

cfr-lm

Clea F. Rees*

v2.0 (SVN Rev: 12139) 2026/10/19

Abstract

`cfr-lm` offers enhanced support for Latin Modern 2.004 in $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ on 8-bit engines¹. A number of features of the Latin Modern fonts are not easily accessible via the default $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ support provided in the official distribution. This package aims to provide $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ support for a number of these features including various styles of digits, upright italic and oblique small-caps shapes, and alternative weights and widths. It also supports the variable width typewriter, ‘dunhill’ and ‘quotation’ fonts. If `microtype` is loaded, the package ensures the custom settings designed for Computer and Latin Modern are loaded.

Contents

1	Introduction	2
2	Requirements	3
3	Limitations	3
4	Font setup	3
4.1	Font families	4
4.2	Shapes, Weights and Widths	4
5	The $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ package	4
6	Additional font commands	9
6.1	<code>nfssect-cfr</code>	9
6.1.1	Widths	10
6.1.2	Weights	10
6.1.3	Shapes	11
6.1.4	Figures	12
6.1.5	Typewriter variants	13
6.1.6	Latin Modern Sans Quotation	13
6.1.7	Latin Modern Roman Dunhill	14
6.2	<code>zeroslash</code>	14

*Bug tracker: codeberg.org/cfr/nfssect/issues | Code: codeberg.org/cfr/nfssect | Mirror: github.com/cfr42/nfssect

¹It is not required, does not support and should not be loaded with Unicode engines. Indeed, it was largely written because I wanted to use features easily accessible on Unicode engines while continuing to compile with pdf $\text{T}_{\text{E}}\text{X}$, which was significantly more reliable at the time and is still considerably faster.

7	Microtype	14
A	Installation	15
A.1	Install the files	15
A.2	Refresh the database	15
A.3	Install the map fragments	16
A.3.1	Method 1	16
A.3.2	Method 2: T _E X Live 2008 (and probably earlier)	16
A.3.3	Method 2: T _E X Live 2009 (and possibly later)	16
A.3.4	Method 3: Current/Recent T _E X Live	17
B	Implementation	17
B.1	L ^A T _E X 2 _ε package	17

1 Introduction

This document explains how to use the `cfr-lm` package to access advanced features of the Latin Modern fonts not otherwise supported by the official `lm` distribution. These features include various styles of digits, upright italic and oblique small-caps italic, alternative weights and widths, and Latin Modern Mono Prop (variable width typewriter), Dunhill and Sans Quotation. By default, the L^AT_EX package provided by `cfr-lm.sty` uses proportional oldstyle digits and variable width typewriter but this can be changed by passing appropriate options when loading the package. The package also supports using e.g. different styles of digits within a document so it is possible to use proportional oldstyle digits by default, say, but tabular lining digits within a particular table. Finally, a command to access the zeroslash character is provided.

`cfr-lm` version 1.3 requires version 2.004 of GUST's Latin Modern fonts, including the support package provided for T_EX. The fonts and T_EX support are included in many T_EX distributions or may be obtained from <http://www.gust.org.pl/projects/e-foundry/latin-modern> or your nearest CTAN mirror.

`cfr-lm` consists of all files listed in `manifest.txt` and these files are released under the L^AT_EX Project Public Licence as explained in the included licensing notices.

Version 1.3 of the package benefited greatly from feedback provided by Enrico Gregorio, who essentially rewrote the style file using `keyval` to show me how I ought to be setting the various options up, and Lars Hellström who demonstrated considerable patience in answering my many questions about using `fontinst` and some peculiarities of the Latin Modern fonts. I hope the changes in the production of the virtual fonts will improve accent placement in ‘faked’ glyphs (i.e. in the case of characters not included in the EC/T1 font encoding which T_EX therefore creates by combining glyphs which are included). The changes involve ignoring all font dimensions given in the AFM files and taking them from the relevant TFM files supplied with Latin Modern instead. The exception to this is the value of `accapheight` which is set to zero in the TFMs. The current virtual font setup uses `fontinst`'s default value in this case.

If you load `microtype`, version 1.4 and later will automatically figure out the family-specific settings to use. This is done using aliases which tell `microtype` to treat the virtual fonts provided by this package in the same way it treats Latin Modern and Computer Modern Roman. See section 7 for details.

2 Requirements

In addition to the usual suspects (L^AT_EX etc.), the L^AT_EX support provided by `cfr-lm.sty` requires:

- `lm`: Latin Modern version 2.004²
- `nfssex-cfr`

If you wish to compile (as opposed to read) the package documentation, additional, packages are required. See `cfr-lm.dtx` for details.

If you wish to recreate the font support files from the base `lm` package, the easiest option is to download the source from [CODEBERG](#) or [GITHUB](#). You can, however, also recreate the font files by hand using the sources included in the [CTAN](#) archive alone³.

3 Limitations

Unlike the official T_EX support for Latin Modern, `cfr-lm` supports only the EC/T1 and Text Companion (TS1) encodings for text. Also unlike the official support, the EC/T1 support depends entirely on virtual fonts. If virtual fonts have disadvantages, then, whatever those disadvantages may be, `cfr-lm` will inherit them. This does not apply to the TS1 encoding or to mathematics, since these rely purely on the support provided by the official distribution so should be identical.

L^AT_EX does not recognise the fonts provided by this package, including the TS1 encoding but excluding the mathematics, *as* Latin Modern. This is problematic because newer kernels treat Computer Modern and Latin Modern differently, but only if they are accessed using the default names. This causes at least two complications.

First, the kernel responds to `cfr-lm` far more ‘noisily’ than one would like, especially since the noise is entirely unnecessary. The warnings occur because L^AT_EX switches the default bold series from `bx` to `b` unless the document fonts are on a list which includes Latin Modern only by the names provided by `lmodern`. As far as I can tell, the ‘noise’ is merely annoying: the actual fonts used and the final output are unaffected, since the kernel tries `bx` if `b` is unavailable, though there is presumably some impact on compilation time. In any case, the new version of `nfssex-cfr` now patches the code the kernel executes at the beginning of the `document` environment by simply adding the appropriate names to the list of Computer and Latin Modern families.

Second, the virtual fonts provided by this package aren’t recognised as supporting the `ts1` encoding, so `cfr-lm` needs to specify this specifically on newer kernels⁴.

4 Font setup

As explained above, the fonts use the EC/T1 and Text Companion (TS1) encodings. The provision for the TS1 and mathematics encodings simply calls the support provided by `lm`. The `cfr-lm` support simply ensures that access is provided automatically when the T1-encoded virtual fonts it provides are active.

²This package should not be used with any other version of Latin Modern due to likely changes to the font metrics, glyph names etc.

³The font definition files will be functionally equivalent to those included in the package, but the spacing in some `\DeclareFontFamily` lines will differ because `fontinst` doesn’t write arguments passed to `\installfamily` verbatim to the output stream.. Other files should be equivalent modulo commented lines.

⁴On older kernels, the package continues to load `textcomp` as it did before.

Table 1: Font families

LM Names	Family	Digits/figures	Notes
Latin Modern Roman	clm	tabular, lining	similar to <code>lm rm</code> default
	clm2	proportional, lining	
	clmj	tabular, oldstyle	<code>cfr-lm rm</code> default
	clm2j	proportional, oldstyle	
Latin Modern Sans	clms	tabular, lining	similar to <code>lm sf</code> default
	clm2s	proportional, lining	
	clmjs	tabular, oldstyle	<code>cfr-lm sf</code> default
	clm2js	proportional, oldstyle	
Latin Modern Mono ^a	clmt, clm2t	tabular, lining	similar to <code>lm tt</code> default
	clmjt, clm2jt	tabular, oldstyle	
Latin Modern Mono Prop ^b	clmv	tabular, lining	<code>cfr-lm tt</code> default
	clm2v	proportional, lining	
	clmjb	tabular, oldstyle	
	clm2jb	proportional, oldstyle	
Latin Modern Sans Quotation	clmq	tabular, lining	
	clm2q	proportional, lining	
	clmqj	tabular, oldstyle	
	clm2jq	proportional, oldstyle	
Latin Modern Roman Dunhill	clmd	tabular, lining	
	clm2d	proportional, lining	
	clmdj	tabular, oldstyle	
	clm2dj	proportional, oldstyle	

^a The duplication in T_EX name here is to avoid T_EX complaining if commands to use proportional digits are issued while one of these fonts is active and to ensure that it is possible to switch smoothly to these fonts if another font with proportional digits is active.

^b Despite the apparent contradiction in their name, this is variable-width typewriter.

4.1 Font families

Table 1 list the font families provided for use in the EC/T1 and Text Companion (TS1) encodings.

4.2 Shapes, Weights and Widths

Shape, eight and width availability is shown in table 2.

Where applicable, oblique small-caps are substituted for italic small-caps; italic or oblique for upright italic; oblique for italic; and upright for small-caps. This means that some of the commands described in section 6 will fail silently to avoid undue clutter in the log file.

5 The L^AT_EX package

To load this package, write `\usepackage{cfr-lm}` in your document preamble. By default, the package will define `clm2j`, `clm2js` and `clm2jb` as the default roman/serif, sans and typewriter fonts but you can control the choice by passing options to the package.

The package recognises five keys summarised in table 3 and detailed below. Three of these keys take various options which take the value true or false. These control the default style of figures to be used for each of roman/serif, sans and typewriter text, and whether variable or monowidth typewriter will be used by default.

Table 2: Shapes, weights & widths

family	widths	weights	shapes
clm, clm2, clmj, clm2j	standard	normal	upright, oblique, italic, upright italic, small-caps, oblique small-caps
		bold demi	upright, oblique, italic upright, oblique
clms, clm2s, clmjs, clm2js	standard	normal bold	upright, oblique upright, oblique
	condensed	demi	upright, oblique
clmt, clm2t, clmjt, clm2jt	standard	normal	upright, oblique, italic, small-caps, oblique small-caps
		bold light	upright, oblique upright, oblique
	condensed	light	upright, oblique
clmv, clm2v, clmjv, clm2jv	standard	normal bold light	upright, oblique upright, oblique upright, oblique
clmq, clm2q, clmqj, clm2jq		normal bold	upright, oblique upright, oblique
clmd, clm2d, clmdj, clm2dj	standard	normal bold	upright, oblique upright, oblique

Table 3: Package options

key	affects	option	possible values
rm	oldstyle/osf ^a lining/lf ^a proportional/prop tabular/tab	true, false	default roman/serif figure style
	zero-slashed	true, false	slashed zero ^c
sf	oldstyle/osf ^a lining/lf ^a proportional/prop tabular/tab variable/var	true, false	default sans figure style
	zero-slashed	true, false	slashed zero ^c
tt	oldstyle/osf ^a lining/lf ^a proportional/prop tabular/tab	true, false	default typewriter figure style
	monowidth/mono variable/var	true, false	default typewriter family
	zero-slashed	true, false	slashed zero ^c
qt	—	true, false	nothing unless <code>\qtfont</code> is defined
maths math	maths	classic, lum, unicode-math, none	default is classic
nomaths ^d nomath ^d	maths	true, false	

^a Lining figures have zero depth i.e. they stand with their bottoms on the current baseline. Oldstyle (‘hanging’) figures may have depth as well as height i.e. they sit on the baseline with their bottoms hanging over the edge. These options are mutually exclusive and exhaustive.

^b Proportional figures have variable widths, depending on the widths of the digits e.g. ‘1’ is typically narrower than ‘6’. Tabular figures have standard, constant width i.e. ‘1’ is as wide as ‘6’, so there is typically more space on each side of ‘1’ than ‘6’. These options are mutually exclusive and exhaustive.

^c Has no effect on 8bit engines. Use `\zeroslash` instead.

^d Deprecated option, which should not be used in new code.

`rm (opt.) = <key-value list>`

Sets the default style of figures for roman (serif).

`rm/oldstyle (opt.) = true|false`

`rm/osf (opt.)` Whether to use oldstyle/hanging figures by default.

`rm/lining (opt.) = true|false`

`rm/lf (opt.)` Whether to use lining figures by default.

Note that `oldstyle` and `osf` are equivalent, while `lining` or `lf` sets the same option but inverted. That is, the following are equivalent:

```
rm={lining=true}
rm={lining}
rm={oldstyle=false}
rm={osf=false}
```

`rm/proportional (opt.) = true|false`

`rm/prop (opt.)` Whether to use proportional figures by default.

`rm/tabular (opt.) = true|false`

`rm/tab (opt.)` Whether to use tabular figures by default.

Note that `proportional` and `prop` are equivalent, while `tabular` or `tab` set the same option but inverted. That is, the following are equivalent:

```
rm={tabular=true}
rm={tabular}
rm={tab=true}
rm={tab}
rm={proportional=false}
rm={prop=false}
```

`rm/zero-slashed (opt.) = true|false`

On Unicode engines, determines whether to enable the slashed zero. The default is false. Dunhill uses the same setting.

`sf (opt.) = <key-value list>`

Set default figure style for sans serif.

The available keys and values are identical to those for serif explained above.

`sf/oldstyle (opt.) = true|false`

`sf/osf (opt.)` Whether to use oldstyle/hanging figures by default.

`sf/lining (opt.) = true|false`

`sf/lf (opt.)` Whether to use lining figures by default.

`sf/proportional (opt.) = true|false`

`sf/prop (opt.)` Whether to use proportional figures by default.

`sf/tabular (opt.) = true|false`

`sf/tab (opt.)` Whether to use tabular figures by default.

`sf/zero-slashed (opt.) = true|false`

Whether to enable the slashed zero on Unicode engines. If using quotation sans, the same setting will apply.

The available keys and values are identical to those for serif explained above.

`tt (opt.) = <key-value list>`

Set defaults for typewriter. These determine not only the default figure style, but also the default style of other characters.

The available keys and values for setting the default figure style are identical to those for serif explained above. The additional keys for choosing between variable- and mono-width typewriter are explained below.

```
tt/oldstyle (opt.) = true|false
  tt/osf (opt.) Whether to use oldstyle/hanging figures by default.

tt/lining (opt.) = true|false
  tt/lf (opt.) Whether to use lining figures by default.

tt/proportional (opt.) = true|false
  tt/prop (opt.) Whether to use proportional figures by default.

tt/tabular (opt.) = true|false
  tt/tab (opt.) Whether to use tabular figures by default.

tt/monowidth (opt.) = true|false
  tt/mono (opt.) Whether to use mono-width typewriter by default.

tt/variable (opt.) = true|false
  tt/var (opt.) Whether to use variable-width typewriter by default.
```

Note that `variable` and `var` are equivalent, while `monowidth` or `mono` set the same option but inverted. That is, the following are equivalent:

```
tt={monowidth=true}
tt={monowidth}
tt={mono=true}
tt={mono}
tt={variable=false}
tt={var=false}
```

```
tt/zero-slashed (opt.) = true|false
```

Whether to enable the slashed zero on Unicode engines. If using quotation sans, the same setting will apply.

```
qt (opt.) = true|false The fourth key itself takes a true or false value but has no effect unless \qtfont is
already defined5.
```

```
maths (opt.) = classic|lum|unicode-math|none
```

```
math (opt.) Whether to configure maths fonts at all and, if so, how. The only effective choices on 8 bit
engines are classic, which is the default, and none, which forgoes the configuration of maths
fonts altogether.
```

On Unicode engines, `unicode-math` loads that package and sets the default maths font to `latinmodern-math.otf`.

For Lua $\LaTeX$ 2 ε , `lum` loads `lum-lmodern` and configures the text maths alphabets to use Latin Modern opentype fonts.

```
nomaths (opt.) = true|false
```

Deprecated.

```
nomath (opt.) These keys are deprecated and temporarily retained for compatibility with existing documents.
Please use maths = none or math = none instead to disable maths font configuration; or use
maths = classic or math = classic to explicitly set the default, if desired.
```

⁵This key is designed to control use of LM Sans Quotation in conjunction with prior redefinitions of appropriate environments. Since this is not the sort of redefining a font package should be doing, the option will have absolutely no effect unless you do some prior work to make use of it. In any case, the font can still be accessed directly using the commands explained in section 6.

If `maths/math` is used without a value, the default `classic` is used. In all other cases, the default value is `true` if an option is given without a value.

For example, `rm={oldstyle=true}` is equivalent to `rm={oldstyle}`. Many of the options are provided for ease of use but are essentially equivalent. For example, `proportional=false` is equivalent to `tabular=true`. This means that the following two commands are equivalent:

```
\usepackage[%
  rm={lining=true,tabular=false},%
  sf={oldstyle,proportional},%
  tt={oldstyle=false,proportional=true,monowidth},%
  maths,%
]{cfr-lm}
```

```
\usepackage[%
  rm={oldstyle=false, proportional=true},%
  sf={lining=false,tabular=false},%
  tt={lining,proportional,variable=false},%
  maths=classic,%
]{cfr-lm}
```

Loading the package without options is equivalent to:

```
\usepackage[%
  rm={oldstyle=true,proportional=true},%
  sf={oldstyle=true,proportional=true},%
  tt={oldstyle=true,proportional=true,variable=true},%
  qt=false,%
  maths=classic%
]{cfr-lm}
```

That is, by default, `oldstyle`, `proportional` figures for roman, sans and typewriter text and `variable` width typewriter will be selected and fonts for mathematics will be configured as for the `lmodern` package.

6 Additional font commands

`cfr-lm` loads `nfssect-cfr` which is an extension and modification of the package `nfssect` supplied by Philipp Lehman as part of The Font Installation Guide. The file extends the font selection commands to facilitate access to various font features. Both the original and the extension are designed for use with a wide range of fonts. For this reason, only a subset of the additional commands are relevant to any particular font support package. Those relevant to `cfr-lm` are described below.

6.1 nfssect-cfr

These commands are available when `cfr-lm` is loaded. If for some reason you wish to make them available at any other time, use `\usepackage{nfssect-cfr}` in your document preamble.

Note that only combinations supported by the fonts will appear as expected because the commands will only have an effect if the active font offers the relevant variant. For example, trying to switch to a condensed width will have no effect if any of the LM Roman fonts is active. This means that only a subset of combinations are possible. In other cases, one of two things should happen. First, a ‘silent’ substitution may be made. For example, if you request `proportional` figures while using `monowidth` typewriter, `tabular` figures will be silently substituted. Second, console messages may warn you that the combination you tried to use isn’t available. If you request `titling` while using `monowidth` typewriter, a console message will warn you it was unavailable.

Table 4: Width macros

width	width command	text command
standard	<code>\regwidth</code>	<code>\textrw{}</code>
condensed	<code>\cdwidth</code>	<code>\textcd{}</code>

Table 5: Weight macros

weight	weight command	text command
light	<code>\lgweight</code>	<code>\textlg{}</code>
semi-bold	<code>\sbweight</code>	<code>\textsb{}</code>

The file `clm-test.tex` gives an idea of what’s possible and also serves as an example illustrating some of the commands provided by `cfr-lm` and other ways of accessing the fonts..

6.1.1 Widths

`\regwidth` Additional macros for changing width are listed in table 4. To switch to an condensed width until `\textrw` further notice, for example, you could use `\cdwidth`. Or use `\texttm{\textlg{\textcd{Hello, \cdwidth world!}}}` to typeset just the text Hello, world! in light-weight condensed monowidth typewriter. `\textcd` Note that the easiest way to switch to semi-bold condensed sans is to resort to using `\fontseries` directly.

```
\textsf{\fontseries{sbc}\selectfont Semi-bold condensed sans}
```

produces

Semi-bold condensed sans

The problem with using the commands provided by `nfssect-cfr` is that they are designed, like standard commands such as `\scshape`, to change *one* aspect of the font at a time⁶. Issuing `\textsf{\textcd{\textsb{}}}` and `\textsf{\textsb{\textcd{}}}` are equivalent to `\textsf{}` because neither standard-width semi-bold nor condensed normal-weight sans is available. The problem is that each command is processed independently, so both switches fail.

Similar considerations in the case of light condensed monowidth typewriter mean that the *order* in which commands are issued is critical. In this case, a light-weight standard-width font is available, but no normal-weight condensed font is provided. Consequently,

```
\texttm{\textlg{\textcd{a successful switch}}}
```

will produce a successful switch while

```
\texttm{\textcd{\textlg{an unsuccessful switch}}}
```

will result in an `unsuccessful switch` and a warning in the log. In this case, the latter command is equivalent to `\texttm{\textlg{}}` because `\textcd{}` can only succeed *after* `\textlg{}`.

6.1.2 Weights

Additional macros for changing the font weight are given in table 5.

```
\textsb{Semi-bold and \textsl{semi-bold oblique} serif}\
```

⁶This is, of course, by design. The problem with using standard commands such as `\bfseries` is that they are designed to change *two* aspects of the font at a time i.e. width and weight.

Table 6: Shape macros

shape	shape command	text command
oblique small-caps	<code>\slshape\scshape</code> ^a	<code>\textsl{\textsc{}}</code> ^a
	<code>\scshape\slshape</code> ^a	<code>\textsc{\textsl{}}</code> ^a
	<code>\itshape\scshape</code> ^b	<code>\textit{\textsc{}}</code> ^b
	<code>\scshape\itshape</code> ^b	<code>\textsc{\textit{}}</code> ^b
	<code>\sishape</code> ^b	<code>\textsi{}</code> ^b
upright italic	<code>\uishape</code>	<code>\textui{}</code>

^a Supported for all versions of L^AT_EX 2_ε.

^b Actually the command switches to *italic* small-caps but since LM does not offer this, oblique small-caps are substituted. Unlike their upper/lower-case cousins, small-caps generally look the same whether the font designer calls them ‘italic small-caps’ or ‘pblque small-caps’, so the substitution is in no way second-best in this case.

```
\texttt{\textlg{Light typewriter}}
```

produces:

Semi-bold and *semi-bold oblique* serif
Light typewriter

6.1.3 Shapes

```
\sishape Extended shape-changing macros are listed in table 6.
\textsi
\uishape \textsc{\textsl{I \emph{always} avoid a kangaroo.}}\
\textui \textsc{\textit{I \emph{always} avoid a kangaroo.}}\
\textsl{\textsc{I \emph{always} avoid a kangaroo.}}\
\textit{\textsc{I \emph{always} avoid a kangaroo.}}\
\textsi{I \emph{always} avoid a kangaroo.}\
\textui{Nobody is despised who can manage a crocodile.}
```

produces:

I ALWAYS AVOID A KANGAROO.
I ALWAYS AVOID A KANGAROO.
I ALWAYS AVOID A KANGAROO.
I ALWAYS AVOID A KANGAROO.
I ALWAYS AVOID A KANGAROO.
 Nobody is despised who can manage a crocodile.

if oblique small-caps/upright italic is available for the active font. If it is not, another shape will be substituted.

```
\textsf{\textsc{\slshape The bit about the kangaroo was from Lewis Carroll.}}\
\textbf{\textui{Sylvia snorkeled snappily.}}
```

produces only:

The bit about the kangaroo was from Lewis Carroll.
Sylvia snorkeled snappily.

where upright sans and bold italic are substituted for italic small-caps sans and bold upright italic since neither is available. Note that the first substitution produces a warning in the log while the second is done ‘silently’.

Table 7: Macros for changing the style of figures.

figure style	style command	text command
lining ^a	<code>\lstyle</code>	<code>\textl{}</code>
oldstyle ^b	<code>\ostyle</code>	<code>\texto{}</code>
proportional ^c	<code>\pstyle</code>	<code>\textp{}</code>
tabular ^d	<code>\tstyle</code>	<code>\texttt{}</code>
proportional ^c , lining ^a	<code>\plstyle</code>	<code>\textpl{}</code>
proportional ^c , oldstyle ^b	<code>\postyle</code>	<code>\textpo{}</code>
tabular ^d , lining ^a	<code>\tlstyle</code>	<code>\textttl{}</code>
tabular ^d , oldstyle ^b	<code>\tostyle</code>	<code>\textto{}</code>

^a lining figures stand on the current baseline: 0123456789. They are generally preferable for use in tabulars, mathematics, code listings, diagrams etc. Contemporary usage also favours them in text, even though traditional typography would frown on this.

^b oldstyle figures sit on or hang from the current baseline: 0123456789. They are generally considered more suitable for use in text than lining figure.

^c proportional figures take up space in proportion to the actual width of the digit: 0123456789. These are generally preferable in most non-specialised contexts.

^d tabular figures each take up a standard width, regardless of the width of the digit: 0123456789. These are better in tabulars where columns of digits should be aligned, code listings set in monowidth typewriter (as is usual) etc.

6.1.4 Figures

`\lstyle` Commands are provided to change either one or both aspects of digits' style (table 7). The macros `\ostyle` all use standard abbreviations and have predictable forms. Any macro of the form `\langle\theta\rangle style` switches a single aspect of the current digits' style. $\langle\theta\rangle$ may be one of `l`, `o`, `p` or `t` which represent `\tstyle` lining, `oldstyle`, `proportional` and `tabular` respectively.

`\textl` The corresponding text commands have the format `\text\langle\theta\rangle`, where $\langle\theta\rangle$ may take the same values `\texto` as before.

`\textp` In this document, proportional lining figures are used by default for roman/serif and sans, while `\texttt` tabular lining are used for typewriter:

```
0123456789
0123456789
0123456789
```

but oldstyle figures are also accessible. For example:

```
\texto{0123456789}\
\textsf{\texto{0123456789}}\
\texttt{\texto{0123456789}}
```

produces:

```
0123456789
0123456789
0123456789
```

First, note that it is necessary to reissue `\texto{}` after switching to sans or typewriter text. This is because both switching to sans or typewriter and switching to another figure style involves a switch of font family⁷.

⁷Compare a switch in width or weight which does not typically involve a change of active font family.

Table 8: Macros for switching to mono-/variable-width typewriter

typewriter font	style command	text command
variable typewriter	<code>\tvstyle</code>	<code>\texttv{}</code>
monowidth typewriter	<code>\tmstyle</code>	<code>\texttm{}</code>

Table 9: LM Sans Quotation

sans quotation	<code>\qtstyle</code>	<code>\textqt{}</code>
----------------	-----------------------	------------------------

Second, note that the output shows *proportional* oldstyle figures for romand and sans, but *tabular* oldstyle for typewriter, because the command `\texto{}` only changes *one* aspect of the style. Because proportional figures were already active for serif and sans, the command switched to proportional oldstyle figures in the first two cases. Contrariwise, since tabular figures were active for typewriter, the same command switched to tabular oldstyle figures in the third case.

In many cases, it is convenient to switch or ensure both aspects of digits together e.g. to ensure tabular lining figures are used in tabulars. Four macros are provided for this purpose. These have the form `\langle\theta\rangle\langle\beta\rangle\textstyle`, where $\langle\theta\rangle$ may be either `p` or `t` and $\langle\beta\rangle$ may be either `l` or `o`. The corresponding text commands have the format `\text\langle\theta\rangle\langle\beta\rangle`.

Taking roman as an example, tabular oldstyle digits may be accessed in several ways:

```

\textpl
\textpo
\texttl
\textto
\texttt{0123456789}

```

will produce three identical lines of figures:

```

0123456789
0123456789
0123456789

```

6.1.5 Typewriter variants

In addition to the package options to specify either LM Mono or LM Mono Prop as default (i.e. either monowidth or variable-width typewriter), it is possible to access the non-default font using the commands in table 8.

Mono-width is default in this document so

```

\texttt{This is monowidth width typewriter.}\n
\texttv{This is variable typewriter} \texttm{except this bit at the end.}

```

produces:

```

This is monowidth width typewriter.
This is variable typewriter except this bit at the end.

```

6.1.6 Latin Modern Sans Quotation

Latin Modern Sans Quotation can be accessed using the macros listed in table 9.

For example, `\textqt{some text in the font}` will produce some text in the font.

Table 10: Latin Modern Roman Dunhill

style	style command	text command
titling	<code>\tistyle</code>	<code>\textti{}</code>

6.1.7 Latin Modern Roman Dunhill

`\textti` Latin Modern Roman Dunhill can be accessed using the macros listed in table 10.
`\tistyle` To ensure the command succeeds independently of the currently active font, you may wish to issue `\normalfont` first. For example:

```
\normalfont\textti{Kinky Querulous Rhinos X-Ray Exultant Risque Zebras}\\
\textti{\textsl{Kinky Querulous Rhinos X-Ray Exultant Risque Zebras}}
```

produces:

Kinky Querulous Rhinos X-Ray Exultant Risque Zebras
Kinky Querulous Rhinos X-Ray Exultant Risque Zebras

6.2 zeroslash

`\zeroslash` `cfr-lm` provides one additional command. On 8bit engines, `\zeroslash` will produce the `Ø` character from the current font. The behaviour is undefined on Unicode engines, where the `zero-slashed` package option should be used instead.

7 Microtype

Support for family-specific microtypographical features supported by `microtype`. This code will do nothing if you do not load `microtype`. If you do use these features, all regular roman and sans families, together with the sans quotation font, will use the settings for Computer Modern Roman. The fallback generic settings will continue to be applied to the typewriter and ‘dunhill’ families.

A Installation

The vast majority of users should **IGNORE** this section entirely. `cfr-lm` is included in all major $\text{\TeX}$ distributions and should be installed as part of your $\text{\TeX}$ installation. Installing the package yourself should be done only as a last resort or an educational exercise.

Note, in particular, that this version of `cfr-lm` should **not** be installed on older $\text{\LaTeX}$ kernels as it is designed to work with the (New) New Font Selection Scheme, as updated around 2020⁸. Use the initial release of `cfr-lm` if your installation of $\text{\LaTeX}$ predates those changes.

Installation varies with $\text{\TeX}$ distribution so you should consult the documentation which came with your system for details. In most cases, you will need to perform three steps:

1. move or copy the package files to appropriate locations on your system;
2. refresh the $\text{\TeX}$ database;
3. incorporate the included map file fragments for the different engines your distribution supports.

The following instructions assume you are using $\text{\TeX}$ Live⁹. They should not be too difficult to adapt if you are using a different distribution.

A.1 Install the files

The files should be installed in one of two locations: *either* the local system-wide $\text{\TeX}$ tree *or* your personal tree. If the package is installed system-wide, all users will have access to it. On the other hand, you may need privileges you do not have to do this in which case you must use your personal tree.

There are serious disadvantages to installing the package into your personal tree. In particular, these pertain to use of `updmap -user` rather than `updmap -sys`. If you are not aware of these disadvantages, please ensure you are fully cognisant of them before proceeding¹⁰. Merely removing the package from your personal tree at a later point will *not* undo the effects.

For $\text{\TeX}$ Live, `kpsewhich -var-value TEXMFLOCAL` will return the path to the local tree and `kpsewhich -var-value TEXMFHOME` the path to your personal tree. The package already includes a hierarchy of files to help you install them correctly. Ignoring any symbolic link in the top directory, move or copy the files in `doc`, `fonts` and `tex` into the appropriate locations. If the tree is initially empty, you can simply move or copy the directories in as they are. If the tree already contains other packages, you may need to merge the package hierarchy with the pre-existing one. For example, if you already have a `doc/fonts` directory, move or copy `doc/fonts/cfr-lm` into `doc/fonts/`. If you have a `doc` directory but not a `doc/fonts`, move `doc/fonts` into `doc/`.

A.2 Refresh the database

Again, this depends on your distribution. For $\text{\TeX}$ Live, `mktexlsr <path to directory>` for the directory you used in the first step should do the trick. Note that you *may* be able to skip this step if you install into your personal tree. Whether this is so depends on the details of your set-up. As a test, move to a directory containing none of the package files and try `kpsewhich cfr-lm.sty`.

⁸The package should™ work fine on older kernels, but the new version is bound to have some bugs and there is no reason to use it on these systems. The sole purpose of the update is to accommodate the breaking changes made to font selection. If you don't have those changes installed locally, nothing should be broken and the newer version of `cfr-lm` offers no advantage at all.

⁹This includes Mac $\text{\TeX}$ for OS X users.

¹⁰See, for example, [Why shouldn't I use `getnonfreefonts` to install additional fonts? Why shouldn't I use `updmap` when installing or removing fonts?](#).

If the file is found, you don't need to refresh the database; otherwise use `mktexlsr` and then try again.

A.3 Install the map fragments

For T_EX Live, there are at least two ways of doing this. The second method varies according to the version of T_EX Live and instructions are provided accordingly. Both methods depend on whether you installed into `TEXMFLOCAL` or `TEXMFHOME`. If you installed system-wide, the choice is relatively straightforward — it obviously makes sense in that case to update the font maps system-wide as well.

If, on the other hand, you installed into your personal tree, the matter is more complex. On the one hand, updating the system-wide maps may create difficulties or confusion for other users because while the map files will list the fonts as available, they will not be able to access them. On the other hand, maintaining personal font map files can produce difficulties and confusions of its own¹¹. Whether it is to be preferred or not is a complex issue and depends on the details of your T_EX distribution, local configuration and personal preference. The one clear case is that in which you install into your personal tree because you lack the privileges needed to install system-wide. In that case, you have no choice but to maintain personal font map files or forgo the use of all fonts not provided by your administrator. Other cases are thankfully beyond the scope of this document.

A.3.1 Method 1

If you installed the package system-wide, use the command:

```
updmap-sys --enable Map=clm.map
```

If you installed the package in your personal tree, you *may* prefer¹¹:

```
updmap --enable Map=clm.map
```

Either way, `updmap` will output a good deal of information after each incantation. This is normal. Just check that it does not end with an error and that it found the new map file.

A.3.2 Method 2: T_EX Live 2008 (and probably earlier)

If you installed the package system-wide, use `updmap-sys --edit`.

If you installed into your personal tree, you *may* prefer to use `updmap --edit`¹¹.

Either way, a configuration file will be opened which you can edit. Move to the end of the file and add the following line:

```
Map clm.map
```

When you are done, save the file. `updmap` or `updmap-sys` will produce a great deal of output if all is well. Just check that it does not end with an error and that `clm.map` is found.

A.3.3 Method 2: T_EX Live 2009 (and possibly later)

If you installed the package system-wide, edit or create `TEXMFLOCAL/web2c/updmap-local.cfg` and add the following line to the end of the file:

```
Map clm.map
```

¹¹See, for example, [Why shouldn't I use `getnonfreefonts` to install additional fonts? Why shouldn't I use `updmap` when installing or removing fonts?](#).

Save the file and tell `tlmgr` to merge in your addition using the command:

```
tlmgr generate updmap
```

`tlmgr` will then tell you that you need to ensure the changes are propagated correctly by calling `updmap-sys`. This should produce a great deal of output. Check that it finds the new map file and does not end with an error.

If you installed into your personal tree, you *may* prefer to use `updmap --edit` as described above for T_EX Live 2008¹².

A.3.4 Method 3: Current/Recent T_EX Live

If you installed the package system-wide, tell `\updmap` to enable the map file:

```
updmap --sys --enable Map=clm.map
```

This should produce a great deal of output. Check that it finds the new map file and does not end with an error.

If you installed into your personal tree, you *could* use `updmap --user` in place of `updmap --sys` as described above for T_EX Live 2008, but this is **not** recommended¹².

To test your installation and that the package works on your system, latex this file (`cfr-lm.tex`). The console output and/or log should tell you whether any fonts were not found. If you are careful not to overwrite it, you may also compare your output with `cfr-lm.pdf`.

B Implementation

You do not need to read the remainder of this document in order to install or use the fonts.

B.1 L^AT_EX 2_ε package

`cfr-lm.sty` (*pkg.*) The L^AT_EX user interface.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \RequirePackage{svn-prov}
3 \ProvidesPackageSVN[\filebase.sty]{$Id: cfr-lm.dtx 12139 2026-10-19 06:54:17Z cfrees $}[v2.0
  \revinfo][Extended support for Latin Modern 2.004]
4 \DefineFileInfoSVN[clm]
5 \RequirePackage{nfssex-cfr}[2024/01/01]

6 \ExplSyntaxOn
7 \tl_new:N \g__clm_enc_tl
8 \sys_if_engine_opentype:TF {
9   \tl_gset:Nn \g__clm_enc_tl {TU}
10 } {
11   \RequirePackage[T1]{fontenc}
12   \tl_gset:Nn \g__clm_enc_tl {T1}
13 }
14 \ExplSyntaxOff
```

`nfssex-cfr` provides `\ProcessKeyOptions`, `\IfFormatAtLeastTF` on older kernels and ensures font encoding subset declarations are valid in font definition files if the format is `post=2020` but not new enough to support them out-of-the-box. The declarations themselves are inserted by `l3build` using `fontscript's` `fntbuild` and defined in `build.lua`¹³. Note that these declarations do **not**

¹²See, for example, [Why shouldn't I use getnonfreefonts to install additional fonts? Why shouldn't I use updmap when installing or removing fonts?](#).

¹³I've been told this shouldn't be on CTAN, so it isn't, but you can find it on GITHUB.

conform to the recommendations provided by the L^AT_EX Project’s helper script. Instead, they pick-up the declarations used by the L^AT_EX Project for Latin Modern. This is done dynamically, so if the developers ever decide to use their script’s recommendations for Latin Modern, any changes will be automatically picked up¹⁴.

```

15 \IfFormatAtLeastTF {2020-02-02}{%
16 %^A To get the oldstyle numbers etc.\ used from TS1, we need to set the subset to 0 or
17 %^A 1.
18 %^A We follow the kernel’s handling of Latin Modern and eschew the builtin circle.
19 \IfFormatAtLeastTF {2025-06-01}{%
20 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm}{TS1:lmr}%
21 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2}{TS1:lmr}%
22 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2d}{TS1:lmr}%
23 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2dj}{TS1:lmr}%
24 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2j}{TS1:lmr}%
25 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2jqs}{TS1:lmss}%
26 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2js}{TS1:lmss}%
27 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2jt}{TS1:lmss}%
28 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2jv}{TS1:lmss}%
29 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2qs}{TS1:lmss}%
30 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2s}{TS1:lmss}%
31 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2t}{TS1:lmss}%
32 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clm2v}{TS1:lmss}%
33 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmd}{TS1:lmss}%
34 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmdj}{TS1:lmss}%
35 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmj}{TS1:lmss}%
36 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmjqs}{TS1:lmss}%
37 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmjs}{TS1:lmss}%
38 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmjt}{TS1:lmss}%
39 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmjt}{TS1:lmss}%
40 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmjt}{TS1:lmss}%
41 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmjt}{TS1:lmss}%
42 \ExpandArgs{nnc}\DeclareEncodingSubset{TS1}{clmjt}{TS1:lmss}%
43 %^A \DeclareEncodingSubset{TS1}{clm}{1}%
44 %^A \DeclareEncodingSubset{TS1}{clm2}{1}%
45 %^A \DeclareEncodingSubset{TS1}{clm2d}{1}%
46 %^A \DeclareEncodingSubset{TS1}{clm2dj}{1}%
47 %^A \DeclareEncodingSubset{TS1}{clm2j}{1}%
48 %^A \DeclareEncodingSubset{TS1}{clm2jqs}{1}%
49 %^A \DeclareEncodingSubset{TS1}{clm2js}{1}%
50 %^A \DeclareEncodingSubset{TS1}{clm2jt}{1}%
51 %^A \DeclareEncodingSubset{TS1}{clm2jv}{1}%
52 %^A \DeclareEncodingSubset{TS1}{clm2qs}{1}%
53 %^A \DeclareEncodingSubset{TS1}{clm2s}{1}%
54 %^A \DeclareEncodingSubset{TS1}{clm2t}{1}%
55 %^A \DeclareEncodingSubset{TS1}{clm2v}{1}%
56 %^A \DeclareEncodingSubset{TS1}{clmd}{1}%
57 %^A \DeclareEncodingSubset{TS1}{clmdj}{1}%
58 %^A \DeclareEncodingSubset{TS1}{clmj}{1}%
59 %^A \DeclareEncodingSubset{TS1}{clmjqs}{1}%
60 %^A \DeclareEncodingSubset{TS1}{clmjs}{1}%
61 %^A \DeclareEncodingSubset{TS1}{clmjt}{1}%
62 %^A \DeclareEncodingSubset{TS1}{clmjt}{1}%
63 %^A \DeclareEncodingSubset{TS1}{clmjt}{1}%
64 %^A \DeclareEncodingSubset{TS1}{clmjt}{1}%
65 %^A \DeclareEncodingSubset{TS1}{clmjt}{1}%

```

¹⁴On the other hand, if the implementation details of subset handling in the format change, the code in this package may break. But that’s already true for so much of the code in `nfssect/nfssect-cfr`, that one more area of fragility probably matters but little.

```

66 %\A \DeclareEncodingSubset{TS1}{clmv}{1}%
67 }%
68 }{%
69 \RequirePackage{textcomp}}
70 \UndeclareTextCommand{\textperthousand}{T1}
71 \ExplSyntaxOn

```

Parts of this file are based on `lmodern.sty` which is included with the Latin Modern fonts released by GUST and available from <http://www.gust.org.pl/projects/e-foundry/latin-modern>.

This draws also on the documentation for the `microtype` package and `MinionPro.sty`. `MinionPro.sty` is available as part of the `minionpro` package and can be obtained from <http://mirror.ctan.org/fonts/minionpro>. `MinionPro.sty` is in the public domain. The documentation for `microtype` is available in English and German from [microtype](#). It is part of the `microtype` package which is itself licensed under the LPPL.

Since removed?

BEGIN bools

```

72 \bool_new:N \l__clm_rm_osf_bool
73 \bool_new:N \l__clm_rm_prop_bool
74 \bool_new:N \l__clm_rm_zero_bool
75 \bool_new:N \l__clm_sf_osf_bool
76 \bool_new:N \l__clm_sf_prop_bool
77 \bool_new:N \l__clm_sf_zero_bool
78 \bool_new:N \l__clm_tt_osf_bool
79 \bool_new:N \l__clm_tt_prop_bool
80 \bool_new:N \l__clm_tt_mono_bool
81 \bool_new:N \l__clm_tt_zero_bool
82 \bool_new:N \l__clm_qt_bool
83 \bool_new:N \l__clm_classicmaths_bool
84 \bool_set_true:N \l__clm_classicmaths_bool
85 \bool_set_false:N \l__clm_lum_bool

```

END bools

```

86 \keys_define:nm { clm }
87 {

```

`rm/oldstyle` (*opt.*) boolkey for roman osf/lf

```

88 rm / oldstyle .bool_set:N = \l__clm_rm_osf_bool,
89 rm / oldstyle .default:n = true,
90 rm / oldstyle .initial:n = true,

```

`rm/osf` (*opt.*) Shorthand

```

91 rm / osf .bool_set:N = \l__clm_rm_osf_bool,
92 rm / osf .default:n = true,

```

`rm/lining` (*opt.*) Inverse

```

93 rm / lining .bool_set_inverse:N = \l__clm_rm_osf_bool,
94 rm / lining .default:n = true,

```

`rm/lf` (*opt.*) Shorthand

```

95 rm / lf .bool_set_inverse:N = \l__clm_rm_osf_bool,
96 rm / lf .default:n = true,

```

`rm/proportional` (*opt.*) boolkey for roman prop/tab figures

```

97  rm / proportional .bool_set:N = \l__clm_rm_prop_bool,
98  rm / proportional .default:n = true,
99  rm / proportional .initial:n = true,

```

`rm/prop` (*opt.*) Shorthand

```

100 rm / prop .bool_set:N = \l__clm_rm_prop_bool,
101 rm / prop .default:n = true,

```

`rm/tabular` (*opt.*) Inverse

```

102 rm / tabular .bool_set_inverse:N = \l__clm_rm_prop_bool,
103 rm / tabular .default:n = true,

```

`rm/tab` (*opt.*) Shorthand

```

104 rm / tab .bool_set_inverse:N = \l__clm_rm_prop_bool,
105 rm / tab .default:n = true,

```

`rm/zero` (*opt.*) Slashed zero

```

106 rm / zero-slashed .bool_set:N = \l__clm_rm_zero_bool,
107 rm / zero-slashed .default:n = true,

```

`sf/oldstyle` (*opt.*) boolkeys for sans osf/lf

```

108 sf / oldstyle .bool_set:N = \l__clm_sf_osf_bool,
109 sf / oldstyle .default:n = true,
110 sf / oldstyle .initial:n = true,

```

`sf/osf` (*opt.*) Shorthand

```

111 sf / osf .bool_set:N = \l__clm_sf_osf_bool,
112 sf / osf .default:n = true,

```

`sf/lining` (*opt.*) Inverse

```

113 sf / lining .bool_set_inverse:N = \l__clm_sf_osf_bool,
114 sf / lining .default:n = true,

```

`sf/lf` (*opt.*) Shorthand

```

115 sf / lf .bool_set_inverse:N = \l__clm_sf_osf_bool,
116 sf / lf .default:n = true,

```

`sf/proportional` (*opt.*) boolkeys for sans prop/tab figures

```

117 sf / proportional .bool_set:N = \l__clm_sf_prop_bool,
118 sf / proportional .default:n = true,
119 sf / proportional .initial:n = true,

```

`sf/prop` (*opt.*) Shorthand

```

120 sf / prop .bool_set:N = \l__clm_sf_prop_bool,
121 sf / prop .default:n = true,

```

`sf/tabular` (*opt.*) Inverse

```

122 sf / tabular .bool_set_inverse:N = \l__clm_sf_prop_bool,
123 sf / tabular .default:n = true,

```

`sf/tab` (*opt.*) Shorthand

```
124 sf / tab .bool_set_inverse:N = \l__clm_sf_prop_bool,
125 sf / tab .default:n = true,
```

`sf/zero` (*opt.*) Slashed zero

```
126 sf / zero-slashed .bool_set:N = \l__clm_sf_zero_bool,
127 sf / zero-slashed .default:n = true,
```

`tt/oldstyle` (*opt.*) boolkeys for typewriter osf/lf

```
128 tt / oldstyle .bool_set:N = \l__clm_tt_osf_bool,
129 tt / oldstyle .default:n = true,
130 tt / oldstyle .initial:n = true,
```

`tt/osf` (*opt.*) Shorthand

```
131 tt / osf .bool_set:N = \l__clm_tt_osf_bool,
132 tt / osf .default:n = true,
```

`tt/lining` (*opt.*) Inverse

```
133 tt / lining .bool_set_inverse:N = \l__clm_tt_osf_bool,
134 tt / lining .default:n = true,
```

`tt/lf` (*opt.*) Shorthand

```
135 tt / lf .bool_set_inverse:N = \l__clm_tt_osf_bool,
136 tt / lf .default:n = true,
```

`tt/proportional` (*opt.*) boolkeys for typewriter prop/tab figures

```
137 tt / proportional .bool_set:N = \l__clm_tt_prop_bool,
138 tt / proportional .default:n = true,
139 tt / proportional .initial:n = true,
```

`tt/prop` (*opt.*) Shorthand

```
140 tt / prop .bool_set:N = \l__clm_tt_prop_bool,
141 tt / prop .default:n = true,
```

`tt/tabular` (*opt.*) Inverse

```
142 tt / tabular .bool_set_inverse:N = \l__clm_tt_prop_bool,
143 tt / tabular .default:n = true,
```

`tt/tab` (*opt.*) Shorthand

```
144 tt / tab .bool_set_inverse:N = \l__clm_tt_prop_bool,
145 tt / tab .default:n = true,
```

`tt/monowidth` (*opt.*) boolkeys for typewriter mono/variable width

```
146 tt / monowidth .bool_set:N = \l__clm_tt_mono_bool,
147 tt / monowidth .default:n = true,
148 tt / monowidth .initial:n = false,
```

`tt/mono` (*opt.*) Shorthand

```
149 tt / mono .bool_set:N = \l__clm_tt_mono_bool,
150 tt / mono .default:n = true,
```

`tt/variable (opt.)` Inverse

```
151 tt / variable .bool_set_inverse:N = \l__clm_tt_mono_bool,
152 tt / variable .default:n = true,
```

`tt/var (opt.)` Shorthand

```
153 tt / var .bool_set_inverse:N = \l__clm_tt_mono_bool,
154 tt / var .default:n = true,
```

`sf/zero (opt.)` Slashed zero

```
155 tt / zero-slashed .bool_set:N = \l__clm_tt_zero_bool,
156 tt / zero-slashed .default:n = true,
```

`rm (opt.)` options for roman

```
157 rm .code:n = {
158   \keys_set:nn { clm / rm } { #1 }
159 },
160 rm .value_required:n = true,
161 rm .usage:n = load,
```

`sf (opt.)` options for sans

```
162 sf .code:n = {
163   \keys_set:nn { clm / sf } { #1 }
164 },
165 sf .value_required:n = true,
166 sf .usage:n = load,
```

`tt (opt.)` options for typewriter

```
167 tt .code:n = {
168   \keys_set:nn { clm / tt } { #1 }
169 },
170 tt .value_required:n = true,
171 tt .usage:n = load,
```

`qt (opt.)` note that this option does nothing unless `\qtfont` is defined appropriately

```
172 qt .bool_set:N = \l__clm_qt_bool,
173 qt .default:n = true,
174 qt .initial:n = false,
175 qt .usage:n = load,
```

`nomaths (opt.)` Make maths configuration optional.

`nomath (opt.)`

```
176 nomaths .bool_set_inverse:N = \l__clm_classicmaths_bool,
177 nomaths .default:n = true,
178 nomaths .initial:n = false,
179 nomaths .usage:n = load,
180 nomath .bool_set_inverse:N = \l__clm_classicmaths_bool,
181 nomath .default:n = true,
182 nomath .initial:n = false,
183 nomath .usage:n = load,
```

`maths (opt.)` Make maths configuration flexible to accommodate Unicode options.

`math (opt.)`

```
184 maths .choices:nn = {classic, lum, unicode-math, none} {
185   \int_case:nnF {\l_keys_choice_int} {
```

```

186     1 {
187         \bool_set_true:N \l__clm_classicmaths_bool
188         \bool_set_false:N \l__clm_lum_bool
189     }
190     2 {
191         \sys_if_engine_luatex:T {
192             \bool_set_true:N \l__clm_lum_bool
193             \bool_set_false:N \l__clm_classicmaths_bool
194             \RequirePackage{lum-lmodern}
195         }
196     }
197     3 {
198         \sys_if_engine_opentype:T {
199             \bool_set_false:N \l__clm_classicmaths_bool
200             \bool_set_false:N \l__clm_lum_bool
201             \RequirePackage{unicode-math}
202             \setmathfont{latinmodern-math.otf}
203         }
204     }
205     4 {
206         \bool_set_false:N \l__clm_classicmaths_bool
207         \bool_set_false:N \l__clm_lum_bool
208     }
209 }
210 },
211 math .meta:n = {maths = #1},
212 maths .usage:n = load,
213 math .usage:n = load,
214 maths .default:n = classic,
215 math .default:n = classic,
216 }

```

options override defaults

```

217 \IfFormatAtLeastTF { 2022-06-01 }
218 {
219     \ProcessKeyOptions [ clm ]
220 }{
221     \RequirePackage { l3keys2e }
222     \ProcessKeysOptions { clm }
223 }
224 \IfFormatAtLeastTF { 2020-10-01 }{
225 }{
226     \RequirePackage { xparse }
227     \providecommand \ExpandArgs [1]
228     { \cs_if_exist_use:c { exp_args:N #1 } }
229 }

```

There is not an easy path from postscript type1 to Unicode fonts. In particular, the `\text{symbol}` macros are simply not supported in many cases. In many fonts, the symbols aren't there anyway. This being Latin Modern, they are¹⁵. The following definitions do *not* give oldstyle numerals, but they do, at least, produce numerals. The mapping here *is* to oldstyle, but, if lining figures are active, they are mapped to lining figures by the font. For monowidth fonts, it does not work anyway.

```

230 \ExplSyntaxOff
231 \csname sys_if_engine_opentype:T\endcsname {%
232     \expandafter\chardef\csname TU\string\textcopyleft\endcsname"EB0F
233     \expandafter\chardef\csname TU\string\textleaf\endcsname"EB40
234     \expandafter\chardef\csname TU\string\textthreequartersemdash\endcsname"F6DE

```

¹⁵Except in the monowidth fonts, for some reason.

```

235 \expandafter\chardef\csname TU\string\textcentoldstyle\endcsname"F7A2
236 \expandafter\chardef\csname TU\string\textdollaroldstyle\endcsname"F724
237 \expandafter\chardef\csname TU\string\textzerooldstyle\endcsname"F730
238 \expandafter\chardef\csname TU\string\textoneoldstyle\endcsname"F731
239 \expandafter\chardef\csname TU\string\texttwooldstyle\endcsname"F732
240 \expandafter\chardef\csname TU\string\textthreeoldstyle\endcsname"F733
241 \expandafter\chardef\csname TU\string\textfouroldstyle\endcsname"F734
242 \expandafter\chardef\csname TU\string\textfiveoldstyle\endcsname"F735
243 \expandafter\chardef\csname TU\string\textsixoldstyle\endcsname"F736
244 \expandafter\chardef\csname TU\string\textsevenoldstyle\endcsname"F737
245 \expandafter\chardef\csname TU\string\texteightoldstyle\endcsname"F738
246 \expandafter\chardef\csname TU\string\textnineoldstyle\endcsname"F739
247 \expandafter\chardef\csname TU\string\textquotestraightbase\endcsname"EB56
248 \expandafter\chardef\csname TU\string\textquotestraightdblbase\endcsname"EB52
249 \expandafter\chardef\csname TU\string\textdblhyphen\endcsname"EB31
250 \expandafter\chardef\csname TU\string\textdblhyphenchar\endcsname"EB32
251 \expandafter\chardef\csname TU\string\texttwelveudash\endcsname"EB6B
252 \expandafter\def\csname TU\string\capitaltie\endcsname{\add@accent{"EB63}}%
253 \expandafter\def\csname TU\string\capitalnewtie\endcsname{\add@accent{"EB64}}%
254 \expandafter\def\csname TU\string\capitalhungarumlaut\endcsname{\add@accent{"EA38}}%
255 \expandafter\def\csname TU\string\capitalring\endcsname{\add@accent{"EA40}}%
256 \expandafter\def\csname TU\string\capitalcedilla\endcsname#1{%
257 \hmode@bgroup\oalign{\null #1\cr\hidewidth\char "B8\hidewidth}\egroup
258 }%

```

The Latin Modern fonts are unfortunately rather buggy, so we need to patch them. Arguably, this should be in the font definition files, but then it is harder to workaround if this is required at some point. Also, I'm not so sure whether engine-specific code should live in such files and this will work only for Lua_T_EX.

Thanks to Marcel Krüger for the following patch¹⁶.

```

259 \csname sys_if_engine luatex:TF\endcsname {%
260 \directlua{
261     local function assign_codepoints_to_names(name_cp_map)
262     return function(data)
263         local characters = data.characters
264         local descriptions = data.descriptions
265         local glyph_name_map = data.resources.unicodes
266         for name, cp in pairs(name_cp_map) do
267             local slot = glyph_name_map[name]
268             if slot then
269                 characters[cp] = table.merged(descriptions[slot])
270             else
271                 tex.error("Failed to find glyph with specified name")
272             end
273         end
274         return data
275     end
276 end
277
278 local remap_lmpatch = assign_codepoints_to_names{
279     gnaborretni = 0x2E18,
280     guarani = 0x20B2,
281     bigcircle = 0x25EF,
282     space_uni0330 = 0x02F7,
283     ['dblgrave.ts1'] = 0x02F5,
284 }
285 fonts.constructors.features.otf.register{
286     name = 'lmpatch',

```

¹⁶<https://chat.stackexchange.com/transcript/message/69163054#69163054>.

```

287     description = "Patch Latin Modern",
288     initializers = {
289         node = remap_lmpatch,
290         base = remap_lmpatch,
291     },
292 }
293 }%
294 \long\def\clm@patch{+lmpatch;}%
295 }{%
296 \let\clm@patch\@empty
297 }%
298 }
299 \ExplSyntaxOn

```

Turn slashed zero options into font features or not.

```

300 \sys_if_engine_opentype:T {
301   \clist_map_inline:nn {rm, sf, tt} {
302     \bool_if:cTF {l__clm_#1_zero_bool} {
303       \expandafter\def\csname clm@#1@zero\endcsname{+zero;}
304     } {
305       \expandafter\let\csname clm@rm@zero\endcsname\@empty
306     }
307   }
308 }

```

Translate user/default settings into bits of Berry names.

`\rmdefault` Make LM default for all families, implementing options for each

`\sfdefault`

`\ttdefault`

```

309 \tl_gset:Ne \rmdefault
310 {
311   clm \bool_if:NT \l__clm_rm_prop_bool { 2 }
312   \bool_if:NT \l__clm_rm_osf_bool { j }
313 }
314 \tl_gset:Ne \sfdefault
315 {
316   clm \bool_if:NT \l__clm_sf_prop_bool { 2 }
317   \bool_if:NT \l__clm_sf_osf_bool { j } s
318 }
319 \tl_gset:Ne \ttdefault
320 {
321   clm \bool_if:NT \l__clm_tt_prop_bool { 2 }
322   \bool_if:NT \l__clm_tt_osf_bool { j }
323   \bool_if:NTF \l__clm_tt_mono_bool { t } { v }
324 }

```

`\qtfont` Handle the qt option, failing gracefully if somebody has enabled the option without defining `\qtfont` appropriately. We do this in a hook at the start of the document so we can respond if the required macro is defined after `cfr-lm` is loaded. It would be nice if there was a more satisfactory approach, but I can't think of one a font package has any business implementing.

```

325 \hook_gput_code:nnn { begindocument } { . }
326 {
327   \bool_if:NT \l__clm_qt_bool
328   {
329     \cs_if_exist:NTF { \qtfont }
330     {
331       \qtfont{\qtstyle}
332     }{
333       \PackageWarning{cfr-lm}
334       {

```

```

335     Option ~ 'qt' ~ cannot ~ be ~ implemented ~ unless \MessageBreak
336     '\backslash qtfont' ~ is ~ defined ~ appropriately. ~ This ~ is ~ not \MessageBreak
337     done ~ automatically ~ to ~ maximise ~ compatibility ~ with ~ other \MessageBreak
338     classes ~ and ~ packages. ~ The ~ suggested ~ use ~ is ~ to ~ have ~ '\backslash
qtfont' \MessageBreak
339     redefine ~ a command ~ such ~ as ~ '\backslash quotefont' ~ which ~ is ~ initially
\MessageBreak
340     set ~ to ~ some ~ reasonable ~ default ~ and ~ used ~ in ~ the \MessageBreak
341     definition ~ of ~ the ~ quote ~ and/or ~ quotation ~ environments. \MessageBreak
342     A ~ font ~ package ~ has ~ no ~ business ~ meddling ~ in ~ such \MessageBreak
343     things ~ so ~ I'm ~ going ~ to ~ ignore ~ this ~ option
344   }
345 }
346 }
347 }
348 %

```

Maths setup is 'based on' (i.e. filched from) lmodern.sty

```

349 \bool_if:NTF \l__clm_classicmaths_bool
350 {
351   \SetSymbolFont{operators} {normal}{OT1}{lmr}{m}{n}
352   \SetSymbolFont{letters} {normal}{OML}{lmm}{m}{it}
353   \SetSymbolFont{symbols} {normal}{OMS}{lmsy}{m}{n}
354   \SetSymbolFont{largesymbols}{normal}{OMX}{lmex}{m}{n}
355   \SetSymbolFont{operators} {bold} {OT1}{lmr}{bx}{n}
356   \SetSymbolFont{letters} {bold} {OML}{lmm}{b}{it}
357   \SetSymbolFont{symbols} {bold} {OMS}{lmsy}{b}{n}
358   \SetSymbolFont{largesymbols}{bold} {OMX}{lmex}{m}{n}
359 %
360   \DeclareFontSubstitution{OT1}{lmr}{m}{n}
361   \DeclareFontSubstitution{OML}{lmm}{m}{it}
362   \DeclareFontSubstitution{OMS}{lmsy}{m}{n}
363   \DeclareFontSubstitution{OMX}{lmex}{m}{n}
364 %

\mathbf maths alphabets
\mathsf
\mathit
\mathtt
365   \SetMathAlphabet{\mathbf}{normal}{OT1}{lmr}{bx}{n}
366   \SetMathAlphabet{\mathsf}{normal}{OT1}{lms}{m}{n}
\mathtt
367   \SetMathAlphabet{\mathit}{normal}{OT1}{lmr}{m}{it}
368   \SetMathAlphabet{\mathtt}{normal}{OT1}{lmtt}{m}{n}
369   \SetMathAlphabet{\mathbf}{bold} {OT1}{lmr}{bx}{n}
370   \SetMathAlphabet{\mathsf}{bold} {OT1}{lms}{bx}{n}
371   \SetMathAlphabet{\mathit}{bold} {OT1}{lmr}{bx}{it}
372   \SetMathAlphabet{\mathtt}{bold} {OT1}{lmtt}{bx}{n}
373 %

\mathsterling
374   \def\mathsterling{\mathit{\mathchar"70BF}}
375 } {
376   \bool_if:NT \l__clm_lum_bool {
377     \SetMathAlphabet{\mathbf}{normal}{TU}{lmr}{bx}{n}
378     \SetMathAlphabet{\mathsf}{normal}{TU}{lms}{m}{n}
379     \SetMathAlphabet{\mathit}{normal}{TU}{lmr}{m}{it}
380     \SetMathAlphabet{\mathtt}{normal}{TU}{lmtt}{m}{n}
381     \SetMathAlphabet{\mathbf}{bold} {TU}{lmr}{bx}{n}
382     \SetMathAlphabet{\mathsf}{bold} {TU}{lms}{bx}{n}
383     \SetMathAlphabet{\mathit}{bold} {TU}{lmr}{bx}{it}
384     \SetMathAlphabet{\mathtt}{bold} {TU}{lmtt}{bx}{n}

```

```

385 }
386 }
387 \ExplSyntaxOff

\dotdigitenc
\textdde
388 \DeclareRobustCommand{\dotdigitenc}{%
389 \not@math@alphabet\dotdigitenc\relax
390 \fontencoding{U}\selectfont}
391 \DeclareTextFontCommand{\textdde}{\dotdigitenc}

```

\zeroslash The slashed zero¹⁷.

```
392 \newcommand*{\zeroslash}{\textdde{\char 250}}
```

Partly from microtype docs; partly from MinionPro package

We need to set up aliases for the font families created by cfr-lm so that microtype recognises them as similar to Computer Modern Roman.

T1/TU families in cfr-lm: clm clm2 clm2d clm2dj clm2j clm2jqs clm2js clm2jt clm2jv clm2qs clm2s clm2t clm2v clmd clmdj clmj clmjqs clmjs clmjt clmjv clmqv clms clmt clmv

See variants set in nfssex-cfr

Ref: <https://tex.stackexchange.com/a/75440>

```
393 \ExplSyntaxOn
```

Aliases for microtype so fonts get the same custom treatment they do with lm.

```

394 \cs_new_nopar:Nn \__clm_microtype_hook:
395 {
396 \clist_map_inline:nn
397 {
398 clm,clm2,clm2j,clmj,% roman
399 clms,clm2js,clm2s,clmjs,% sans
400 clmqv,clm2jqs,clm2qs,clmjqs%
401 % clm2d,clm2dj,clmd,clmdj,% leave unaliased?
402 % clm2jv,clmjv,clm2v,clmv,% leave unaliased?
403 % clmt,clm2t,clmjt,clm2jt% leave unaliased?
404 }{
405 \DeclareMicrotypeAlias { ##1 } { cmr }
406 }
407 }

```

If \Microtype@Hook exists, add our material to it; if not, create it.

```

408 \ifpackageloaded{microtype}
409 {
410 \__clm_microtype_hook:
411 }{
412 \cs_if_free:cTF { Microtype@Hook }
413 {% MinionPro has \global before this
414 \cs_new_eq:NN \Microtype@Hook \__clm_microtype_hook:
415 }{
416 \g@addto@macro\Microtype@Hook{\__clm_microtype_hook:}
417 }
418 }
419 \ExplSyntaxOff

420 %% end cfr-lm.sty

```

¹⁷It would be possible to take this from the Unicode font, where applicable, but this is not currently implemented for per-character switches.

Change History

cfr-lm *v2.0 (SVN Rev: 12139)* 28 / 31

functions). Scrap the component <code>tls</code> , since we're expanding the lot anyhow.	25	<code>nomath</code> : Follow <code>lmodern.sty</code> : add <code>nomaths/ nomath</code> option.	22
v1.8		v2.0	
<code>cfr-lm.sty</code> : <code>nfssect-cfr</code> enables TC subset declarations in font definition files for 'in between' formats.	18	<code>cfr-lm.sty</code> : Support for Unicode engines.	17
Determine TC encoding subset declarations dynamically and include in font definition files.	18	<code>math</code> : Add <code>maths/math</code> option to configure maths, if used.	22
v1.8a		<code>\mathtt</code> : Correct typewriter series in maths bold version.	26
<code>cfr-lm.sty</code> : Remove TS1 subset defs from fd files and put them back into the sty where they actually work.	18	<code>nomath</code> : Deprecate options <code>nomaths</code> and <code>nomath</code>	22
v1.9		<code>rm/zero: rm / zero-slashed</code> option for Unicode engines.	20
<code>cfr-lm.sty</code> : Maths configuration now optionally disabled.	26	<code>sf/zero: sf / zero-slashed</code> option for Unicode engines.	21
Re-put TS1 subset defs into fd files and skip them in the sty for current $\text{\LaTeX 2}\epsilon$	18	<code>tt / zero-slashed</code> option for Unicode engines.	22
Remove redundant provision of <code>\IfFormatAtLeastTF</code> as <code>nfssect-cfr</code> takes care of this, as noted above.	23	v2.0 (SVN Rev: 12139) General: Try switching to DTX/INS. Use <code>l3build</code>	1
		<code>cfr-lm.sty</code> : Remove alias cs and just define the hook.	27

Index

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

Symbols	<code>cfr-lm.sty</code> (pkg.)	<u>1</u>	50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66
<code>\@empty</code>	<code>\char</code>	257, 392	
<code>\@ifpackageloaded</code>	<code>\chardef</code>	232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251	<code>\DeclareFontSubstitution</code> 360, 361, 362, 363
<code>_clm_microtype_hook:</code>	<code>\clist_map_inline:nn</code>	301, 396	<code>\DeclareMicrotypeAlias</code> 405
A	<code>\clm@patch</code>	294, 296	<code>\DeclareRobustCommand</code> . 388
<code>\add@accent</code> 252, 253, 254, 255	<code>\crrcr</code>	257	<code>\DeclareTextFontCommand</code> 391
B	<code>\cs_if_exist:NTF</code>	329	<code>\def</code> 252, 253, 254, 255, 256, 294, 303, 374
<code>\backslash</code> 336, 338, 339	<code>\cs_if_exist_use:c</code>	228	<code>\directlua</code> 260
<code>\bool_if:cTF</code> 302	<code>\cs_if_free:cTF</code>	412	<code>\dotdigitenc</code> <u>388</u>
<code>\bool_if:NT</code> . 311, 312, 316, 317, 321, 322, 327, 376	<code>\cs_new_eq:NN</code>	414	E
<code>\bool_if:NTF</code> 323, 349	<code>\cs_new_nopar:Nn</code>	394	<code>\egroup</code> 257
<code>\bool_new:N</code>	<code>\csname</code> 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 259, 303, 305		<code>\endcsname</code> 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 259, 303, 305
<code>\bool_set_false:N</code> 85, 188, 193, 199, 200, 206, 207	C		<code>\expandafter</code> 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 303, 305
<code>\bool_set_true:N</code> 84, 187, 192	<code>\capitalcedilla</code>	256	
C	<code>\capitalhungarumlaut</code>	254	
<code>\capitalnewtie</code>	<code>\capitalring</code>	255	
<code>\capitaltie</code>	<code>\capitaltie</code>	252	
<code>\cdwidth</code>	<code>\cdwidth</code>	10	
D	<code>\DeclareEncodingSubset</code> 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,		

- `\ExpandArgs` 19,
 20, 21, 22, 23, 24, 25,
 26, 27, 28, 29, 30, 31,
 32, 33, 34, 35, 36, 37,
 38, 39, 40, 41, 42, 227
- F**
- `\fontencoding` 390
- G**
- `\g@addto@macro` 416
`\g__clm_enc_tl` 7, 9, 12
`\global` 413
- H**
- `\hidewidth` 257
`\hmode@bgroup` 257
`\hook_gput_code:nmn` ... 325
- I**
- `\IfFormatAtLeastTF`
 15, 18, 217, 224
`\int_case:nnF` 185
- K**
- `\keys_define:nn` 86
`\keys_set:nn` .. 158, 163, 168
- L**
- `\l__clm_classicmaths_bool`
 ... 83, 84, 176, 180,
 187, 193, 199, 206, 349
`\l__clm_lum_bool` ... 85,
 188, 192, 200, 207, 376
`\l__clm_qt_bool` 82, 172, 327
`\l__clm_rm_osf_bool`
 . 72, 88, 91, 93, 95, 312
`\l__clm_rm_prop_bool` ...
 73, 97, 100, 102, 104, 311
`\l__clm_rm_zero_bool` 74, 106
`\l__clm_sf_osf_bool`
 75, 108, 111, 113, 115, 317
`\l__clm_sf_prop_bool` 76,
 117, 120, 122, 124, 316
`\l__clm_sf_zero_bool` 77, 126
`\l__clm_tt_mono_bool` 80,
 146, 149, 151, 153, 323
`\l__clm_tt_osf_bool` . 78,
 128, 131, 133, 135, 322
`\l__clm_tt_prop_bool` 79,
 137, 140, 142, 144, 321
`\l__clm_tt_zero_bool` 81, 155
`\l_keys_choice_int` 185
`\let` 296, 305
`\long` 294
`\lstyle` 12
- M**
- `math` (opt.) 8, 184
`\mathbf` 365, 377, 381
`\mathchar` 374
`\mathit` .. 365, 374, 379, 383
- `maths` (opt.) 8, 184
`\mathsf` 365, 378, 382
`\mathsterling` 374
`\mathtt` 365, 380, 384
`\MessageBreak`
 335, 336, 337,
 338, 339, 340, 341, 342
`\Microtype@Hook` ... 414, 416
- N**
- `\newcommand` 392
`nomath` (opt.) 8, 176
`nomaths` (opt.) 8, 176
`\not@math@alphabet` 389
`\null` 257
- O**
- `\oalign` 257
 options:
`math` 8, 184
`maths` 8, 184
`nomath` 8, 176
`nomaths` 8, 176
`qt` 8, 172
`rm` 7, 157
`rm/lf` 7, 95
`rm/lining` 7, 93
`rm/oldstyle` 7, 88
`rm/osf` 7, 91
`rm/prop` 7, 100
`rm/proportional` .. 7, 97
`rm/tab` 7, 104
`rm/tabular` 7, 102
`rm/zero` 106
`rm/zero-slashed` 7
`sf` 7, 162
`sf/lf` 7, 115
`sf/lining` 7, 113
`sf/oldstyle` 7, 108
`sf/osf` 7, 111
`sf/prop` 7, 120
`sf/proportional` .. 7, 117
`sf/tab` 7, 124
`sf/tabular` 7, 122
`sf/zero` 126, 155
`sf/zero-slashed` 7
`tt` 7, 167
`tt/lf` 8, 135
`tt/lining` 8, 133
`tt/mono` 8, 149
`tt/monowidth` 8, 146
`tt/oldstyle` 8, 128
`tt/osf` 8, 131
`tt/prop` 8, 140
`tt/proportional` .. 8, 137
`tt/tab` 8, 144
`tt/tabular` 8, 142
`tt/var` 8, 153
`tt/variable` 8, 151
`tt/zero-slashed` 8
`\ostyle` 12
- P**
- `\PackageWarning` 333
`\plstyle` 13
`\postyle` 13
`\ProcessKeysOptions` ... 222
`\providecommand` 227
`\pstyle` 12
- Q**
- `qt` (opt.) 8, 172
`\qtfont` 325
`\qtstyle` 13, 331
- R**
- `\regwidth` 10
`\relax` 389
`rm` (opt.) 7, 157
`rm/lf` (opt.) 7, 95
`rm/lining` (opt.) 7, 93
`rm/oldstyle` (opt.) 7, 88
`rm/osf` (opt.) 7, 91
`rm/prop` (opt.) 7, 100
`rm/proportional` (opt.) 7, 97
`rm/tab` (opt.) 7, 104
`rm/tabular` (opt.) 7, 102
`rm/zero` (opt.) 106
`rm/zero-slashed` (opt.) ... 7
`\rmdefault` 309
- S**
- `\selectfont` 390
`\SetMathAlphabet`
 365, 366, 367,
 368, 369, 370, 371,
 372, 377, 378, 379,
 380, 381, 382, 383, 384
`\setmathfont` 202
`\SetSymbolFont`
 351, 352, 353,
 354, 355, 356, 357, 358
`sf` (opt.) 7, 162
`sf/lf` (opt.) 7, 115
`sf/lining` (opt.) 7, 113
`sf/oldstyle` (opt.) ... 7, 108
`sf/osf` (opt.) 7, 111
`sf/prop` (opt.) 7, 120
`sf/proportional` (opt.) 7, 117
`sf/tab` (opt.) 7, 124
`sf/tabular` (opt.) 7, 122
`sf/zero` (opt.) 126, 155
`sf/zero-slashed` (opt.) ... 7
`\sfdefault` 309
`\sishape` 11
`\string` 232, 233, 234, 235,
 236, 237, 238, 239,
 240, 241, 242, 243,
 244, 245, 246, 247,
 248, 249, 250, 251,
 252, 253, 254, 255, 256
`\sys_if_engine luatex:T` 191
`\sys_if_engine opentype:T`
 198, 300

<code>\sys_if_engine_opentype:TF</code>	<code>\textquotestraightdblbases</code>	<code>\tostyle</code>	13
8	248	<code>\tstyle</code>	12
T	<code>\textrw</code>	<code>tt</code> (opt.)	7, 167
<code>\textcd</code>	10	<code>tt/lf</code> (opt.)	8, 135
<code>\textcentoldstyle</code>	235	<code>tt/lining</code> (opt.)	8, 133
<code>\textcopyleft</code>	232	<code>tt/mono</code> (opt.)	8, 149
<code>\textdblhyphen</code>	249	<code>tt/monowidth</code> (opt.)	8, 146
<code>\textdblhyphenchar</code>	250	<code>tt/oldstyle</code> (opt.)	8, 128
<code>\textdde</code>	388, 392	<code>tt/osf</code> (opt.)	8, 131
<code>\textdollaroldstyle</code>	236	<code>tt/prop</code> (opt.)	8, 140
<code>\texteightoldstyle</code>	245	<code>tt/proportional</code> (opt.)	8, 137
<code>\textfiveoldstyle</code>	242	<code>tt/tab</code> (opt.)	8, 144
<code>\textfouroldstyle</code>	241	<code>tt/tabular</code> (opt.)	8, 142
<code>\textl</code>	12	<code>tt/var</code> (opt.)	8, 153
<code>\textleaf</code>	233	<code>tt/variable</code> (opt.)	8, 151
<code>\textnineoldstyle</code>	246	<code>tt/zero-slashed</code> (opt.)	8
<code>\texto</code>	12	<code>\ttdefault</code>	309
<code>\textoneoldstyle</code>	238	<code>\tvstyle</code>	13
<code>\textp</code>	12	U	
<code>\textperthousand</code>	70	<code>\uishape</code>	11
<code>\textpl</code>	13	<code>\UndeclareTextCommand</code>	70
<code>\textpo</code>	13	Z	
<code>\textqt</code>	13	<code>\zeroslash</code>	14, 392
<code>\textquotestraightbase</code>	247	<code>\tmstyle</code>	13
<code>\textquotestraightdblbases</code>	248		
<code>\textsevenoldstyle</code>	244		
<code>\textsi</code>	11		
<code>\textsixoldstyle</code>	243		
<code>\textt</code>	12		
<code>\textthreeoldstyle</code>	240		
<code>\textthreequartersemdash</code>	234		
<code>\textti</code>	14		
<code>\texttl</code>	13		
<code>\texttm</code>	13		
<code>\textto</code>	13		
<code>\texttv</code>	13		
<code>\texttwelveudash</code>	251		
<code>\texttwooldstyle</code>	239		
<code>\textui</code>	11		
<code>\textzerooldstyle</code>	237		
<code>\tistyle</code>	14		
<code>\tl_gset:Ne</code>	309, 314, 319		
<code>\tl_gset:Nn</code>	9, 12		
<code>\tl_new:N</code>	7		
<code>\tlstyle</code>	13		