyph" sign, but nonetheless appear to be empty. For example:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0020 – 002F		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
U+0030 – 003F	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
U+0040 – 004F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
U+0050 – 005F	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
U+0060 – 006F	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
U+0070 – 007F	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	-
U+00A0 – 00AF		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
U+00B0 – 00BF	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿

The reason is that Unicode contains a lot of special spaces or otherwise invisible characters, e.g., `U+0020` is the normal space, `U+00A0` is a non-breaking space, `U+00AD` is a soft-hyphen (what LaTeX users would indicate with `\-`), and so forth. Especially the row `U+2000-200F` in Table 6 looks strange as it appears to be totally empty, but in fact most of its slots contain spaces of different width.

General Punctuation

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+2000 - 200F															-	-
U+2010 - 201F	-	-	‒	–	—	―	‖	=	‘	’	,	-	“	”	„	-
U+2020 - 202F	†	‡	•	-	-	-	…	-	-	-	-	-	-	-	-	

Another somewhat surprising area is the “Mathematical Alphanumeric Symbols” block in math fonts, starting at `U+1D400`. There you see a number of missing characters, the first two being `U+1D455` (math italic small h) and `U+1D49D` (math script B).

Mathematical Alphanumeric Symbols

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+1D400 - 1D40F	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
U+1D410 - 1D41F	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**a**	**b**	**c**	**d**	**e**	**f**
U+1D420 - 1D42F	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**
U+1D430 - 1D43F	**w**	**x**	**y**	**z**	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
U+1D440 - 1D44F	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*a*	*b*
U+1D450 - 1D45F	*c*	*d*	*e*	*f*	*g*	-	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*
U+1D460 - 1D46F	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*	***A***	***B***	***C***	***D***	***E***	***F***	***G***	***H***
U+1D470 - 1D47F	***I***	***J***	***K***	***L***	***M***	***N***	***O***	***P***	***Q***	***R***	***S***	***T***	***U***	***V***	***W***	***X***
U+1D480 - 1D48F	***Y***	***Z***	***a***	***b***	***c***	***d***	***e***	***f***	***g***	***h***	***i***	***j***	***k***	***l***	***m***	***n***
U+1D490 - 1D49F	***o***	***p***	***q***	***r***	***s***	***t***	***u***	***v***	***w***	***x***	***y***	***z***	$\mathcal{A}$	-	$\mathcal{C}$	$\mathcal{D}$
U+1D4A0 - 1D4AF	-	-	$\mathcal{G}$	-	-	$\mathcal{J}$	$\mathcal{K}$	-	-	$\mathcal{N}$	$\mathcal{O}$	$\mathcal{P}$	$\mathcal{Q}$	-	$\mathcal{S}$	$\mathcal{T}$

In this case the reason is *not* that the font fails to implement the characters, but that these characters have already been defined in earlier revisions of the Unicode standard in the lower Unicode plane. For example, the “h” is the Planck constant `U+210E` and `U+212C` is the script capital B, etc. The Unicode Consortium decided not to encode the *same* character twice, hence the apparent holes.

4 Examples

In this section we show the results of a few calls to `\displayfonttable`. The tables are a bit easier to navigate if they use color in some places, but for *TUGboat* this is not practical, so we use black and gray.

Please note that this documentation was produced with LuaTEX. If you reuse the examples with XETEX, you may have to specify the font names differently (i.e., following to the fontspec documentation for this engine).

4.1 Computer Modern Sans — 7-bit font

Our first example is the original Computer Modern Sans, with character codes ≤ 127.

Command used:

```
\displayfonttable*[color=none, range-end=7F]{cmss10}
```

Table 7: cmss10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0000 - 000F	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl
U+0010 - 001F	ı	ȷ	`	´	ˇ	˘	¯	°	¸	ß	æ	œ	ø	Æ	Œ	Ø
U+0020 - 002F	˗	!	”	#	$	%	&	’	(	)	*	+	,	-	.	/

Table 7: cmss10 *cont.*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0030 - 003F	0	1	2	3	4	5	6	7	8	9	:	;	¡	=	¿	?
U+0040 - 004F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
U+0050 - 005F	P	Q	R	S	T	U	V	W	X	Y	Z	[	“	]	^	·
U+0060 - 006F	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
U+0070 - 007F	p	q	r	s	t	u	v	w	x	y	z	–	—	˝	~	¨

4.2 TEX Gyre Heros — 8-bit font

This example shows the TEX Gyre Heros 8-bit font, in the T1 encoding, with character codes ≤ 255. We used `hex-digits-row-format` to shorten the row titles on the left:

```
\displayfonttable*[color=none,hex-digits-row-format=U+#1]{ec-qhvr}
```

Table 8: ec-qhvr

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+000	`	´	ˆ	˜	¨	˝	˚	ˇ	˘	¯	˙	¸	˛	,	‹	›
U+001	“	”	„	«	»	–	—		₀	ı	ȷ	ff	fi	fl	ffi	ffl
U+002	␣	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
U+003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
U+004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
U+005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
U+006	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
U+007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	-
U+008	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
U+009	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§
U+00A	ă	ą	ć	č	ď	ě	ę	ğ	í	ľ	ł	ń	ň	ŋ	ő	ŕ
U+00B	ř	ś	š	ş	ť	ţ	ű	ů	ÿ	ź	ž	ż	ij	¡	¿	£
U+00C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
U+00D	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS
U+00E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
U+00F	ð	ñ	ò	ó	ô	õ	ö	œ	ø	ù	ú	û	ü	ý	þ	ß

4.3 Latin Modern Math — 8-bit fonts

The traditional Latin Modern Math Italic, Symbol and Extension fonts. The symbol font (`lmsy10`) has two characters added to the Computer Modern symbol repertoire, seen in the last row of the table. Commands used:

```
\displayfonttable*[color=none]{lmmi10}
\displayfonttable*[color=none]{lmsy10}
\displayfonttable*[color=none]{lmex10}
```

Table 9: lmmi10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0000 - 000F	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	α	β	γ	δ	ϵ
U+0010 - 001F	ζ	η	θ	ι	κ	λ	μ	ν	ξ	π	ρ	σ	τ	υ	ϕ	χ
U+0020 - 002F	ψ	ω	ε	ϑ	ϖ	ϱ	ς	φ	$\leftharpoonup$	$\leftharpoondown$	$\rightharpoonup$	$\rightharpoondown$	‘	’	$\triangleright$	$\triangleleft$
U+0030 - 003F	0	1	2	3	4	5	6	7	8	9	.	,	<	/	>	$\star$
U+0040 - 004F	∂	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
U+0050 - 005F	P	Q	R	S	T	U	V	W	X	Y	Z	$\flat$	$\natural$	$\sharp$	$\smile$	$\frown$
U+0060 - 006F	ℓ	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o

Table 9: lmmi10 *cont.*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0070 - 007F	p	q	r	s	t	u	v	w	x	y	z	$\imath$	$\jmath$	$\wp$	$\vec{}$	$\frown$

Table 10: lmsy10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0000 - 000F	$-$	$\cdot$	$\times$	$*$	$\div$	$\diamond$	$\pm$	$\mp$	$\oplus$	$\ominus$	$\otimes$	$\oslash$	$\odot$	$\bigcirc$	$\circ$	$\bullet$
U+0010 - 001F	$\asymp$	$\equiv$	$\subseteq$	$\supseteq$	$\leq$	$\geq$	$\preceq$	$\succeq$	$\sim$	$\approx$	$\subset$	$\supset$	$\ll$	$\gg$	$\prec$	$\succ$
U+0020 - 002F	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$	$\leftrightarrow$	$\nearrow$	$\searrow$	$\simeq$	$\Leftarrow$	$\Rightarrow$	$\Uparrow$	$\Downarrow$	$\Leftrightarrow$	$\nwarrow$	$\swarrow$	$\propto$
U+0030 - 003F	$\prime$	∞	$\in$	$\ni$	$\triangle$	$\triangledown$	$/$	$-$	$\forall$	$\exists$	$\neg$	$\emptyset$	$\Re$	$\Im$	$\top$	$\bot$
U+0040 - 004F	$\aleph$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{O}$
U+0050 - 005F	$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{R}$	$\mathcal{S}$	$\mathcal{T}$	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{W}$	$\mathcal{X}$	$\mathcal{Y}$	$\mathcal{Z}$	$\cup$	$\cap$	$\uplus$	$\wedge$	$\vee$
U+0060 - 006F	$\vdash$	$\dashv$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$	$\langle$	$\rangle$	$\vert$	$\Vert$	$\updownarrow$	$\Updownarrow$	$\backslash$	$\wr$
U+0070 - 007F	$\surd$	$\amalg$	∇	$\int$	$\sqcup$	$\sqcap$	$\sqsubseteq$	$\sqsupseteq$	$\S$	$\dagger$	$\ddagger$	$\P$	$\clubsuit$	$\diamondsuit$	$\heartsuit$	$\spadesuit$
U+00A0 - 00AF	-	-	-	-	-	-	-	-	-	-	-	-	$\leqslant$	$\geqslant$	-	-

Table 11: lmex10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0000 - 000F	$($	$)$	$[$	$]$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$	$\langle$	$\rangle$	$\vert$	$\Vert$	$/$	$\backslash$
U+0010 - 001F	$($	$)$	$($	$)$	$[$	$]$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$	$\langle$	$\rangle$	$/$	$\backslash$
U+0020 - 002F	$($	$)$	$[$	$]$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$	$\langle$	$\rangle$	$/$	$\backslash$	$/$	$\backslash$
U+0030 - 003F	$($	$)$	$[$	$]$	$\lfloor$	$\rfloor$	$\vert$	$\vert$	$($	$)$	$($	$)$	$\{$	$\}$	$\vert$	$\vert$
U+0040 - 004F	$\backslash$	$)$	$\vert$	$\vert$	$\langle$	$\rangle$	$\sqcup$	$\bigsqcup$	$\oint$	$\oint$	$\odot$	$\odot$	$\oplus$	$\bigoplus$	$\otimes$	$\bigotimes$
U+0050 - 005F	$\sum$	$\prod$	$\int$	$\cup$	$\cap$	$\uplus$	$\wedge$	$\vee$	$\sum$	$\prod$	$\int$	$\bigcup$	$\bigcap$	$\biguplus$	$\bigwedge$	$\bigvee$
U+0060 - 006F	$\coprod$	$\coprod$	$\widehat{}$	$\widehat{}$	$\widehat{}$	$\widetilde{}$	$\widetilde{}$	$\widetilde{}$	$[$	$]$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$
U+0070 - 007F	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\vert$	$\lceil$	$\Vert$	$\uparrow$	$\downarrow$	$\overbrace{}$	$\underbrace{}$	$\overbrace{}$	$\underbrace{}$	$\Uparrow$	$\Downarrow$

4.4 Latin Modern Math compared to New Computer Modern Math

This example shows the extra symbols available in New Computer Modern Math in comparison to Latin Modern Math as the base font. We use the following setup (including settings for the grayscaled *TUGboat* output, as an example of color overrides):

```
\displayfonttable[hex-digits=head+foot, range-end=1FFFF,
                  compare-with=New Computer Modern Math,
                  title-format=\caption{Latin Modern Math compared to
                                        New Computer Modern Math},
                  title-format-cont=\caption{LM Math vs.\ NewCM Math,
                                        \emph{cont.}},
```

```
compare-color=black, compare-bgcolor=black!5,
missing-glyph-color=black!50, color=black!75]
{Latin Modern Math}
```

That is, glyphs only in `NewCM` are shown with a light gray background.

We also extended the range to cover `U+10000` to `U+1FFFF` in order to include the Unicode Math alphabets.

Table 12: Latin Modern Math compared to New Computer Modern Math

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Basic Latin																
U+0020 – 002F		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
U+0030 – 003F	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
U+0040 – 004F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
U+0050 – 005F	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
U+0060 – 006F	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
U+0070 – 007F	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
Latin-1 Supplement																
U+00A0 – 00AF		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
U+00B0 – 00BF	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
U+00C0 – 00CF	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
U+00D0 – 00DF	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
U+00E0 – 00EF	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
U+00F0 – 00FF	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
Latin Extended-A																
U+0100 – 010F	Ā	ā	Ă	ă	Ą	ą	Ć	ć	Ĉ	ĉ	Ċ	ċ	Č	č	Ď	ď
U+0110 – 011F	Đ	đ	Ē	ē	Ĕ	ĕ	Ė	ė	Ę	ę	Ě	ě	Ĝ	ĝ	Ğ	ğ
U+0120 – 012F	Ġ	ġ	Ģ	ģ	Ĥ	ĥ	Ħ	ħ	Ĩ	ĩ	Ī	ī	Ĭ	ĭ	Į	į
U+0130 – 013F	İ	ı	IJ	ij	Ĵ	ĵ	Ķ	ķ	ĸ	Ĺ	Í	Ļ	ļ	Ľ	ľ	Ŀ
U+0140 – 014F	ŀ	Ł	ł	Ń	ń	Ņ	ņ	Ň	ň	ʼn	Ŋ	ŋ	Ō	ō	Ŏ	ŏ
U+0150 – 015F	Ő	ő	Œ	œ	Ŕ	ŕ	Ŗ	ŗ	Ř	ř	Ś	ś	Ŝ	ŝ	Ş	ş
U+0160 – 016F	Š	š	Ţ	ţ	Ť	ť	Ŧ	ŧ	Ũ	ũ	Ū	ū	Ŭ	ŭ	Ů	ů
U+0170 – 017F	Ű	ű	Ų	ų	Ŵ	ŵ	Ŷ	ŷ	Ÿ	Ź	ź	Ż	ż	Ž	ž	ſ
Latin Extended-B																
U+0180 – 018F	ƀ	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
U+01A0 – 01AF	Ơ	ơ	–	–	–	–	–	–	–	–	–	–	–	–	–	Ư
U+01B0 – 01BF	ư	–	–	–	–	Ƶ	–	–	–	–	–	–	–	–	–	–
U+0210 – 021F	–	–	–	–	–	–	–	–	Ș	ș	Ț	ț	–	–	–	–
U+0230 – 023F	–	–	–	–	–	–	–	ȷ	–	–	–	–	–	–	–	–
Spacing Modifier Letters																
U+02C0 – 02CF	–	–	–	–	–	–	ˆ	ˇ	–	–	–	–	–	–	–	–
U+02D0 – 02DF	–	–	–	–	–	–	–	–	˘	˙	˚	˛	˜	˝	–	–
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Table 12: LM Math vs. NewCM Math, *cont.*

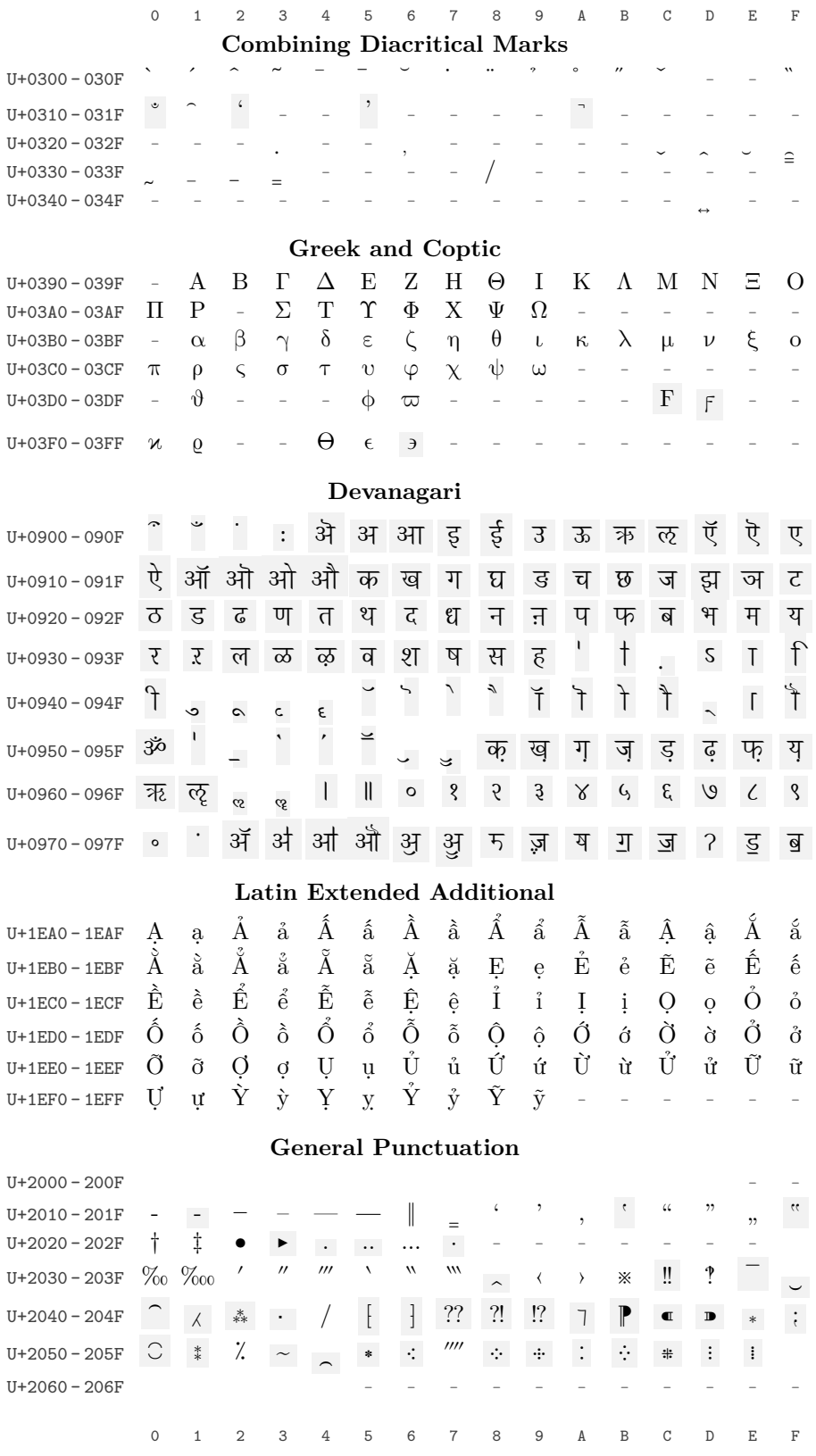


Table 12: LM Math vs. NewCM Math, *cont.*

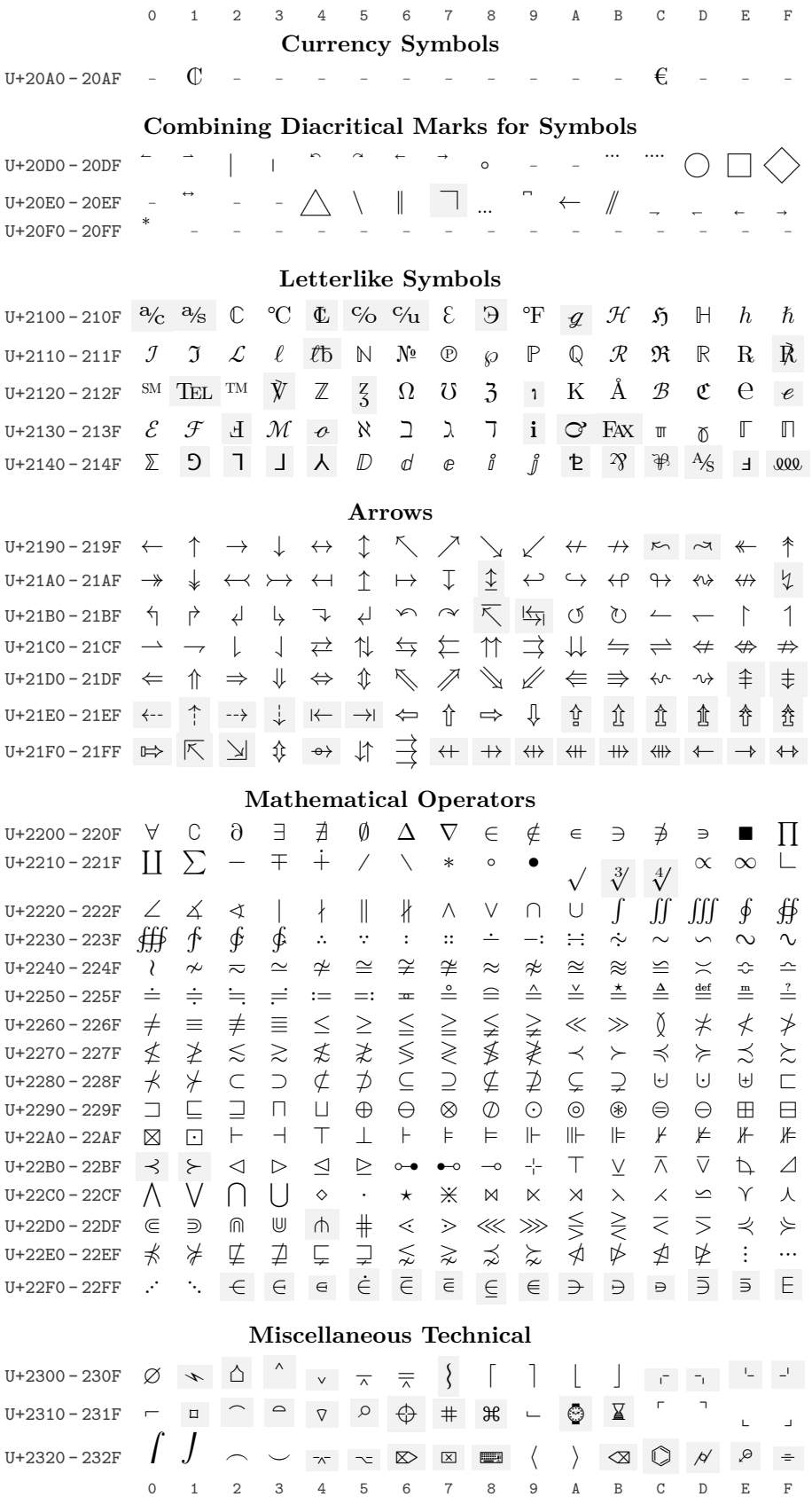


Table 12: LM Math vs. NewCM Math, *cont.*

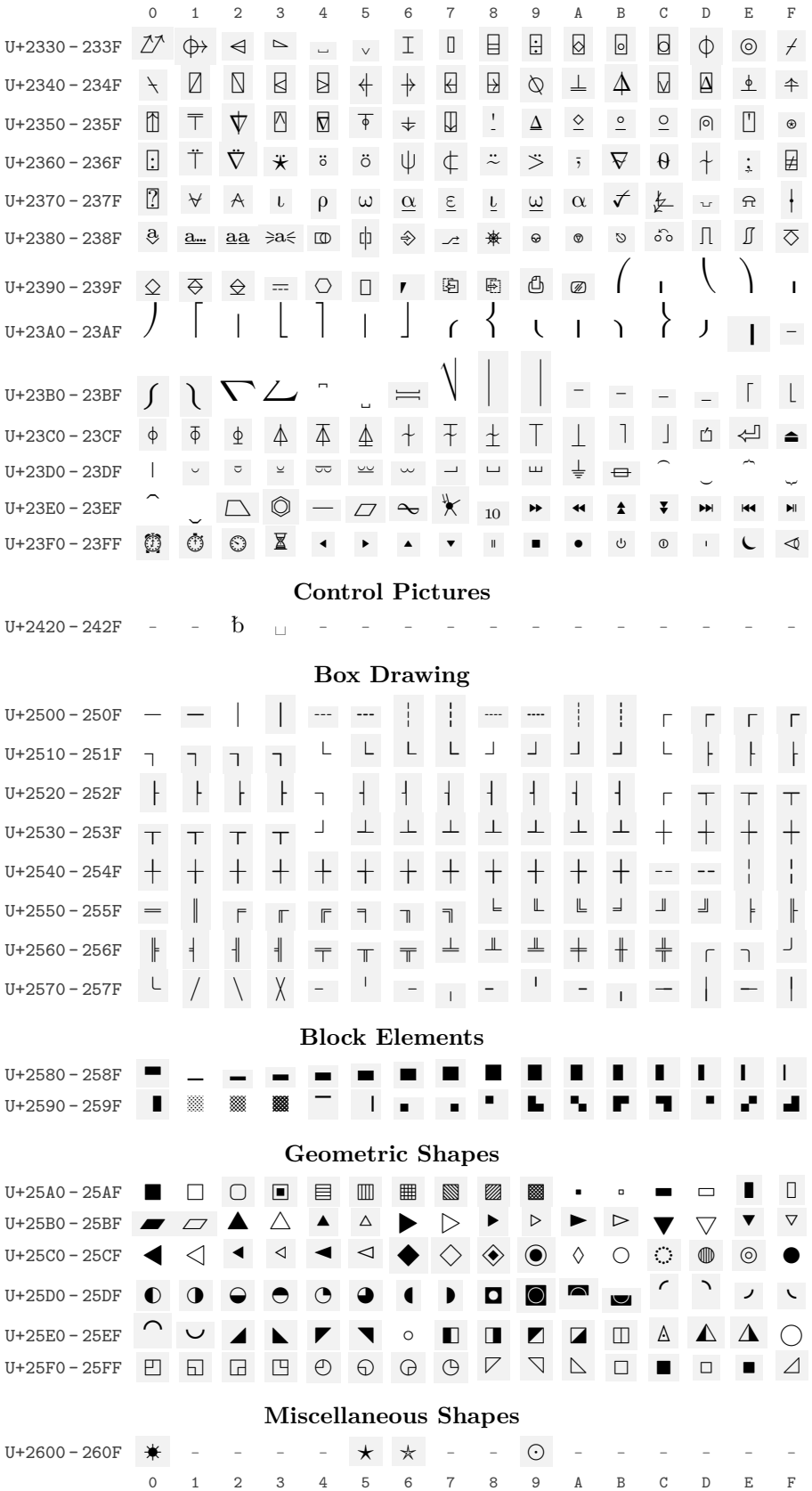


Table 12: LM Math vs. NewCM Math, *cont.*

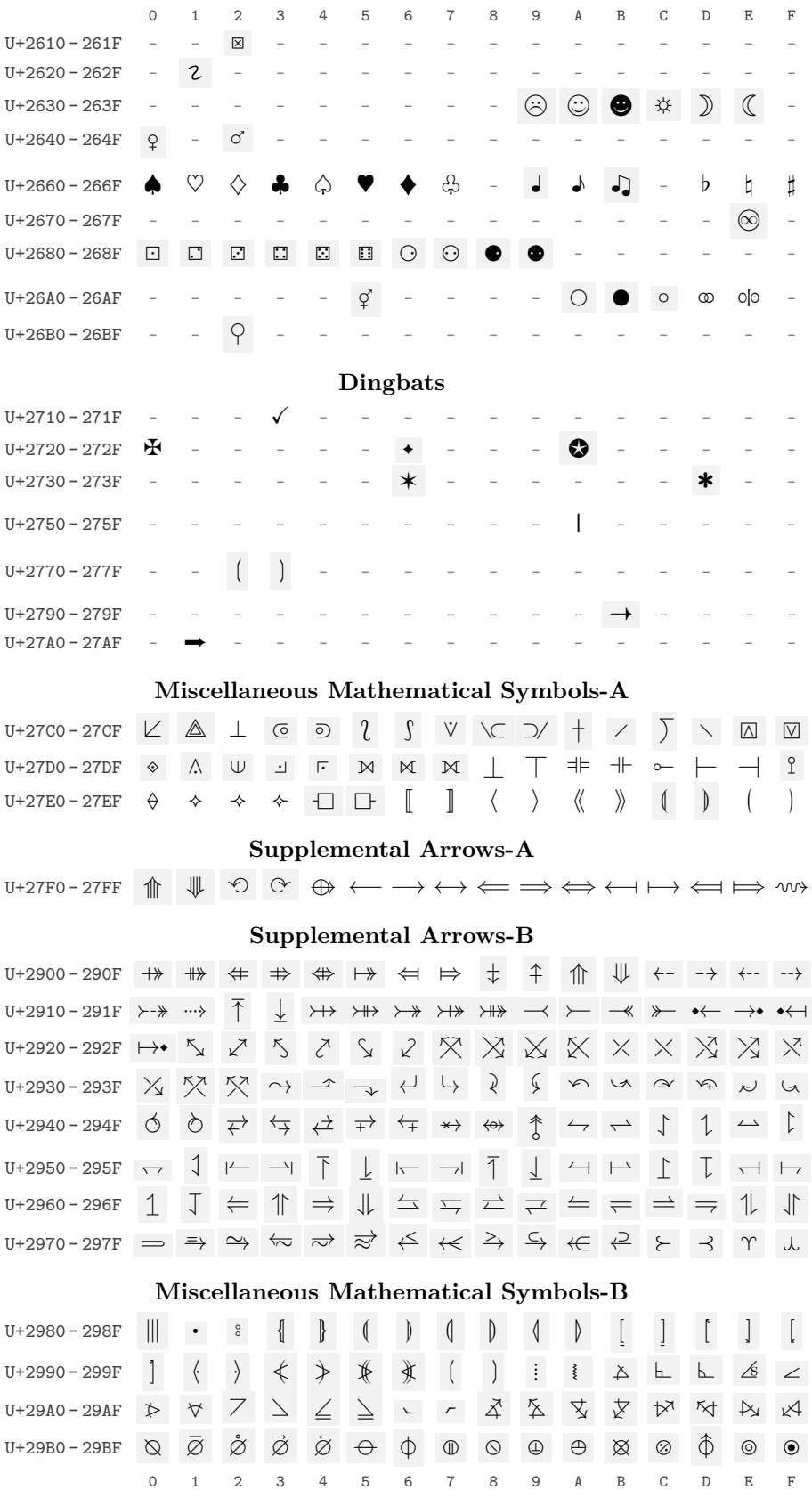


Table 12: LM Math vs. NewCM Math, *cont.*

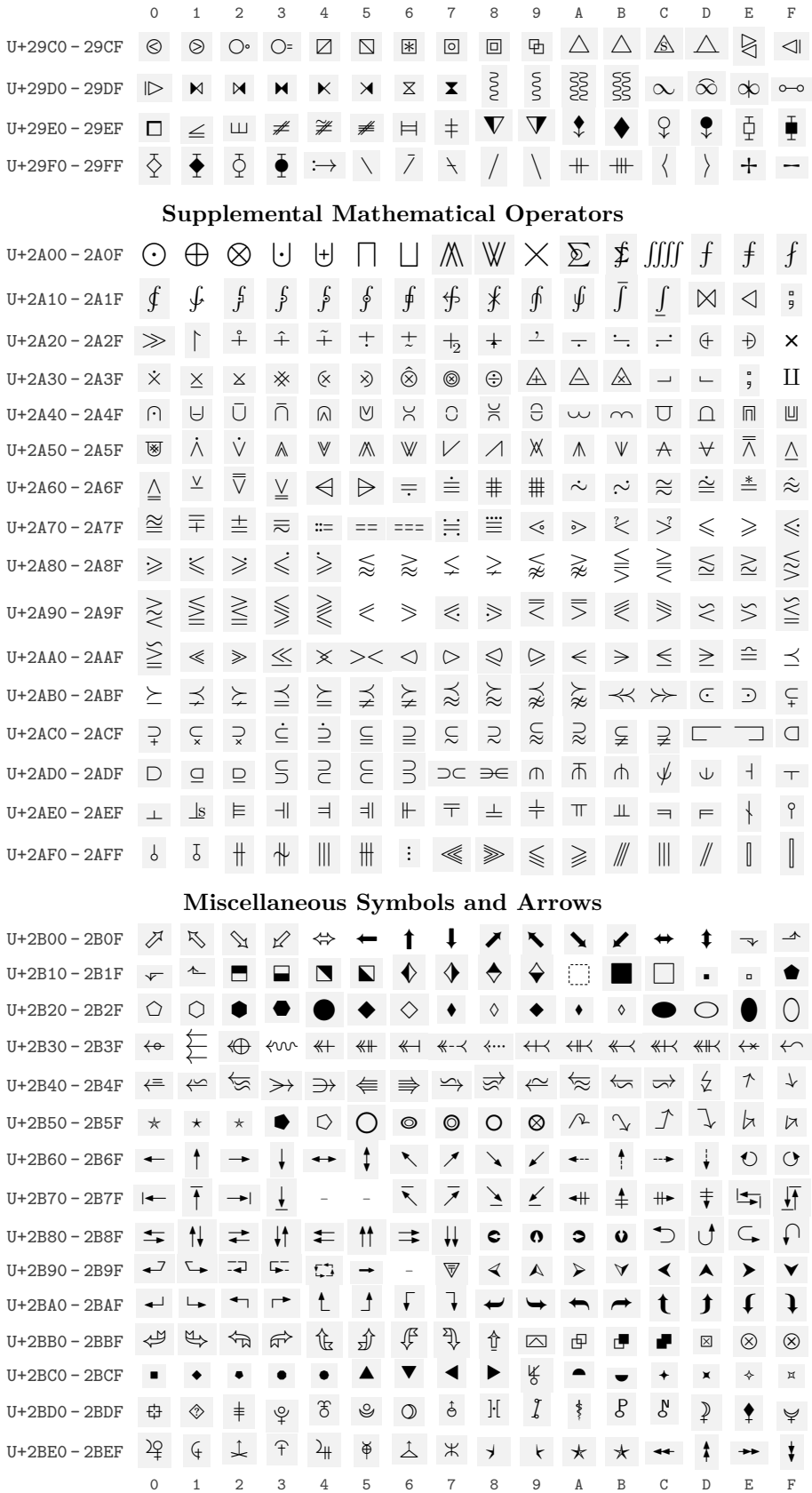


Table 12: LM Math vs. NewCM Math, *cont.*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+2BF0 – 2BFF	⯰	⯱	⯲	⯳	⯴	⯵	⯶	⯷	⯸	⯹	⯺	⯻	⯼	⯽	⯾	⯿

Supplemental Punctuation

U+2E10 – 2E1F	–	–	–	–	–	–	–	–	⸮	–	–	–	–	–	–	–

CJK Symbols and Punctuation

U+3010 – 301F	–	–	〒	–	–	–	〖	〗	–	–	–	–	–	–	–	–
U+3030 – 303F	〰	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Private Use Area

U+E000 – E00F	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π
U+E010 – E01F	Ρ	Σ	Τ	ϒ	Φ	Χ	Ψ	Ω	α	β	γ	δ	ε	ζ	η	θ
U+E020 – E02F	ι	κ	λ	μ	ν	ξ	ο	π	ρ	ς	σ	τ	υ	φ	χ	ψ
U+E030 – E03F	ω	ϵ	⫻	⫽	⤳	⤷	✳	✳	✳	✳	↱	↘	↘	↗	–	–
U+E040 – E04F	–	*Α*	*Β*	*Γ*	*Δ*	*Ε*	*Ζ*	*Η*	*Θ*	*Ι*	*Κ*	*Λ*	*Μ*	*Ν*	*Ξ*	*Ο*
U+E050 – E05F	*Π*	*Ρ*	*Σ*	*Τ*	*ϒ*	*Φ*	*Χ*	*Ψ*	*Ω*	*α*	*β*	*γ*	*δ*	*ε*	*ζ*	*η*
U+E060 – E06F	*θ*	*ι*	*κ*	*λ*	*μ*	*ν*	*ξ*	*ο*	*π*	*ρ*	*ς*	*σ*	*τ*	*υ*	*φ*	*χ*
U+E070 – E07F	*ψ*	*ω*	*ϵ*	–	–	–	–	–	–	–	–	–	–	–	–	–
U+E370 – E37F	–	–	–	–	–	–	⨜	⨝	–	–	–	–	–	–	–	–
U+E390 – E39F	–	–	–	–	–	⨛	–	⨟	⨠	⨡	⨢	⨣	–	–	–	–
U+E3D0 – E3DF	–	–	–	⨍	–	–	–	–	–	–	–	–	–	–	–	–
U+EA50 – EA5F	–	–	–	–	–	–	–	⨏	–	–	–	–	–	–	–	–

Alphabetic Presentation Forms

U+FB00 – FB0F	ff	fi	fl	ffi	ffl	–	–	–	–	–	–	–	–	–	–	–

Variation Selectors

U+FE00 – FE0F			–	–	–	–	–	–	–	–	–	–	–	–	–	–

Arabic Presentation Forms-B

U+FEF0 – FEFF	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

Mathematical Alphanumeric Symbols

U+1D400 – 1D40F	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
U+1D410 – 1D41F	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**a**	**b**	**c**	**d**	**e**	**f**
U+1D420 – 1D42F	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**
U+1D430 – 1D43F	**w**	**x**	**y**	**z**	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
U+1D440 – 1D44F	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*a*	*b*
U+1D450 – 1D45F	*c*	*d*	*e*	*f*	*g*	–	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*
U+1D460 – 1D46F	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*	***A***	***B***	***C***	***D***	***E***	***F***	***G***	***H***
U+1D470 – 1D47F	***I***	***J***	***K***	***L***	***M***	***N***	***O***	***P***	***Q***	***R***	***S***	***T***	***U***	***V***	***W***	***X***
U+1D480 – 1D48F	***Y***	***Z***	***a***	***b***	***c***	***d***	***e***	***f***	***g***	***h***	***i***	***j***	***k***	***l***	***m***	***n***
U+1D490 – 1D49F	***o***	***p***	***q***	***r***	***s***	***t***	***u***	***v***	***w***	***x***	***y***	***z***	𝒜	–	𝒞	𝒟
U+1D4A0 – 1D4AF	–	–	𝒢	–	–	𝒥	𝒦	–	–	𝒩	𝒪	𝒫	𝒬	–	𝒮	𝒯
U+1D4B0 – 1D4BF	𝒰	𝒱	𝒲	𝒳	𝒴	𝒵	𝒶	𝒷	𝒸	𝒹	–	𝒻	–	𝒽	𝒾	𝒿
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Table 12: LM Math vs. NewCM Math, *cont.*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+1D4C0 – 1D4CF	𝓀	𝓁	𝓂	𝓃	–	𝓅	𝓆	𝓇	𝓈	𝓉	𝓊	𝓋	𝓌	𝓍	𝓎	𝓏
U+1D4D0 – 1D4DF	𝒜	ℬ	𝒞	𝒟	ℰ	ℱ	𝒢	ℋ	ℐ	𝒥	𝒦	ℒ	ℳ	𝒩	𝒪	𝒫
U+1D4E0 – 1D4EF	𝒬	ℛ	𝒮	𝒯	𝒰	𝒱	𝒲	𝒳	𝒴	𝒵	𝒶	𝒷	𝒸	𝒹	ℯ	𝒻
U+1D4F0 – 1D4FF	𝒼	𝒽	𝒾	𝒿	𝓀	𝓁	𝓂	𝓃	𝓄	𝓅	𝓆	𝓇	𝓈	𝓉	𝓊	𝓋
U+1D500 – 1D50F	𝓌	𝓍	𝓎	𝓏	𝔄	𝔅	–	𝔇	𝔈	𝔉	𝔊	–	–	𝔍	𝔎	𝔏
U+1D510 – 1D51F	𝔐	𝔑	𝔒	𝔓	𝔔	–	𝔖	𝔗	𝔘	𝔙	𝔚	𝔛	𝔜	–	𝔞	𝔟
U+1D520 – 1D52F	𝔠	𝔡	𝔢	𝔣	𝔤	𝔥	𝔦	𝔧	𝔨	𝔩	𝔪	𝔫	𝔬	𝔭	𝔮	𝔯
U+1D530 – 1D53F	𝔰	𝔱	𝔲	𝔳	𝔴	𝔵	𝔶	𝔷	𝔸	𝔹	–	𝔻	𝔼	𝔽	𝔾	–
U+1D540 – 1D54F	𝕀	𝕁	𝕂	𝕃	𝕄	–	𝕆	–	–	–	𝕊	𝕋	𝕌	𝕍	𝕎	𝕏
U+1D550 – 1D55F	𝕐	–	𝕒	𝕓	𝕔	𝕕	𝕖	𝕗	𝕘	𝕙	𝕚	𝕛	𝕜	𝕝	𝕞	𝕟
U+1D560 – 1D56F	𝕠	𝕡	𝕢	𝕣	𝕤	𝕥	𝕦	𝕧	𝕨	𝕩	𝕪	𝕫	𝕬	𝕭	𝕮	𝕯
U+1D570 – 1D57F	𝕰	𝕱	𝕲	𝕳	𝕴	𝕵	𝕶	𝕷	𝕸	𝕹	𝕺	𝕻	𝕼	𝕽	𝕾	𝕿
U+1D580 – 1D58F	𝖀	𝖁	𝖂	𝖃	𝖄	𝖅	𝖆	𝖇	𝖈	𝖉	𝖊	𝖋	𝖌	𝖍	𝖎	𝖏
U+1D590 – 1D59F	𝖐	𝖑	𝖒	𝖓	𝖔	𝖕	𝖖	𝖗	𝖘	𝖙	𝖚	𝖛	𝖜	𝖝	𝖞	𝖟
U+1D5A0 – 1D5AF	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
U+1D5B0 – 1D5BF	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f
U+1D5C0 – 1D5CF	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
U+1D5D0 – 1D5DF	w	x	y	z	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
U+1D5E0 – 1D5EF	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**a**	**b**
U+1D5F0 – 1D5FF	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**
U+1D600 – 1D60F	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
U+1D610 – 1D61F	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*
U+1D620 – 1D62F	*Y*	*Z*	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*
U+1D630 – 1D63F	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*	***A***	***B***	***C***	***D***
U+1D640 – 1D64F	***E***	***F***	***G***	***H***	***I***	***J***	***K***	***L***	***M***	***N***	***O***	***P***	***Q***	***R***	***S***	***T***
U+1D650 – 1D65F	***U***	***V***	***W***	***X***	***Y***	***Z***	***a***	***b***	***c***	***d***	***e***	***f***	***g***	***h***	***i***	***j***
U+1D660 – 1D66F	***k***	***l***	***m***	***n***	***o***	***p***	***q***	***r***	***s***	***t***	***u***	***v***	***w***	***x***	***y***	***z***
U+1D670 – 1D67F	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
U+1D680 – 1D68F	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f
U+1D690 – 1D69F	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
U+1D6A0 – 1D6AF	w	x	y	z	𝚤	𝚥	–	–	**Α**	**Β**	**Γ**	**Δ**	**Ε**	**Ζ**	**Η**	**Θ**
U+1D6B0 – 1D6BF	**Ι**	**Κ**	**Λ**	**Μ**	**Ν**	**Ξ**	**Ο**	**Π**	**Ρ**	**ϴ**	**Σ**	**Τ**	**Υ**	**Φ**	**Χ**	**Ψ**
U+1D6C0 – 1D6CF	**Ω**	**∇**	**α**	**β**	**γ**	**δ**	**ε**	**ζ**	**η**	**θ**	**ι**	**κ**	**λ**	**μ**	**ν**	**ξ**
U+1D6D0 – 1D6DF	**ο**	**π**	**ρ**	**ς**	**σ**	**τ**	**υ**	**φ**	**χ**	**ψ**	**ω**	**∂**	**ϵ**	**ϑ**	**ϰ**	**ϕ**
U+1D6E0 – 1D6EF	**ϱ**	**ϖ**	*Α*	*Β*	*Γ*	*Δ*	*Ε*	*Ζ*	*Η*	*ϴ*	*Ι*	*Κ*	*Λ*	*Μ*	*Ν*	*Ξ*
U+1D6F0 – 1D6FF	*Ο*	*Π*	*Ρ*	*ϴ*	*Σ*	*Τ*	*Υ*	*Φ*	*Χ*	*Ψ*	*Ω*	*∇*	*α*	*β*	*γ*	*δ*
U+1D700 – 1D70F	*ε*	*ζ*	*η*	*θ*	*ι*	*κ*	*λ*	*μ*	*ν*	*ξ*	*ο*	*π*	*ρ*	*ς*	*σ*	*τ*
U+1D710 – 1D71F	*υ*	*φ*	*χ*	*ψ*	*ω*	*∂*	*ϵ*	*ϑ*	*ϰ*	*ϕ*	*ϱ*	*ϖ*	***Α***	***Β***	***Γ***	***Δ***
U+1D720 – 1D72F	***Ε***	***Ζ***	***Η***	***ϴ***	***Ι***	***Κ***	***Λ***	***Μ***	***Ν***	***Ξ***	***Ο***	***Π***	***Ρ***	***ϴ***	***Σ***	***Τ***
U+1D730 – 1D73F	***Υ***	***Φ***	***Χ***	***Ψ***	***Ω***	***∇***	***α***	***β***	***γ***	***δ***	***ε***	***ζ***	***η***	***θ***	***ι***	***κ***
U+1D740 – 1D74F	***λ***	***μ***	***ν***	***ξ***	***ο***	***π***	***ρ***	***ς***	***σ***	***τ***	***υ***	***φ***	***χ***	***ψ***	***ω***	***∂***
U+1D750 – 1D75F	***ϵ***	***ϑ***	***ϰ***	***ϕ***	***ϱ***	***ϖ***	**Α**	**Β**	**Γ**	**Δ**	**Ε**	**Ζ**	**Η**	**Θ**	**Ι**	**Κ**
U+1D760 – 1D76F	**Λ**	**Μ**	**Ν**	**Ξ**	**Ο**	**Π**	**Ρ**	**ϴ**	**Σ**	**Τ**	**Υ**	**Φ**	**Χ**	**Ψ**	**Ω**	**∇**
U+1D770 – 1D77F	**α**	**β**	**γ**	**δ**	**ε**	**ζ**	**η**	**θ**	**ι**	**κ**	**λ**	**μ**	**ν**	**ξ**	**ο**	**π**
U+1D780 – 1D78F	**ρ**	**ς**	**σ**	**τ**	**υ**	**φ**	**χ**	**ψ**	**ω**	**∂**	**ϵ**	**ϑ**	**ϰ**	**ϕ**	**ϱ**	**ϖ**
U+1D790 – 1D79F	***Α***	***Β***	***Γ***	***Δ***	***Ε***	***Ζ***	***Η***	***ϴ***	***Ι***	***Κ***	***Λ***	***Μ***	***Ν***	***Ξ***	***Ο***	***Π***
U+1D7A0 – 1D7AF	***Ρ***	***ϴ***	***Σ***	***Τ***	***Υ***	***Φ***	***Χ***	***Ψ***	***Ω***	***∇***	***α***	***β***	***γ***	***δ***	***ε***	***ζ***
U+1D7B0 – 1D7BF	***η***	***θ***	***ι***	***κ***	***λ***	***μ***	***ν***	***ξ***	***ο***	***π***	***ρ***	***ς***	***σ***	***τ***	***υ***	***φ***
U+1D7C0 – 1D7CF	***χ***	***ψ***	***ω***	***∂***	***ϵ***	***ϑ***	***ϰ***	***ϕ***	***ϱ***	***ϖ***	**Ϝ**	𝟋	–	–	**0**	**1**
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Table 12: LM Math vs. NewCM Math, *cont.*

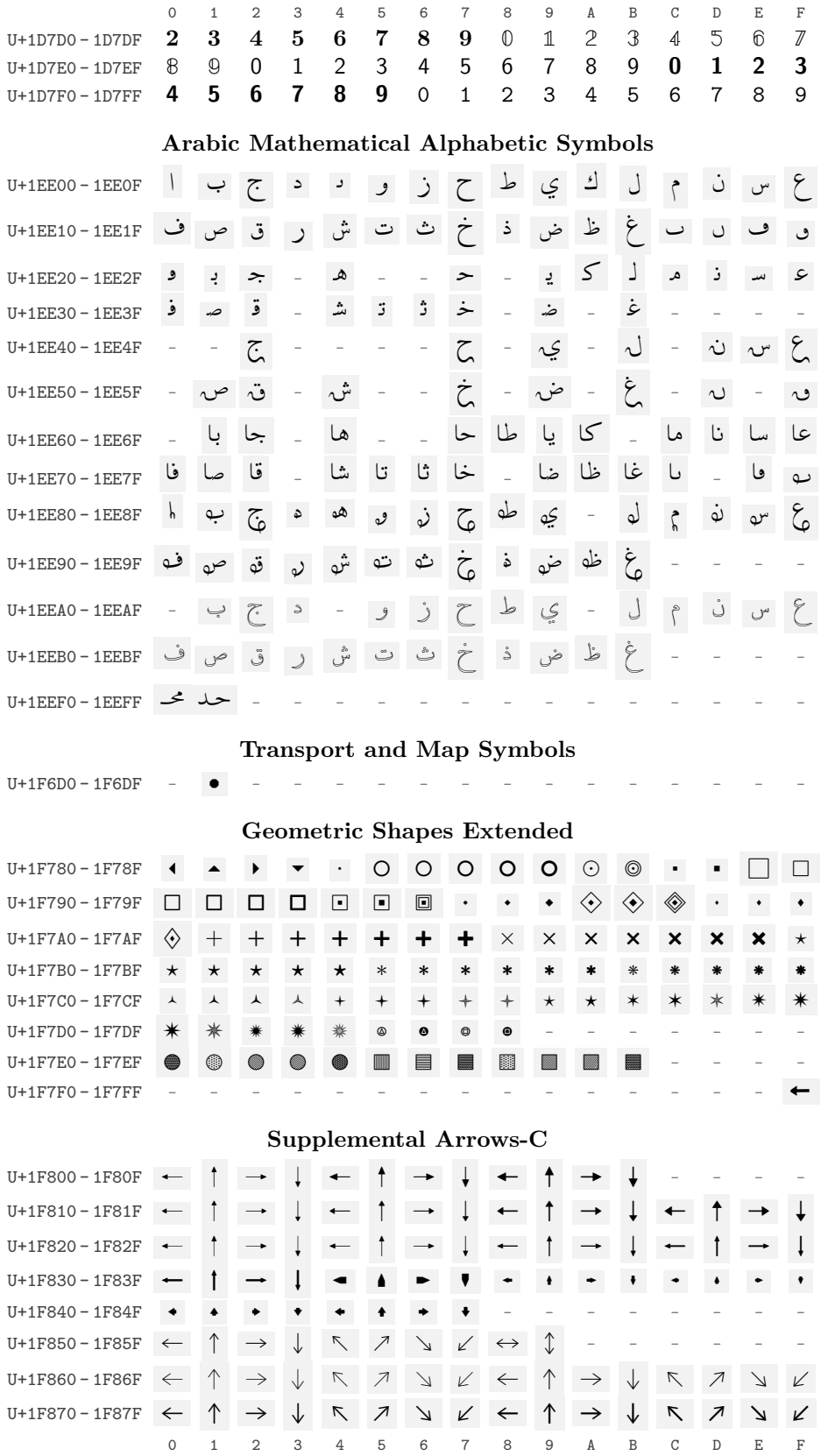
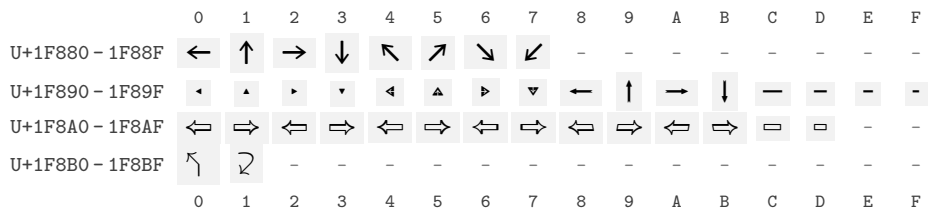


Table 12: LM Math vs. NewCM Math, *cont.*



Total number of glyphs shown from `Latin Modern Math`: 2045
Comparison font `New Computer Modern Math` has 0 missing and 2103 extra glyphs

4.5 Garamond Libre's Byzantine Musical Symbols

As a final example we exhibit the Byzantine Musical Symbols as provided by Garamond Libre. Command used:

```
\displayfonttable[range-start=1D000, range-end=1D0FF,
                  hex-digits=block,
                  missing-glyph-color=black!50, color=black!75,
                  statistics-format=Total number of glyphs in
                    this block of #1 is #2]
                 {Garamond Libre}
```

Note that we have altered the text produced by the statistics, because the default is somewhat misleading if only a portion of the font is displayed. This produces the following table:

Table 13: Garamond Libre

Byzantine Musical Symbols

Total number of glyphs in this block of Garamond Libre is 246

5 The package implementation

1 ⟨*package⟩

By default the package uses coloring to improve the table appearance and therefore requires a color package.

```
\RequirePackage{xcolor}
⟨@@=fmuft⟩
```

We need the package xparse for specifying the document-level interface commands and l3keys2e to use the expl3 key value methods within LaTeX 2ε. These packages automatically require expl3 so there is no need to load that explicitly. Actually, expl3, l3keys2e and the xparse functionality is now all part of the LaTeX kernel so the next line is actually not needed at all with a current LaTeX kernel, but in order to support older installations we keep it for now.

```
\RequirePackage{xparse,l3keys2e}
```

Here we introduce the package and specify its version number:

```
\ProvidesExplPackage{unicodefonttable}
                    {\unicodefonttabledate}
                    {\unicodefonttableversion}
                    {Producing font tables for Unicode and other fonts}
```

5.1 User interface commands

Throughout the implementation we will define a number of keys (and their allowed values). We introduce them at the point where they are used, so they are sprinkled throughout the code.[2]

\fonttablesetup To set up user defaults for the keys we provide a standard interface. The command \unicodefonttabletablesetup expects a key/value list and can be called as often as necessary.

```
\NewDocumentCommand \fonttablesetup { m }
  { \keys_set:nn {__fmuft} {#1} \ignorespaces }
```

(*End of definition for* \fonttablesetup. *This function is documented on page 903.*)

\displayfonttable The document-level command for generating a font table.

```
\NewDocumentCommand\displayfonttable {s O{} m o}{%
  \IfBooleanTF #1
     {
```

For the starred form we preset a number of keys with values suitable when displaying 8-bit legacy fonts. With such fonts Unicode block headers make little sense (as the fonts do not conform to the Unicode layout and since they have at most 265 glyphs). It is therefore also unnecessary to loop over the whole Unicode range of the first plane. If necessary all of them can still be overwritten in the optional argument.

Some font names contain special character, e.g., _ or &. We therefore turn the font name into a string before passing it on but we allow for the possibility that the font name was given in a macro, so we do this with \tl_to_str:o (tough if the font name starts out with a special character then you have to use \string or something to prevent the special character from causing issues).

```
       \exp_args:Nne
       \__fmuft_display_fonttable:nnn
          {nostatistics,display-block=none,hex-digits=head,range-end=FF,#2}
          { \tl_to_str:o {#3} } {#4}
     }
     {
       \exp_args:Nne
       \__fmuft_display_fonttable:nnn {#2}{ \tl_to_str:o {#3} }{#4}
     }
}
```

[2] This fits with the way this package was developed. I first implemented a single rigid table layout without configuration possibilities and then thought about which parts I wanted to have flexible. I then replaced the rigid code with code that is affected by setting key/value pairs.

(*End of definition for* `\displayfonttable`. *This function is documented on page 903.*)

`\__fmuft_display_fonttable:nnn` This command is the main workhorse of the package. It produces a `longtable` containing all font glyphs with 16 glyphs per row. The first optional argument is used to configure the table through key/value pairs, the mandatory argument is the font name to display (in `fontspec` conventions) and the final optional argument is the font feature list if any. If the latter is not provided it will get a special value (`--NoValue--`) assigned by `xparse`, which is something that can be tested for.

```
\cs_new:Npn \__fmuft_display_fonttable:nnn #1#2#3 {
  \group_begin:
```

First initialize the font that should be displayed (perhaps with a feature list) and then update the key/value list using `#1`.

```
    \fontspec{#2}[#3]
    \keys_set:nn{__fmuft}{#1}
```

If the LuaTEX engine is used without HarfBuzz and the display range includes code points above `U+EFFFF` the output shows remappings and not what is in the font, so we issue a warning.

```
    \bool_lazy_and:nnT
        { \sys_if_engine_luatex_p: }
        { \int_compare_p:nNn { "EFFFF } < { "\l__fmuft_range_end_tl } }
        {
          \directlua{token.put_next(token.create(font.getfont(font.current()).hb~
                    and~ 'use_none:n'~  or~ 'use:n'))}
            { \msg_warning:nn {unicodefonttable}{noharfbuzz} }
        }
```

If the user has asked for a comparsion to some other font we need to set this up:

```
    \tl_if_empty:NTF \l__fmuft_compare_with_tl
       { \tl_clear:N \l__fmuft_compare_font_tl }
       {
         \setfontface \l__fmuft_compare_font_tl {\l__fmuft_compare_with_tl}[]
         \cs_set_eq:NN \__fmuft_handle_missing_glyph:n
                       \__fmuft_handle_missing_glyph_compare:n
       }
```

Typesetting the font tables in twocolumn mode makes little sense due to their width, and if `longtable` is used it will complain. However there is one case where it should work: in a page-wide float. To make this happen we claim that we are not in twocolumn mode if the display is inside a vertical box.

```
    \if_mode_vertical: \if_mode_inner: \@twocolumnfalse \fi: \fi:
```

Then we start the table with 17 columns. We use `longtable` if we produce a caption and `longtable*` if not (so that the table number is not increased, which would look odd if you have other tables in your document).

```
    \begin{longtable\bool_if:NF\l__fmuft_display_header_bool{*}}
          {@{}r@{\quad}*{16}{c}@{}}
```

Special headers and footers are set up first:

```
      \__fmuft_setup_header_footer:nn{#2}{#3}
```

Then we produce all table rows with the glyphs.

```
      \__fmuft_produce_table_rows:
```

At the very end we may typeset some statistics. This can't be done in the table footer, because the data is dynamic (e.g., number of glyphs processed) and the table footers are static and do not change based on the table content.

```
      \__fmuft_handle_table_ending:n {#2}
    \end{longtable\bool_if:NF\l__fmuft_display_header_bool{*}}
  \group_end:
}
```

(*End of definition for* `\__fmuft_display_fonttable:nnn`.)

```
\msg_new:nnn {unicodefonttable}{noharfbuzz}
    { You~ asked~ for~ displaying~ glyphs~ with~ code \iow_newline:
      points~ above~ U+EFFFF~ \msg_line_context: ,~ i.e.,~ from~ the~
      'Supplementary~ Private~ Use~ Area-A'\iow_newline:
      without~ specifying~ '[Renderer=Harfbuzz]'~ when~
      loading~ the~ font.
      \iow_newline:\iow_newline:
      With~ LuaLaTeX,~ this~ Unicode~ region~ is~ used~
      for~ remappings~ (if~ the~ HarfBuzz~ engine~ is~ not~ used).~
      Thus,~ the~ results~ shown~ do~ not~ reflect~ what~
      is~ in~ the~ font!
    }
```

`\fonttableglyphcount` `\g__fmuft_glyph_int` `\g__fmuft_glyph_only_B_int` `\g__fmuft_glyph_also_B_int`

While generating the font table we count the number of glyphs we see (and typeset). The total is available in the command `\fonttableglyphcount` after the table got finished and will be reset to zero when the next table starts.

```
\DeclareDocumentCommand \fonttableglyphcount {}
                        { \int_use:N \g__fmuft_glyph_int }
\int_new:N \g__fmuft_glyph_int
```

When comparing fonts we also record data for the second font: the number of glyphs in both and the number of glyphs only in the second one.

```
\int_new:N \g__fmuft_glyph_only_B_int
\int_new:N \g__fmuft_glyph_also_B_int
```

(*End of definition for* `\fonttableglyphcount` *and others. This function is documented on page 903.*)

5.2 The overall table layout

`\__fmuft_setup_header_footer:nn`

Setting up header and footer lines of the table. This macro receives the *font name* and the *font features* specified by the user as its arguments.

```
\cs_new:Npn \__fmuft_setup_header_footer:nn #1#2{
```

On the first page of the table the header may show a caption or some other sort of title based on the value of `\l__fmuft_display_header_bool`. The formatting is handled by `\__fmuft_format_table_title:nn` which can be customized through the key `title-format`.

```
    \bool_if:NT \l__fmuft_display_header_bool
       { \__fmuft_format_table_title:nn{#1}{#2} \__fmuft_debug_nl:n{T}\\*[6pt] }
```

We may also want to display a line of hex digits. This is controlled through the key `hex-digits` that accepts different values: `head`, `foot`, `head+foo`, `block` (after a block title) or `none`.

```
    \bool_if:NT \l__fmuft_header_hex_digits_bool
      { \__fmuft_display_row_of_hex_digits:   \__fmuft_debug_nl:n{H}\\*      }
  \endfirsthead
```

Headers for later table pages have a continuation title and maybe a row of hex digits.

```
    \bool_if:NT \l__fmuft_display_header_bool
      { \__fmuft_format_table_cont:nn{#1}{#2} \__fmuft_debug_nl:n{T}\\*[6pt] }
    \bool_if:NT \l__fmuft_header_hex_digits_bool
      { \__fmuft_display_row_of_hex_digits:   \__fmuft_debug_nl:n{H}\\*      }
  \endhead
```

Footers of the table are either empty or show a row of hex digits.

```
    \bool_if:NT \l__fmuft_footer_hex_digits_bool
      { \__fmuft_display_row_of_hex_digits:   \__fmuft_debug_nl:n{H}\\*      }
  \endfoot
```

The footer of the last page of the table will always be empty. Any special row, such as a row of hex digits, will be provided in the table body. The reason is that we may want to display statistics at the very end of the table and those can't be placed into a static footer.

```
  \endlastfoot
}
```

(*End of definition for* `\__fmuft_setup_header_footer:nn`.)

`\l__fmuft_header_hex_digits_bool` `\l__fmuft_footer_hex_digits_bool` `\l__fmuft_blockwise_hex_digits_bool`

Here are the booleans we use in the code.

```
\bool_new:N \l__fmuft_header_hex_digits_bool
\bool_new:N \l__fmuft_footer_hex_digits_bool
\bool_new:N \l__fmuft_blockwise_hex_digits_bool
```

(*End of definition for* `\l__fmuft_header_hex_digits_bool`, `\l__fmuft_footer_hex_digits_bool`, *and* `\l__fmuft_blockwise_hex_digits_bool`.)

`\__fmuft_display_row_of_hex_digits:` `\__fmuft_format_hex_digit:n`

Producing a row of hex digits is simple.

```
\cs_new:Npn \__fmuft_display_row_of_hex_digits: {
    & \__fmuft_format_hex_digit:n{0}     & \__fmuft_format_hex_digit:n{1}
    & \__fmuft_format_hex_digit:n{2}     & \__fmuft_format_hex_digit:n{3}
    & \__fmuft_format_hex_digit:n{4}     & \__fmuft_format_hex_digit:n{5}
    & \__fmuft_format_hex_digit:n{6}     & \__fmuft_format_hex_digit:n{7}
    & \__fmuft_format_hex_digit:n{8}     & \__fmuft_format_hex_digit:n{9}
    & \__fmuft_format_hex_digit:n{A}     & \__fmuft_format_hex_digit:n{B}
    & \__fmuft_format_hex_digit:n{C}     & \__fmuft_format_hex_digit:n{D}
    & \__fmuft_format_hex_digit:n{E}     & \__fmuft_format_hex_digit:n{F}  }
```

Each digit is typeset in typewriter and in script size. We offer font and color customizations. Note that it is important to set an explicit family. Otherwise the hex digits are formatted using the current table font (which may or may not work at all).

```
\cs_new:Npn \__fmuft_format_hex_digit:n #1 {
  \l__fmuft_hex_digits_font_tl \l__fmuft_color_tl #1 }
```

(*End of definition for* `\__fmuft_display_row_of_hex_digits:` *and* `\__fmuft_format_hex_digit:n`.)

`\l__fmuft_color_tl`

The token list to hold definition if set up.

```
\tl_new:N \l__fmuft_color_tl
```

(*End of definition for* `\l__fmuft_color_tl`.)

Key setup (overall table) Here are the definitions for the keys used in the code above:

```
\keys_define:nn {__fmuft} {
```

The `header` key is a boolean that determines if a header title should be produced (default)

```
        ,header .bool_set:N   = \l__fmuft_display_header_bool
        ,header .default:n    = true
        ,header .initial:n    = true
```

To ease the setup we also support the key `noheader` which is a short form for `header=false`.

```
        ,noheader .bool_set_inverse:N = \l__fmuft_display_header_bool
        ,noheader .default:n          = true
```

The default for the `title-format` key is to produce a `\caption` listing the font name and any features (if given). Note the `\IfValueTF` command (provided by `xparse`) that checks if the second argument got any value or has the special `--NoValue--` value.

```
106         ,title-format      .cs_set:Np = \__fmuft_format_table_title:nn #1#2
107         ,title-format      .initial:n =
108           \IfValueTF{#2} { \caption{ #1~ (features:~ \texttt{\small#2}) } }
109                          { \caption{ #1 } }
```

The default continuation title ignores the given features, so the formatting is somewhat simpler. It uses `\caption[]{...}` to make a caption that doesn't alter the table number.

```
110         ,title-format-cont .cs_set:Np = \__fmuft_format_table_cont:nn #1#2
111         ,title-format-cont .initial:n = \caption[]{#1~ \emph{cont.}}
```

The key `hex-digits` is implemented as a choice, where each allowed value sets different booleans that are then used in the code.

```
112         ,hex-digits  .choice:
113         ,hex-digits / block   .code:n  =
114           \bool_set_true:N   \l__fmuft_blockwise_hex_digits_bool
115           \bool_set_false:N  \l__fmuft_header_hex_digits_bool
116           \bool_set_false:N  \l__fmuft_footer_hex_digits_bool
117         ,hex-digits / foot   .code:n  =
118           \bool_set_true:N   \l__fmuft_footer_hex_digits_bool
119           \bool_set_false:N  \l__fmuft_header_hex_digits_bool
120           \bool_set_false:N  \l__fmuft_blockwise_hex_digits_bool
121         ,hex-digits / head   .code:n  =
122           \bool_set_true:N   \l__fmuft_header_hex_digits_bool
123           \bool_set_false:N  \l__fmuft_footer_hex_digits_bool
124           \bool_set_false:N  \l__fmuft_blockwise_hex_digits_bool
125         ,hex-digits / head+foot  .code:n  =
126           \bool_set_true:N   \l__fmuft_header_hex_digits_bool
127           \bool_set_true:N   \l__fmuft_footer_hex_digits_bool
128           \bool_set_false:N  \l__fmuft_blockwise_hex_digits_bool
129         ,hex-digits / none   .code:n  =
130           \bool_set_false:N  \l__fmuft_header_hex_digits_bool
131           \bool_set_false:N  \l__fmuft_footer_hex_digits_bool
132           \bool_set_false:N  \l__fmuft_blockwise_hex_digits_bool
133         ,hex-digits  .initial:n = head
```

The font for hex digits are set with `hex-digits-font`.

```
134        ,hex-digits-font  .tl_set:N  = \l__fmuft_hex_digits_font_tl
135        ,hex-digits-font  .initial:n = \ttfamily \scriptsize
```

Customizing the row header (on the left) can be done with this key. Defaults for font, fontsize, and color is set on the outside, but can, of course, be overwritten inside if that is desired.

```
136         ,hex-digits-row-format  .cs_set:Np = \__fmuft_format_row_hex_digits:n #1
137         ,hex-digits-row-format  .initial:n = U+#1 0 \, - \, #1 F
```

The `color` key is used in most places that get colored; some have their own key but default to the main color.

```
138         ,color .choice:
139         ,color / none    .code:n    = \tl_clear:N \l__fmuft_color_tl
140         ,color / unknown .code:n    = \tl_set:Nn \l__fmuft_color_tl { \color {#1} }
141         ,color           .initial:n = blue
142   }
```

__fmuft_handle_table_ending:n At the end of the table we may want to display a final row of hex digits and perhaps some statistics, i.e., the number of typeset glyphs.

```
\cs_new:Npn \__fmuft_handle_table_ending:n #1 {
  \__fmuft_debug_nl:n{H} \\*
  \bool_if:NT \l__fmuft_footer_hex_digits_bool
    { \__fmuft_display_row_of_hex_digits: \__fmuft_debug_nl:n{H} \\*   }
  \bool_if:NT \l__fmuft_display_statistics_bool
    { \\*[2pt]
      \multicolumn{17}{l}{ \l__fmuft_stats_font_tl
```

If we do font comparison, we use a different command for displaying statistics and pass more data to it.

```
        \tl_if_empty:NTF \l__fmuft_compare_with_tl
          {
            \__fmuft_format_stats:nn{#1}{\fonttableglyphcount}
          }
          {
            \__fmuft_format_compare_stats:nnnnnn{#1}{\fonttableglyphcount}
               { \l__fmuft_compare_with_tl }
```

The extra arguments are total glyph number in second font, glyphs missing in second font and glyphs only in second font.

```
               { \int_eval:n { \int_use:N\g__fmuft_glyph_also_B_int +
                               \int_use:N\g__fmuft_glyph_only_B_int }
               }
               { \int_eval:n { \fonttableglyphcount -
                               \int_use:N\g__fmuft_glyph_also_B_int }
               }
               { \int_use:N\g__fmuft_glyph_only_B_int }
          }
```

We don't know exactly how wide the table is (and nor does the user) but one may need to use `\parbox` when formatting the statistic line(s). So we back up a bit (rather random) which allows us to use `\parbox{\linewidth}{...` in the key without thinking too much about it.

```
        \hspace*{-3cm}
      }
    }
}
```

Key setup (for statistics) Here are the keys used above. By default we produce statistics.

```
\keys_define:nn {__fmuft} {
    ,statistics .bool_set:N = \l__fmuft_display_statistics_bool
    ,statistics .default:n   = true
    ,statistics .initial:n   = true
```

the key `nostatistics` is just short for `statistics=false`:

```
    ,nostatistics .bool_set_inverse:N = \l__fmuft_display_statistics_bool
    ,nostatistics .default:n = true
```

The default font we use is `\normalfont`. Again we need to supply a family to avoid getting the font used in the table body.

```
    ,statistics-font    .tl_set:N  = \l__fmuft_stats_font_tl
    ,statistics-font    .initial:n = \normalfont\small
```

And here we have the default text. There is only space for a single line. If more text is needed one needs to provide some explicit `\parbox`.

```
    ,statistics-format  .cs_set:Np = \__fmuft_format_stats:nn #1#2
    ,statistics-format  .initial:n = Total~ number~ of~ glyphs~ shown~ from~ #1:~#2
  }
```

(*End of definition for* `\__fmuft_handle_table_ending:n`.)

`\__fmuft_debug_nl:n` While developing the code I had a bit of trouble getting the line endings correct, so I added a little macro that made them visible (displaying its argument in the table margin when the key `debug` is used. By default it does nothing.

```
\cs_new:Npn \__fmuft_debug_nl:n #1 {}
```

Key setup (debugging) This key is really internal and is therefore not documented above (and its behavior may changes over time).

```
\keys_define:nn {__fmuft} {
  debug .code:n = \cs_set:Npn \__fmuft_debug_nl:n ##1
                              {\rlap{\normalfont\scriptsize \qquad ##1}} }
```

(*End of definition for* `\__fmuft_debug_nl:n`.)

5.3 The producing the table content

The body of the table consists of rows with sixteen glyphs each and to produce it we loop through all possible Unicode points starting at `U+0000` and ending with `U+FFFF`. This is implemented with a four-level nested loop that runs through the values `0`, `1`, …, `F` with the current hex value in each of the four positions stored in some variable.

`\g__fmuft_hex_H_tl` `\g__fmuft_hex_A_tl` `\g__fmuft_hex_B_tl` `\g__fmuft_hex_C_tl` `\g__fmuft_hex_H_tl` is a bit special because, it is initially not zero, but empty, so that slots in the lower plane are denoted by 4 hex digits. We really only need three further variables, as the value in the innermost loop can used directly.

```
\tl_new:N \g__fmuft_hex_H_tl   % higher plane
\tl_new:N \g__fmuft_hex_A_tl
\tl_new:N \g__fmuft_hex_B_tl
\tl_new:N \g__fmuft_hex_C_tl
```

(*End of definition for* `\g__fmuft_hex_H_tl` *and others.*)

`\c__fmuft_hex_digits_clist` Here is the sequence we loop through on each level, except the one for the outer level.

```
\clist_const:Nn\c__fmuft_hex_digits_clist{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
```

(*End of definition for* `\c__fmuft_hex_digits_clist`.)

`\__fmuft_produce_table_rows:` `\__fmuft_handle_hex_H:n` `\__fmuft_handle_hex_A:n` `\__fmuft_handle_hex_B:n` `\__fmuft_handle_hex_C:n` `\__fmuft_handle_hex_D:n` The overall code layout is then fairly simply:

```
\cs_new:Npn \__fmuft_produce_table_rows: {
```

First to some general initialization

```
  \__fmuft_initialize_table_rows:
```

and then loop we start the loop. The outer level is a bit special as currently Unicode has only slots allocated in plane 0, 1, 2 and E (well, and F, but that is a private area) so we loop only over those and instead of `0` we use an empty value. Not covered is the whole of plane 16 which too is now a private area.

```
  \clist_map_function:nN { { } , 1, 2, E, F } \__fmuft_handle_hex_H:n }
```

Most fonts do not have glyphs in the higher planes, which is why by default we don't loop using a nonempty `\__fmuft_handle_hex_H:n`. But if the user wants to scan and display the higher slots they can by setting `range-end` appropriatly.

So after setting `\__fmuft_handle_hex_H:n` we loop over `\c__fmuft_hex_digits_clist` for the next hex digit (which we call "A").

```
\cs_new:Npn \__fmuft_handle_hex_H:n #1 { \tl_gset:Nn\g__fmuft_hex_H_tl{#1}
  \clist_map_function:NN \c__fmuft_hex_digits_clist \__fmuft_handle_hex_A:n }
```

Handling "A" means storing its value for later use and then start a loop for setting the second (or third on higher planes) hex digits:

```
\cs_new:Npn \__fmuft_handle_hex_A:n #1 { \tl_gset:Nn\g__fmuft_hex_A_tl{#1}
   \clist_map_function:NN \c__fmuft_hex_digits_clist \__fmuft_handle_hex_B:n }
```

Same game for "B" and "C"[3]:

```
\cs_new:Npn \__fmuft_handle_hex_B:n #1 { \tl_gset:Nn\g__fmuft_hex_B_tl{#1}
   \clist_map_function:NN \c__fmuft_hex_digits_clist \__fmuft_handle_hex_C:n }

\cs_new:Npn \__fmuft_handle_hex_C:n #1 { \tl_gset:Nn\g__fmuft_hex_C_tl{#1}
   \clist_map_function:NN \c__fmuft_hex_digits_clist \__fmuft_handle_hex_D:n }
```

In the innermost loop we now have the full Unicode number available, so there we have to decide what to do with it. This is done by `\__fmuft_handle_hex_D:n` that receives the full number, e.g., `1A7C` or `1AD00`, as its argument.

```
\cs_new:Npn \__fmuft_handle_hex_D:n #1 {
  \__fmuft_handle_slot:x
     { " \g__fmuft_hex_H_tl \g__fmuft_hex_A_tl
         \g__fmuft_hex_B_tl \g__fmuft_hex_C_tl #1 }
}
```

(*End of definition for* `\__fmuft_produce_table_rows:` *and others.*)

`\g__fmuft_row_tl`

We first collect the glyphs for a whole row before deciding to typeset it, because if the row is entirely empty we want to omit it. The data for the row is collected slot by slot and the typesetting information (the glyph or the indication for a missing glyph is appended to `\g__fmuft_row_tl`.

```
\tl_new:N \g__fmuft_row_tl
```

(*End of definition for* `\g__fmuft_row_tl`.)

`\__fmuft_handle_slot:n`
`\__fmuft_handle_slot:x`

If the current slot number under inspection contains a glyph in our font we want to typeset it. But we don't do this immediately, instead we build up the whole row and typeset it later. We therefore append a `&` and the glyph (including the necessary formatting) to the token list `\g__fmuft_row_tl`.

```
\cs_new:Npn \__fmuft_handle_slot:n #1 {
   \__fmuft_if_uchar_exists:nTF { #1 }
     { \tl_gput_right:Nn \g__fmuft_row_tl
                         { & \__fmuft_format_glyph:n { \symbol{#1} } }
```

We then increment the overall glyph count and record that we have seen at least one glyph in the current row. There is not much point in displaying rows that are completely empty; indeed, we'd end up with extremely large tables which are mostly empty.

```
       \int_gincr:N\g__fmuft_glyph_int
       \bool_gset_true:N \g__fmuft_glyph_seen_bool
```

If we do font comparison we also check if the glyph is in the second font and if so record that fact.

```
       \tl_if_empty:NF \l__fmuft_compare_font_tl
          {
            \group_begin:
              \l__fmuft_compare_font_tl
              \__fmuft_if_uchar_exists:nT { #1 }
```

[3] Actually this is a white lie. In reality we do a lot of extra stuff when handling "C" so later one we give a second definition for `\__fmuft_handle_hex_C:n` but for understanding the overall picture the simpler one shown here is better.

```
                    { \int_gincr:N \g__fmuft_glyph_also_B_int }
                \group_end:
              }
        }
```

If the current slot has no glyph in the font we also add a `&` followed by something that indicates the glyph is missing. If we do font comparison, it may show the glyph from the other font (if it exists there) in some special way to indicate which glyph should be in this slot.

```
        { \__fmuft_handle_missing_glyph:n {#1} }
}
\cs_generate_variant:Nn \__fmuft_handle_slot:n {x}
```

(*End of definition for* `\__fmuft_handle_slot:n`.)

`\__fmuft_handle_missing_glyph:n`
`\__fmuft_handle_missing_glyph_std:n`
`\__fmuft_handle_missing_glyph_compare:n`

In the standard case we typeset a special symbol to indicate that the glyph is missing. For this case we provide some customization through keys: `\l__fmuft_missing_glyph_tl` holds the symbol for a missing glyph (default: a hyphen). It is typeset in a specific color and we allow for setting it in a special font. The actual symbol number in `#1` is not needed in this scenario.

```
\cs_new:Npn \__fmuft_handle_missing_glyph_std:n #1 {
 \tl_gput_right:Nn \g__fmuft_row_tl
    { &
      \__fmuft_format_glyph:n {
%        \colorbox{black!30}                      % <--- povide interface
                 {\l__fmuft_missing_glyph_color_tl
                           \l__fmuft_missing_glyph_font_tl
                           \l__fmuft_missing_glyph_tl }
      }
    }
}
```

Key setup (missing glyphs) Here are the keys for customizing the missing glyph representation.

```
\keys_define:nn {__fmuft} {
   missing-glyph-color .choice:
  ,missing-glyph-color / none    .code:n    =
       \tl_clear:N \l__fmuft_missing_glyph_color_tl
  ,missing-glyph-color / unknown .code:n    =
       \tl_set:Nn \l__fmuft_missing_glyph_color_tl { \color {#1} }
%
  ,missing-glyph-font  .tl_set:N  = \l__fmuft_missing_glyph_font_tl
  ,missing-glyph-font  .initial:n = \ttfamily \scriptsize
  ,missing-glyph       .tl_set:N  = \l__fmuft_missing_glyph_tl
  ,missing-glyph       .initial:n = -  }
```

The default definition for the color is to use the same as the one specified by the `color` key. We therefore define the default outside of the l3keys method.

```
\tl_new:N  \l__fmuft_missing_glyph_color_tl
\tl_set:Nn \l__fmuft_missing_glyph_color_tl {\l__fmuft_color_tl}
```

This is the version that handles a missing glyph by checking the `compare-with` font to see if that font contains the glyph. If yes, the substitute glyph will be typeset, otherwise the missing glyph symbol is shown by calling `\__fmuft_handle_missing_glyph_std:n`.

```
\cs_new:Npn \__fmuft_handle_missing_glyph_compare:n #1 {
  \group_begin:
```

Locally switch to the other font, then check for the glyph:

```
    \l__fmuft_compare_font_tl
    \__fmuft_if_uchar_exists:nTF { #1 }
      {
```

If available, format it (together with the `&`) but use a special color and perhaps a background color.

```
        \tl_gput_right:Nn \g__fmuft_row_tl
          { &
            \__fmuft_format_glyph:n
              { \l__fmuft_compare_bgcolor_tl { \l__fmuft_compare_color_tl
                                               \l__fmuft_compare_font_tl
                                               \symbol {#1} }
              }
          }
```

Having seen a glyph only in the second font we record this fact.

```
        \int_gincr:N \g__fmuft_glyph_only_B_int
```

Also tell the algorithm that we have seen a glyph to typeset. If we don't do this then a row consisting of only substitute glyphs is not typeset. However, we don't update the glyph count, because this is not a glyph from the main font we display.

```
        \bool_gset_true:N \g__fmuft_glyph_seen_bool
      }
```

If the alternate font doesn't have the glyph either we typeset the missing glyph symbol.

```
      { \__fmuft_handle_missing_glyph_std:n {} }
  \group_end:
}
```

Key setup (comparison) In order to display glyphs from a secondary font we need a secondary color for the glyph itself and possibly some background color.

```
\tl_new:N \l__fmuft_compare_with_tl
\tl_new:N \l__fmuft_compare_color_tl
\tl_new:N \l__fmuft_compare_bgcolor_tl

\keys_define:nn {__fmuft}
    {
      ,compare-with .tl_set:N  = \l__fmuft_compare_with_tl
      ,compare-with .initial:n =
      ,compare-color .choice:
      ,compare-color / none    .code:n
            = \tl_clear:N \l__fmuft_compare_color_tl
      ,compare-color / unknown .code:n
            = \tl_set:Nn  \l__fmuft_compare_color_tl { \color {#1} }
      ,compare-color           .initial:n = red
      ,compare-bgcolor .choice:
      ,compare-bgcolor / none    .code:n
            = \tl_clear:N \l__fmuft_compare_bgcolor_tl
      ,compare-bgcolor / unknown .code:n
            = \tl_set:Nn  \l__fmuft_compare_bgcolor_tl { \colorbox {#1} }
      ,compare-bgcolor           .initial:n = black!10
```

If we run a comparison we show different statistics that have their own key.

```
      ,statistics-compare-format  .cs_set:Np
            = \__fmuft_format_compare_stats:nnnnnn #1#2#3#4#5#6
      ,statistics-compare-format  .initial:n
            = \parbox{\linewidth}{
                  Total~ number~ of~ glyphs~ shown~ from~ \texttt{#1}:~#2\\
```

```
                    Comparison~ font~ \texttt{#3}~ has~ #5~ missing~ and~ #6~
                    extra~ glyphs}
    }
```

By default, i.e., if no font for comparison has been specified, we handle missing glyphs by displaying a missing glyph symbol.

```
\cs_new_eq:NN \__fmuft_handle_missing_glyph:n
              \__fmuft_handle_missing_glyph_std:n
```

(*End of definition for* `\__fmuft_handle_missing_glyph:n`, `\__fmuft_handle_missing_glyph_std:n`, *and* `\__fmuft_handle_missing_glyph_compare:n`.)

`\__fmuft_format_glyph:n` Every glyph is typeset in a box of equal width with the glyph centered and if necessary protruding on both sides.

```
\cs_new:Npn \__fmuft_format_glyph:n #1 {
  \hbox_to_wd:nn {\l__fmuft_glyph_box_dim} { \hss #1 \hss } }
```

Key setup (glyph typesetting) The key to customize the width. The 6pt are fine for most cases.

```
\dim_new:N\l__fmuft_glyph_box_dim

\keys_define:nn {__fmuft} {
   glyph-width .dim_set:N = \l__fmuft_glyph_box_dim
  ,glyph-width .initial:n = 6pt
}
```

(*End of definition for* `\__fmuft_format_glyph:n`.)

`\__fmuft_if_uchar_exists:n` For testing whether or not a slot position contains a glyph we need to resort to low-level methods, because so far `expl3` doesn't offer an interface.

```
\prg_set_conditional:Npnn \__fmuft_if_uchar_exists:n #1 { TF , T }
  { \tex_iffontchar:D \tex_font:D #1 \scan_stop:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

(*End of definition for* `\__fmuft_if_uchar_exists:n`.)

5.4 Handling a single row

`\__fmuft_handle_hex_C:n` As promised here is the read definition for `\__fmuft_handle_hex_C:n` in all its glory.

```
\cs_set:Npn \__fmuft_handle_hex_C:n #1 {
```

We are now at the start of a new row (but with the last row not yet typeset) and this last row may need a Unicode block heading before it. This is the reason why we have to delay the typesetting, because in case the line doesn't contain any glyphs we want to typeset neither and that is only known after all slots in the row have been processed.

```
  \__fmuft_maybe_typeset_a_row_and_display_a_block_title:
```

We then store away the value for the third hex digit (denoted as C) in order to start with the next row.

```
  \tl_gset:Nn\g__fmuft_hex_C_tl{#1}
```

Being at the start of a new row we might be at the start of a new Unicode block. If so we have to update the block title to add in front of the row when we typeset it (or in front of one of the next rows if the first rows in the is block have no glyphs). If we are still in the same block no update happens.

```
\__fmuft_update_block_title:n { \g__fmuft_hex_H_tl
                                \g__fmuft_hex_A_tl
                                \g__fmuft_hex_B_tl
                                \g__fmuft_hex_C_tl }
```

We now check if this row is within the requested range, i.e., greater than or equal to `\l__fmuft_range_start_tl` and not greater than `\l__fmuft_range_end_tl`.

```
  \int_compare:nNnF
      { " \g__fmuft_hex_H_tl \g__fmuft_hex_A_tl
          \g__fmuft_hex_B_tl \g__fmuft_hex_C_tl 0 }
    < { "\l__fmuft_range_start_tl }
    {
      \int_compare:nNnTF
          { " \g__fmuft_hex_H_tl \g__fmuft_hex_A_tl
              \g__fmuft_hex_B_tl \g__fmuft_hex_C_tl 0 }
        > { "\l__fmuft_range_end_tl }
```

If we are past the `end-range` we break out the clist mapping, to avoind unnecessary repetition. This should be propagated back to the outer clists as well (not done).

```
        {  \clist_map_break: }
```

If we are within range we process the slots in the row by first initializing `\g__fmuft_row_tl` with the row title (the info on the left) and then loop through all slots the row to append glyphs (or missing glyphs) to `\g__fmuft_row_tl` to build up everything we need to finally typeset it.

```
        {
          \tl_gset:Nx \g__fmuft_row_tl
             {
               \exp_not:N \__fmuft_format_row_title:n
                   { \g__fmuft_hex_H_tl \g__fmuft_hex_A_tl
                     \g__fmuft_hex_B_tl \g__fmuft_hex_C_tl  }
             }
          \clist_map_function:NN \c__fmuft_hex_digits_clist
                                 \__fmuft_handle_hex_D:n
        }
    }
}
```

(*End of definition for* `\__fmuft_handle_hex_C:n`.)

`\__fmuft_format_row_title:n` The function to format the row title on the left, as used above.

```
\cs_new:Npn \__fmuft_format_row_title:n #1 {
  \texttt { \footnotesize \l__fmuft_color_tl \__fmuft_format_row_hex_digits:n {#1} }
}
```

(*End of definition for* `\__fmuft_format_row_title:n`.)

Key setup (ranges) For the range we have two keys, its start and the end. By default the whole range from 0 to FFFF is processed.

```
\tl_new:N \l__fmuft_range_start_tl
\tl_new:N \l__fmuft_range_end_tl

\keys_define:nn {__fmuft}
    {
      ,range-start .tl_set:N  = \l__fmuft_range_start_tl
      ,range-start .initial:n = 0000
      ,range-end   .tl_set:N  = \l__fmuft_range_end_tl
      ,range-end   .initial:n = FFFF
    }
```

uft_maybe_typeset_a_row_and_display_a_block_title:

The function handles the just-finished row and, if the row does not consist only of missing glyphs, typesets it. If necessary it also typesets a Unicode block name first.

```
\cs_new:Npn \__fmuft_maybe_typeset_a_row_and_display_a_block_title: {
```

We first check if the row had any real glyphs.

```
  \bool_if:NTF \g__fmuft_glyph_seen_bool
    {
```

If the row needs typesetting the fun part starts. We first look at the content of `\g__fmuft_block_title_tl`.

```
      \tl_if_empty:NTF \g__fmuft_block_title_tl
         {
```

It is empty we are in the middle of a block and we can ignore the Unicode title. However, we have to see if the previous row (or several) was missing (i.e., contained no glyphs). In that case we leave a little extra space, otherwise we just finish the previous row

```
            \bool_if:NTF \g__fmuft_row_missing_bool
                 { \__fmuft_debug_nl:n{A}\\[6pt] }
                 { \__fmuft_debug_nl:n{B}\\      }
         }
         {
```

Otherwise we first have to typeset the Unicode block title (or whatever should happen instead).

```
            \typeout{ Processing~ \tl_use:N \g__fmuft_block_title_tl }
            \bool_if:NTF \l__fmuft_display_block_bool
                 {
```

If we are to typeset the title the action depends a bit on whether we are at the very first row or typesetting a later block.

```
                   \bool_if:NTF \g__fmuft_first_row_bool
                     {
                       \bool_gset_false:N \g__fmuft_first_row_bool
                       \__fmuft_debug_nl:n{C}\\[-4pt]
                     }
                     {
                       \__fmuft_debug_nl:n{D}\\[8pt]
                       \noalign{\vskip 1pt plus 1pt} % space above block: customizable?
                     }
%                  \noalign{\smallskip} % space above block: customizable?
                   \multicolumn{17}{c}{\normalfont \bfseries
                                       \tl_use:N \g__fmuft_block_title_tl}
```

After the block title is typeset we may want to add a row of hex digits as well if that was requested, otherwise we only leave a bit of extra space.

```
                   \bool_if:NTF \l__fmuft_blockwise_hex_digits_bool
                        { \__fmuft_debug_nl:n{E}\\*
                          \__fmuft_display_row_of_hex_digits:
                          \__fmuft_debug_nl:n{H}\\*[2pt]
                        }
                        { \__fmuft_debug_nl:n{F}\\*[2pt] }
                 }
                 {
```

If the Unicode block title is not typeset we may still have to do someting special and again it differs if we at the very beginning of the table (because there we do nothing except changing the state of `\g__fmuft_first_row_bool`).

```
                \bool_if:NTF \g__fmuft_first_row_bool
                  { \bool_gset_false:N \g__fmuft_first_row_bool }
                  {
                    \__fmuft_debug_nl:n{G~ (new~ block)}
                    \l__fmuft_display_block_action_tl
                  }
              }
```

Once we are past the block title we clear it, so that it is not retypeset before the next row.

```
            \tl_gclear:N \g__fmuft_block_title_tl
          }
```

The final action is to typeset the row and reset the booleans (in case they were true; if they are false already then we do this unnecessarily, but that is probably faster than testing first).

```
      \bool_gset_false:N \g__fmuft_glyph_seen_bool
      \bool_gset_false:N \g__fmuft_row_missing_bool
      \tl_use:N \g__fmuft_row_tl
    }
```

Current row had no glyphs; remember that fact, and that is all we have to do in that case.

```
    {
      \bool_gset_true:N \g__fmuft_row_missing_bool
    }
}
```

(*End of definition for* `\__fmuft_maybe_typeset_a_row_and_display_a_block_title:`.)

5.5 Initialisation at the start of the table

`\g__fmuft_first_row_bool` `\g__fmuft_glyph_seen_bool` `\g__fmuft_row_missing_bool`

Declare the three booleans used in the code below. They will tell us answers to the following questions:

- Are we processing the first row?
- Have we seen any glyph so far (in the current row)?
- Did we have one or more missing rows recently?

```
\bool_new:N \g__fmuft_first_row_bool
\bool_new:N \g__fmuft_glyph_seen_bool
\bool_new:N \g__fmuft_row_missing_bool
```

(*End of definition for* `\g__fmuft_first_row_bool`, `\g__fmuft_glyph_seen_bool`, *and* `\g__fmuft_row_missing_bool`.)

`\__fmuft_initialize_table_rows:`

At the start of a table we are processing the first row and so we (obviously) haven't seen a glyph yet and there wasn't a missing row recently.

```
\cs_new:Npn \__fmuft_initialize_table_rows: {
    \bool_gset_true:N \g__fmuft_first_row_bool
    \bool_gset_false:N \g__fmuft_glyph_seen_bool
    \bool_gset_false:N \g__fmuft_row_missing_bool
```

And clearly the glyph count for the font(s) is zero.

```
    \int_gzero:N \g__fmuft_glyph_int
    \int_gzero:N \g__fmuft_glyph_only_B_int
    \int_gzero:N \g__fmuft_glyph_also_B_int
}
```

(*End of definition for* `\__fmuft_initialize_table_rows:`.)

5.6 Handling block titles

g__fmuft_block_title_tl We keep the current block title in this token list.

```
\tl_new:N \g__fmuft_block_title_tl
```

(*End of definition for* `g__fmuft_block_title_tl`.)

__fmuft_update_block_title:n A block title is updated when the hex digits A,B,C have a certain value, so this is nothing more than a huge case switch.

```
\cs_new:Npn \__fmuft_update_block_title:n #1 {
    \tl_gset:Nx \g__fmuft_block_title_tl  {
    \int_case:nnF{ "#1 }
       {
         { "000 }{ Basic~ Latin }
         { "008 }{ Latin-1~ Supplement }
         { "010 }{ Latin~ Extended-A }
         { "018 }{ Latin~ Extended-B }
         { "025 }{ IPA~ Extensions }
         { "02B }{ Spacing~ Modifier~ Letters }
         { "030 }{ Combining~ Diacritical~ Marks }
         { "037 }{ Greek~ and~ Coptic }
         { "040 }{ Cyrillic }
         { "053 }{ Armenian }
         { "059 }{ Hebrew }
         { "060 }{ Arabic }
         { "070 }{ Syriac }
         { "075 }{ Arabic~ Supplement }
         { "078 }{ Thaana }
         { "07C }{ NKo }
         { "090 }{ Devanagari }
         { "098 }{ Bengali }
         { "0A0 }{ Gurmukhi }
         { "0A8 }{ Gujarati }
         { "0B0 }{ Oriya }
         { "0B8 }{ Tamil }
         { "0C0 }{ Telugu }
         { "0C8 }{ Kannada }
         { "0D0 }{ Malayalam }
         { "0D8 }{ Sinhala }
         { "0E0 }{ Thai }
         { "0E8 }{ Lao }
         { "0F0 }{ Tibetan }
         { "100 }{ Myanmar }
         { "10A }{ Georgian }
         { "110 }{ Hangul~ Jamo }
         { "120 }{ Ethiopic  }
         { "138 }{ Ethiopic~ Supplement }
         { "13A }{ Cherokee }
         { "140 }{ Unified~ Canadian~ Aboriginal~ Syllabics }
         { "168 }{ Ogham }
         { "16A }{ Runic }
         { "170 }{ Tagalog }
         { "172 }{ Hanunoo }
         { "174 }{ Buhid }
         { "176 }{ Tagbanwa }
         { "178 }{ Khmer }
         { "180 }{ Mongolian }
         { "190 }{ Limbu }
         { "195 }{ Tai~ Le }
```

```
463         { "198 }{ New~ Tai~ Le }
464         { "19E }{ Khmer~ Symbols }
465         { "1A0 }{ Buginese }
466         { "1B0 }{ Balinese }
467         { "1D0 }{ Phonetic~ Extensions }
468         { "1D8 }{ Phonetic~ Extensions~ Supplement }
469         { "1DC }{ Combining~ Diacritical~ Marks~ Supplement }
470         { "1E0 }{ Latin~ Extended~ Additional }
471         { "1F0 }{ Greek~ Extended }
472         { "200 }{ General~ Punctuation }
473         { "207 }{ Superscripts~ and~ Subscripts }
474         { "20A }{ Currency~ Symbols }
475         { "20D }{ Combining~ Diacritical~ Marks~ for~ Symbols }
476         { "210 }{ Letterlike~ Symbols }
477         { "215 }{ Number~ Forms }
478         { "219 }{ Arrows }
479         { "220 }{ Mathematical~ Operators }
480         { "230 }{ Miscellaneous~ Technical }
481         { "240 }{ Control~ Pictures }
482         { "244 }{ Optical~ Character~ Recognition }
483         { "246 }{ Enclosed~ Alphanumerics }
484         { "250 }{ Box~ Drawing }
485         { "258 }{ Block~ Elements }
486         { "25A }{ Geometric~ Shapes }
487         { "260 }{ Miscellaneous~ Shapes }
488         { "270 }{ Dingbats }
489         { "27C }{ Miscellaneous~ Mathematical~ Symbols-A }
490         { "27F }{ Supplemental~ Arrows-A }
491         { "280 }{ Braille~ Patterns }
492         { "290 }{ Supplemental~ Arrows-B }
493         { "298 }{ Miscellaneous~ Mathematical~ Symbols-B }
494         { "2A0 }{ Supplemental~ Mathematical~ Operators }
495         { "2B0 }{ Miscellaneous~ Symbols~ and~ Arrows }
496         { "2C0 }{ Glagolitic }
497         { "2C6 }{ Latin~ Extended-C }
498         { "2C8 }{ Coptic }
499         { "2D0 }{ Georgian~ Supplement }
500         { "2D3 }{ Tifinagh }
501         { "2D8 }{ Ethiopic~ Extended }
502         { "2E0 }{ Supplemental~ Punctuation }
503         { "2E8 }{ CJK~ Radicals~ Supplement }
504         { "2F0 }{ Kangxi~ Radicals }
505         { "2FF }{ Ideographic~ Description~ Characters }
506         { "300 }{ CJK~ Symbols~ and~ Punctuation }
507         { "304 }{ Hiragana }
508         { "30A }{ Katakana }
509         { "310 }{ Bopomofo }
510         { "313 }{ Hangul~ Compatibility~ Jamo }
511         { "319 }{ Kanbun }
512         { "31A }{ Bopomofo~ Extended }
513         { "31C }{ CJK~ Strokes }
514         { "31F }{ Katakana~ Phonetic~ Extensions }
515         { "320 }{ Enclosed~ CJK~ Letters~ and~ Months }
516         { "330 }{ CJK~ Compatibility }
517         { "4DC }{ Yijing~ Hexagram~ Symbols }
518         { "A00 }{ Yi~ Syllables }
519         { "A49 }{ Yi~ Radicals }
520         { "A70 }{ Modifier~ Tone~ Letters }
521         { "A72 }{ Latin~ Extended-D }
```

```
          { "A80 }{ Syloti~ Nagri }
          { "A84 }{ Phags-pa }
          { "A88 }{ Saurashtra }
          { "A8E }{ Devanagari Extended }
          { "A90 }{ Kayah Li }
          { "A93 }{ Rejang }
          { "A96 }{ Hangul Jamo Extended-A }
          { "A98 }{ Javanese }
          { "A9E }{ Myanmar Extended-B }
          { "AA0 }{ Cham }
          { "AA6 }{ Myanmar Extended-A }
          { "AA8 }{ Tai Viet }
          { "AAE }{ Meetei Mayek Extensions }
          { "AB0 }{ Ethiopic Extended-A }
          { "AB3 }{ Latin Extended-E }
          { "AB7 }{ Cherokee Supplement }
          { "ABC }{ Meetei Mayek }
          { "AC0 }{ Hangul Syllables }
          { "D7B }{ Hangul Jamo Extended-B }
          { "D80 }{ High Surrogates }
          { "DB8 }{ High Private Use Surrogates }
          { "DC0 }{ Low Surrogates }
          { "E00 }{ Private~ Use~ Area }
          { "F90 }{ CJK~ Compatibility~ Ideographs }
          { "FB0 }{ Alphabetic~ Presentation~ Forms }
          { "FB5 }{ Arabic~ Presentation~ Forms-A }
          { "FE0 }{ Variation~ Selectors }
          { "FE1 }{ Vertical~ Forms }
          { "FE2 }{ Combining~ Half~ Marks  }
          { "FE3 }{ CJK~ Compatibility~ Forms }
          { "FE5 }{ Small~ Form~ Variants }
          { "FE7 }{ Arabic~ Presentation~ Forms-B }
          { "FF0 }{ Halfwidth~ and Fullwidth~ Forms }
          { "FFF }{ Specials~ ... }
%% ... Plane 1 ...
         { "1000 }{ Linear~ B~ Syllabary }
         { "1008 }{ Linear~ B~ Ideograms }
         { "1010 }{ Aegean~ Numbers }
         { "1014 }{ Ancient~ Greek~ Numbers }
         { "1019 }{ Ancient~ Symbols }
         { "101D }{ Phaistos~ Disc }
         { "1028 }{ Lycian }
         { "102A }{ Carian }
         { "102E }{ Coptic~ Epact~ Numbers }
         { "1030 }{ Old~ Italic }
         { "1033 }{ Gothic }
         { "1035 }{ Old~ Permic }
         { "1038 }{ Ugaritic }
         { "103A }{ Old~ Persian }
         { "1040 }{ Deseret }
         { "1045 }{ Shavian }
         { "1048 }{ Osmanya }
         { "104B }{ Osage }
         { "1050 }{ Elbasan }
         { "1053 }{ Caucasian~ Albanian }
         { "1060 }{ Linear~ A }
         { "1080 }{ Cypriot~ Syllabary }
         { "1084 }{ Imperial~ Aramaic }
         { "1086 }{ Palmyrene }
```

```
        { "1088 }{ Nabataean }
        { "108E }{ Hatran }
        { "1090 }{ Phoenician }
        { "1092 }{ Lydian }
        { "1098 }{ Meroitic~ Hieroglyphs }
        { "109A }{ Meroitic~ Cursive }
        { "10A0 }{ Kharoshthi }
        { "10A6 }{ Old~ South~ Arabian }
        { "10A8 }{ Old~ North~ Arabian }
        { "10AC }{ Manichaean }
        { "10B0 }{ Avestan }
        { "10B4 }{ Inscriptional~ Parthian }
        { "10B6 }{ Inscriptional~ Pahlavi }
        { "10B8 }{ Psalter~ Pahlavi }
        { "10C0 }{ Old~ Turkic }
        { "10C8 }{ Old~ Hungarian }
        { "10E6 }{ Rumi~ Numeral~ Symbols }
        { "1100 }{ Brahmi }
        { "1108 }{ Kaithi }
        { "110D }{ Sora~ Sompeng }
        { "1110 }{ Chakma }
        { "1115 }{ Mahajani }
        { "1118 }{ Sharada }
        { "111E }{ Sinhala~ Archaic~ Numbers }
        { "1120 }{ Khojki }
        { "1128 }{ Multani }
        { "112B }{ Khudawadi }
        { "1130 }{ Grantha }
        { "1140 }{ Newa }
        { "1148 }{ Tirhuta }
        { "1158 }{ Siddham }
        { "1160 }{ Modi }
        { "1166 }{ Mongolian~ Supplement }
        { "1168 }{ Takri }
        { "1170 }{ Ahom }
        { "118A }{ Warang~ Citi }
        { "11A0 }{ Zanabazar~ Square }
        { "11A5 }{ Soyombo }
        { "11AC }{ Pau~ Cin~ Hau }
        { "11C0 }{ Bhaiksuki }
        { "11C7 }{ Marchen }
        { "11D0 }{ Masaram~ Gondi }
        { "1200 }{ Cuneiform }
        { "1240 }{ Cuneiform~ Numbers~ and~ Punctuation }
        { "1248 }{ Early~ Dynastic~ Cuneiform }
        { "1300 }{ Egyptian~ Hieroglyphs }
        { "1440 }{ Anatolian~ Hieroglyphs }
        { "1680 }{ Bamum~ Supplement }
        { "16A4 }{ Mro }
        { "16AD }{ Bassa~ Vah }
        { "16B0 }{ Pahawh~ Hmong }
        { "16F0 }{ Miao }
        { "16FE }{ Ideographic~ Symbols~ and~ Punctuation }
        { "1700 }{ Tangut }
        { "1880 }{ Tangut~ Components }
        { "1B00 }{ Kana~ Supplement }
        { "1B10 }{ Kana~ Extended-A }
        { "1B17 }{ Nushu }
        { "1BC0 }{ Duployan }
```

```
        { "1BCA }{ Shorthand~ Format~ Controls }
        { "1D00 }{ Byzantine~ Musical~ Symbols }
        { "1D10 }{ Musical~ Symbols }
        { "1D20 }{ Ancient~ Greek~ Musical~ Notation }
        { "1D30 }{ Tai~ Xuan~ Jing~ Symbols }
        { "1D36 }{ Counting~ Rod~ Numerals }
        { "1D40 }{ Mathematical~ Alphanumeric~ Symbols }
        { "1D80 }{ Sutton~ SignWriting }
        { "1E00 }{ Glagolitic~ Supplement }
        { "1E80 }{ Mende~ Kikakui }
        { "1E90 }{ Adlam }
        { "1EE0 }{ Arabic~ Mathematical~ Alphabetic~ Symbols }
        { "1F00 }{ Mahjong~ Tiles }
        { "1F03 }{ Domino~ Tiles }
        { "1F0A }{ Playing~ Cards }
        { "1F10 }{ Enclosed~ Alphanumeric~ Supplement }
        { "1F20 }{ Enclosed~ Ideographic~ Supplement }
        { "1F30 }{ Miscellaneous~ Symbols~ and~ Pictographs }
        { "1F60 }{ Emoticons }
        { "1F65 }{ Ornamental~ Dingbats }
        { "1F68 }{ Transport~ and~ Map~ Symbols }
        { "1F70 }{ Alchemical~ Symbols }
        { "1F78 }{ Geometric~ Shapes~ Extended }
        { "1F80 }{ Supplemental~ Arrows-C }
        { "1F90 }{ Supplemental~ Symbols~ and~ Pictographs }
        { "2000 }{ CJK~ Unified~ Ideographs~ Extension~ B }
        { "2A70 }{ CJK~ Unified~ Ideographs~ Extension~ C }
        { "2B74 }{ CJK~ Unified~ Ideographs~ Extension~ D }
        { "2B82 }{ CJK~ Unified~ Ideographs~ Extension~ E }
        { "2CEB }{ CJK~ Unified~ Ideographs~ Extension~ F }
        { "2F80 }{ CJK~ Compatibility~ Ideographs~ Supplement }
        { "E010 }{ Tags }
        { "E000 }{ Variation~ Selectors~ Supplement }
        { "F000 }{ Supplementary~ Private~ Use~ Area-A }
% higher up not covered!
      }
```

If none of the above has matched we are somewhere within a block so we want keep the current name. However, since the case statement was executed within a `\tl_gset:Nx` we have to do this by passing the current block name back.

```
      { \tl_use:N \g__fmuft_block_title_tl }
  }
}
```

(*End of definition for* `\__fmuft_update_block_title:n`.)

Key setup (display blocks) The Unicode blocks may get indicated in different ways: with titles, only through rules, or not at all. Here is the necessary setup.

```
\bool_new:N \l__fmuft_display_block_bool
\tl_new:N   \l__fmuft_display_block_action_tl

\keys_define:nn {__fmuft}
    {
      ,display-block .choice:
      ,display-block / titles  .code:n =
        \bool_set_true:N \l__fmuft_display_block_bool
        \tl_set:Nn \l__fmuft_display_block_action_tl {\\}
      ,display-block / rules   .code:n =
        \bool_set_false:N \l__fmuft_display_block_bool
```

```
689         \tl_set:Nn \l__fmuft_display_block_action_tl {\\ \midrule}
690       ,display-block / none   .code:n =
691         \bool_set_false:N \l__fmuft_display_block_bool
692         \tl_set:Nn \l__fmuft_display_block_action_tl {\\}
693       ,display-block .initial:n  = titles
694   }
```

That's all of the programming using the L3 layer.

```
695 \ExplSyntaxOff
```

What remains is to require all packages needed …

```
696 \RequirePackage{longtable,booktabs,caption,fontspec}
```

…and executing all options passed to the package via `\usepackage`.

```
697 \ProcessKeysPackageOptions{__fmuft}
698 ⟨/package⟩
```

6 The standalone `unicodefont.tex` file

```
699 ⟨*standalone⟩
700 \documentclass{article}
701 \setlength\textwidth{470pt}
702 \setlength\oddsidemargin{0pt}
703 \addtolength\textheight{7\baselineskip}
704 \addtolength\topmargin{-3\baselineskip}
705 \usepackage{unicodefonttable}
706 \def\DEFAULTfontname{Latin Modern Roman}
707 \def\DEFAULTfontfeatures{}
708 \def\DEFAULTtableconfig{}
709 \def\DEFAULTunicodefont{}
710 \begin{document}
711 \typeout{^^J}
712 \ifx\generatetable\undefined
713 \else
714   \typein[\answer]{^^JReuse settings from last time (default yes)?^^J^^J%
715     [ font name = \DEFAULTfontname^^J
716       \space unicode? =  \ifx\DEFAULTunicodefont\empty yes^^J
717                            \space font features = \DEFAULTfontfeatures
718                          \else no\fi^^J
719       \space table config = \DEFAULTtableconfig \space]}
720 \fi
721 \ifx\answer\empty
722   \let\FontNameToTable\DEFAULTfontname
723   \let\IsUnicodeFont\DEFAULTunicodefont
724   \let\FontFeaturesToApply\DEFAULTfontfeatures
725   \let\TableConfigurationToApply\DEFAULTtableconfig
726 \else
727 \typein[\FontNameToTable]%
728        {^^JInput external font name as understood by fontspec, e.g.,^^J%
729          'TeX Gyre Pagella' or 'lmroman10-regular.otf'%
730          \ifx\DEFAULTfontname\empty\else
731             ^^J^^J[default \DEFAULTfontname]\fi:}
732 \ifx\FontNameToTable\empty \let\FontNameToTable\DEFAULTfontname \fi
733 \typein[\IsUnicodeFont]%
734        {^^JIs this a Unicode font?^^J^^J%
735         \ifx\DEFAULTunicodefont\empty  [default yes]\else [default no]\fi:}
736 \ifx\IsUnicodeFont\empty
```

```
%  \ifx\DEFAULTunicodefont\empty
%  \else
    \let\IsUnicodeFont\DEFAULTunicodefont
%  \fi
\else
  \ifx\DEFAULTunicodefont\empty
  \else
     \let\IsUnicodeFont\empty
  \fi
\fi
\ifx\IsUnicodeFont\empty
  \typein[\FontFeaturesToApply]%
         {^^JInput font feature key/value list to apply%
           \ifx\DEFAULTfontfeatures\empty\else
           ^^J^^J[default \DEFAULTfontfeatures]\fi:}
  \ifx\FontFeaturesToApply\empty \let\FontFeaturesToApply\DEFAULTfontfeatures \fi
\else
  \let\FontFeaturesToApply\DEFAULTfontfeatures
\fi
\typein[\TableConfigurationToApply]%
       {^^JInput table configuration key/value list to apply%
        \ifx\DEFAULTtableconfig\empty\else
          ^^J^^J[default
          \expandafter\detokenize\expandafter{\DEFAULTtableconfig}]\fi:}
\ifx\TableConfigurationToApply\empty
  \let\TableConfigurationToApply\DEFAULTtableconfig
\fi
\edef\generatetable{\noexpand\displayfonttable
  \ifx\IsUnicodeFont\empty\else *\fi
  \ifx\TableConfigurationToApply\empty\else
      [\expandafter\unexpanded\expandafter{\TableConfigurationToApply}]\fi
  {\FontNameToTable}%
  \ifx\FontFeaturesToApply\empty\else[\FontFeaturesToApply]\fi
}
\fi
\makeatletter
\protected@write\@auxout{}{\gdef\string\generatetable
     {\expandafter\detokenize\expandafter{\generatetable}}}
\protected@write\@auxout{}{\gdef\string\DEFAULTfontname{\FontNameToTable}}
\protected@write\@auxout{}{\gdef\string\DEFAULTunicodefont{\IsUnicodeFont}}
\protected@write\@auxout{}{\gdef\string\DEFAULTfontfeatures{\FontFeaturesToApply}}
\protected@write\@auxout{}{\gdef\string\DEFAULTtableconfig
                  {\expandafter\detokenize\expandafter{\TableConfigurationToApply}}}
\makeatother
\generatetable
\end{document}
```

⟨/standalone⟩

7 A samples file

⟨∗samples⟩

```
%<<VERBATIMLINE
%!TEX program = lualatex

%VERBATIMLINE
\documentclass{article}

\usepackage{xparse,color}
```

```
792
793 \usepackage{fontspec}
794
795 \setmainfont{Linux Biolinum O}
796 \setmonofont{SourceCodePro}
797
798 \usepackage{unicodefonttable}
799
800 \addtolength\textwidth{30pt}
801
802 \begin{document}
803
804 \listoftables
805
806 \section{Computer Modern  --- 8bit font}
807
808 \displayfonttable*[color=none,
809   range-end = 7F,
810 ]{cmr10}
811
812 %\section{Computer Modern Sans --- 8bit font} \displayfonttable*[]{cmss10}
813
814 \newpage
815
816 \section{TeX Gyre Heros (Helvetica)  --- 8bit font}
817
818 \displayfonttable*[color=red,nostatistics=false,
819   hex-digits = head+foot,
820   range-end = FF,
821 ]{ec-qhvr}
822
823 \newpage
824
825 \section{Latin Modern Sans --- OTF font}
826
827 \displayfonttable[
828 %  display-block = rules,
829   %              missing-glyph = \tiny\setlength\fboxsep{0pt}\fbox{$\times$},
830   hex-digits = block,
831   title-format-cont = \caption[]{\emph{continued}},
832 ]{Latin Modern Sans}
833
834 \newpage
835
836 \section{\TeX{} Gyre Pagella (Palatino) oldstyle figures --- OTF font}
837
838 \displayfonttable{TeX Gyre Pagella}[Numbers=OldStyle]
839
840 \newpage
841
842 \section{Comparing Latin Modern Math with New Computer Modern Math}
843
844 \displayfonttable[compare-with=NewCMMath-Regular.otf, range-end=1FFFF]
845                  {latinmodern-math.otf}
846
847 \end{document}
848 ⟨/samples⟩
```

Index

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

Symbols

A

B

C

D

E

F

T

U

V

◇ Frank Mittelbach
Mainz, Germany
https://www.latex-project.org
https://ctan.org/pkg/unicodefonttable