The TFM User Manual

TEX Font Metrics, Version 1.1 patchlevel 1 "Carolingan Miniscules"

Didier Verna <didier@didierverna.net>

Table of Contents

Copying **1**

1 Introduction **3**

2 Installation **5**

3 Normal Use **7**
3.1 TFM Files 7
3.2 Font Information 7
3.3 Character Information 8
3.4 Ligatures and Kernings 9
3.4.1 Kernings 9
3.4.2 Ligatures 10
3.5 Scaling and Freezing 10
3.5.1 Scaling 10
3.5.2 Freezing 10

4 Abnormal Situations **13**
4.1 Conditions Overview 13
4.2 Erroneous Usage 13
4.3 TFM Compliance Problems 13

5 Other Considerations **17**
5.1 Configuration 17
5.2 Version Numbering 17

6 Conclusion **19**

Appendix A Supported Platforms **21**

Appendix B Indexes **23**
B.1 Concepts 23
B.2 Functions 24
B.3 Variables 26
B.4 Data Types 27

Appendix C Acknowledgments **29**

Copying

1 Introduction

TFM (for TEX Font Metrics) is the standard font description format used by TEX. The TFM library parses and decodes TFM files into an abstract data structure, providing easy access to the corresponding font information in Common Lisp.

This is the TFM User Manual. TFM also has a *Reference Manual*.

2 Installation

First of all, see Tfm's homepage for tarballs, Git repository and online documentation. Tfm is an ASDF 3 library. If you download a Tfm tarball, or clone the repository, you need to unpack somewhere in the ASDF source registry. Otherwise, Tfm is also available via Quicklisp. See Appendix A [Supported Platforms], page 21, for more information on portability and dependencies.

Tfm's main system is called '`net.didierverna.tfm`'. Depending on your installation, you may thus either `asdf:load-system`, or `ql:quickload` it in your Lisp image.

In addition to the Lisp library itself, the Tfm distribution offers documentation in the form of 2 different manuals: the User Manual (you are reading it) and the *Reference Manual*. If you want to compile the manuals by yourself, please follow the instructions below.

1. Edit `make/config.make` to your specific needs.
2. Type `make`. By default, the documentation is built in info, PDF and HTML formats. If you want other formats (DVI and PostScript are available), type `make all-formats`. You can also type individually `make dvi` and/or `make ps` in order to get the corresponding format.
3. Type `make install` to install the documentation. If you have compiled the documentation in DVI and PostScript format, those will be installed as well.

The reference manual's Texinfo source is included in the distribution (and in the repository), although it is generated automatically by Declt. Before compiling, it is possible to regenerate a local version of it with hyperlinks to your installation by typing `make localref`. For this to work, you need SBCL and Declt though. If you ever need to regenerate the regular version, you can also type `make generate`.

Type `make uninstall` to uninstall the library.

3 Normal Use

TFM itself resides in a package called `net.didierverna.tfm`. You can automatically nickname this package with the following function.

`nickname-package` [*NICKNAME*] [Function]
Add *NICKNAME* (`:tfm` by default) to the `:net.didierverna.tfm` package.

3.1 TFM Files

The main entry point to TFM is `load-font`, the function which allows you to load a `tfm` file in memory.

`load-font` *FILE* [Function]
Load *FILE* into a new font, and return it.

FILE must be a pathname designator. Currently, TFM doesn't support searching for TFM files in common or standard places (such as TEXLive installations), so you need to know where those files are located, and provide a specific pathname.

Some `tfm` files actually contain OFM or JFM data rather than just plain TFM. TFM is able to detect those extensions, but doesn't currently support them. If OFM or JFM contents is detected, `load-font` signals an `extended-tfm` warning, and returns `nil`.

`extended-tfm` [Warning]
Accessors:

- `file`: the argument to `load-font` (a pathname designator).
- `value`: currently either `"OFM"` or `"JFM"`.

Otherwise, if everything goes well (but see Chapter 4 [Abnormal Situations], page 13), the return value is an instance of some font class. In the context of this library, the term "font" denotes such an instance. There are currently three different font classes. The most general and frequent one is simply called `font`. TEX, however, recognizes two special font categories for which the corresponding classes are `math-symbols-font`, and `math-extension-font`. Both of these classes are subclasses of `font`.

3.2 Font Information

General font information decoded from the TFM data can be individually retrieved via a number of font accessors described below.

- `name`: computed by TFM as base name of the file the font was loaded from (for example `"cmr10"`).
- `file`: the argument to `load-font`.
- `checksum`: part of the TFM format, provided by Metafont.
- `frozen`: whether the font is frozen (see Section 3.5.2 [Freezing], page 10).
- `design-size`: in units of TEX point. For example, the design size of `cmr10` (the Computer Modern 10pt roman font) is 10. Apart from the *slant* (see below), all other dimensions are given in design size units, not TEX point units (unless the font has been frozen; See Section 3.5.2 [Freezing], page 10). See also Section 3.5.1 [Scaling], page 10, for ways to override the font's original design size.
- `original-design-size`: the fon't original design size.

TFM is able to decode Xerox PARC headers when provided (see TEX Font Metric Files, *David Fuchs, TUGboat Volume 2, n.1, pages 12–16* and PLtoPF[9,10]). The following information is extracted from them, and defaults to `nil` otherwise. Note that when a header of sufficient length is available, it is in fact impossible to tell for sure whether it is a Xerox PARC one or not. TFM simply assumes it is. As no other kind of header has been encountered yet (and the whole TEX Live distribution has been tested), it is a pretty safe assumption. Finally, if a header size is greater than that of a Xerox PARC one, TFM discards what's left of it.

- `encoding`: the font's character coding scheme if known (an informal BCPL string, that is, plain ASCII and no parentheses). The class TFM instantiates when loading a font depends on this string (see Section 3.1 [TFM Files], page 7), and in particular, on the specific (case insensitive) values `"TeX math symbols"` and `"TeX math extension"`.
- `family`: another BCPL string, if known (for example `"CMR"`).
- `7bits-safe` 0 or 1 if known. When 1, it means that no character of code lesser than 128 can lead to a character of code greater than 128 by ways of ligatures or extension recipes.
- `face-number` a number identifying the font, if known.
- `weight`, `slope`, `expansion`, and `face-code`: decoded from, and filled in when `face-number` is lesser than 18 (see also TFtoPL[10]). The weight is `:medium`, `:bold`, or `:light`. The slope is `:roman` or `:italic`. The expansion is `:regular`, `:condensed`, or `:extended`. Finally, the face code is the concatenation of the upcased first letters of these values. For example, `"MRR"` stands for "medium roman regular".

A font also contains a number of properties extracted from the so-called "parameters" section. They all default to 0.

- `slant`: a scalar ratio (not in design size units; that's the only exception).
- `interword-space`, `interword-stretch`, and `interword-shrink`: the font's normal interword "glue" as three separate values.
- `em` and `ex`: TEX's usual font-dependent units.
- `extra-space`: the font's additional space to put at the end of sentences.

TEX math symbols and extension fonts have respectively 15 and 6 additional parameters which are also accessible.

- `num1..3`, `denom1..2`, `sup1..3`, `sub1..2`, `subdrop`, `supdrop`, `delim1..2`, and `axis-height` are available in the `math-symbols-font` subclass.
- `default-rule-thickness` and `big-op-spacing1..5` are available in the `math-extension-font` subclass.

Finally, if the font has any parameters left, they are collected in an array of numeric values, accessible via the `parameters` function.

3.3 Character Information

A TFM font contains at most 256 characters, represented by numerical codes (called *character codes*). Some general information about the font's characters is available through the following accessors.

- `min-code` and `max-code`: respectively the smallest and greatest character codes for this font.
- `character-count`: the number of characters defined in this font. Note that some characters between `min-code` and `max-code` may not be defined, so the character count may be smaller than *max-code* - *min-code* + *1*. Also, the character count doesn't include a potential boundary character (see below), unless it exists for real in the font (it has non-zero metrics).

- `boundary-character`: the font's boundary character if defined, or `nil`. The boundary character is the only character the code of which is allowed to be outside the *[min-code,max-code]* range. If this character doesn't exist for real in the font, it has zero metrics and is not included in the character count.

The characters in a font are individually accessible by their code, thanks to the following function (when this function returns `nil`, it means that no character is defined for the code in question).

`get-character` *CODE FONT* [Function]
Return *FONT*'s *CODE* character, or `nil`.

This function actually returns an instance of a class called `character-metrics`, which contains character-specific data. In the context of this library, the term "character" denotes an instance of this class. The following accessors return the different character properties available.

- `code`: the character `code` which makes it accessible from the font instance (see the function `get-character`).
- `font`: the font the character belongs to.
- `width`, `height`, and `depth`: the character box dimensions, in design size units (but see Section 3.5.2 [Freezing], page 10).
- `italic-correction`: the character's italic correction, in design size units (but see Section 3.5.2 [Freezing], page 10). TEX uses this value for regular characters followed by the command `\/`, and also in math mode for superscript placement.
- `next-character`: the next character in a character list, or `nil`. If non-`nil`, this character is part of a chain of characters of ascending size, and not the last one (see TEX: the Program [544]). It is mutually exclusive with the existence of an extension recipe (see below), and also with the existence of a ligature or kerning program for this character (see Section 3.4 [Ligatures and Kernings], page 9).

Some characters are said to be "extensible" (meaning that they are constructed out of up to four components, themselves characters). To test whether a character is extensible, use the following function.

`extensiblep` *CHARACTER* [Function]
Return `T` if *CHARACTER* has an extension recipe.

When a character is extensible, the four components of the extension recipe may be retrieved with the `top-`, `middle-`, `bottom-`, and `repeated-character` accessors. These functions return a character, or `nil` if the corresponding component isn't used in the recipe. Note however that a repeated character always exists in an extension recipe (so only the top, middle, and bottom characters may be `nil`).

3.4 Ligatures and Kernings

The TFM format encodes so called "ligature / kerning programs", allowing font designers to specify particular spacing between characters, or character replacements. Ligatures and kernings depend on *pairs* of characters. TFM decodes this information and stores it in a more accessible fashion.

3.4.1 Kernings

In order to get kerning information for some pair of characters in a font, use the following function.

`kerning` *CHARACTER1 CHARACTER2* [Function]
Return kerning for *CHARACTER1* and *CHARACTER2*, or `nil`.
Both characters must belong to the same font.

The kerning, if it exists, is in design size units (but see Section 3.5.2 [Freezing], page 10).

3.4.2 Ligatures

In order to get ligature information for some pair of characters in a font, use the following function.

`ligature` *CHARACTER1 CHARACTER2* [Function]
Return ligature for *CHARACTER1* and *CHARACTER2*, or `nil`.
Both characters must belong to the same font.

Unless there's no such ligature, the value returned by this function is an instance of a class called `ligature`. In the context of this library, the term "ligature" denotes an instance of this class.

A ligature contains four bits of information accessible as described below.

- `composite`: the character to insert between the two original ones.
- `delete-before` and `delete-after`: whether to delete the characters before and after the composite.
- `pass-over`: the number of characters to skip in order to reach the next one, once the ligature has been made.

3.5 Scaling and Freezing

As we have seen, all dimensions in a TFM file are expressed in design size units. The idea is to make it simple for TEX to scale a font. Indeed, when a TEX user does `\font\foo=cmr10 at 12pt` or `\font\foo=cmr10 scaled 1200`, all TEX has to do is change the font's design size. The drawback, on the other hand, is that in order to actually use the various font dimensions, they need to be multiplied by the design size in question every time.

3.5.1 Scaling

TFM allows you to modify the design size of a font in two different ways. The first option is to override the original design size at load-time, by calling `load-font` with the `:design-size` key. For example, `(load-font "cmr10.tfm" :design-size 12)` will load the 10pt Computer Modern roman font at a design size of 12pt. The other option is to change the design size of a font later on, simply by using the `design-size` writer, for example like this: `(setf (design-size font) 12)`.

Note that you can always revert the font to the original design size, since that value is remembered (see Section 3.2 [Font Information], page 7).

In any case, the design size must in principle be expressed in TEX point units, and must be greater or equal to 1. One exception to this rule is if you are *not* working with TEX points. In such a situation, you could probably hack your own custom design size in, to express the font's various metrics in another unit of measurement.

3.5.2 Freezing

If you want to save some arithmetic operations, you may want to *freeze* the font. Freezing means that all dimensions normally expressed in design size units will be converted to TEX point units once and for all (by multiplying them by the design size in question). This includes character metrics, kerns, interword space *etc.* Everything. That way, you won't have to do the multiplication by yourself every time a dimension is accessed.

Again, there are two ways to freeze a font. The first option is to do it at load-time, by calling `load-font` with the `:freeze` key (a Boolean). For instance, after calling `(load-font "cmr10.tfm" :design-size 12 :freeze t)`, `(em font)` will return 12.000034 instead of 1.0000029.

The other way to freeze a font is to do it later on, by using the `freeze` function.

`freeze` *FONT* [Function]
Freeze *FONT*.

Freezing a font means that all dimensions normally expressed in design size units are multiplied by it, so as to lead values in TEX point units. If *FONT* is already frozen, this function does nothing and returns `nil`. Otherwise, it returns `t`.

You can unfreeze a font at any time with the following function.

`unfreeze` *FONT* [Function]
Unfreeze *FONT*.

Unfreezing means performing the inverse of what `freeze` does. If *FONT* is not frozen, this function does nothing and returns `nil`. Otherwise, it returns `t`.

Finally, note that if you change the design size of a frozen font, the font remains frozen and all concerned dimensions are updated to reflect the change.

4 Abnormal Situations

We already mentioned the `extended-tfm` warning in Section 3.1 [TFM Files], page 7. In fact, Tfm comes with a rich ontology of conditions (and restarts) that give you full control on the behavior to adopt when facing abnormal situations.

4.1 Conditions Overview

All Tfm conditions are grounded in a root hierarchy that could be used, for instance if you want to catch more or less everything without too much detail.

The root of all Tfm conditions is simply called `tfm`. Warnings and errors are all sub-conditions of `tfm-warning` and `tfm-error` respectively. Then, there are two main categories of conditions. Conditions related to a misuse of the library's API are classified under the `tfm-usage` hierarchy, while those related to problems with the TFM format are found under the `tfm-compliance` one.

Finally, the warning/error and usage/compliance families are mixed together (via multiple inheritance) to produce the four last "abstract" conditions from which all other derive (except for the already mentioned `extended-tfm` warning): `tfm-usage-warning`, and `tfm-usage-error`, `tfm-compliance-warning`, and `tfm-compliance-error`.

4.2 Erroneous Usage

Because the Tfm's public API is so small, there are only two usage conditions, so we'll start with those ones. Neither of these conditions is restartable at the level of Tfm.

An attempt at accessing the top, middle, bottom, or repeated character of a non extensible character will signal a `not-extensible` usage error. Remember that you can check a character's extensibility with the `extensiblep` predicate.

`not-extensible` [Usage Error]

Accessor:

- `value`: the character being accessed.

An attempt at retrieving a ligature or kerning for two characters from different fonts will signal a `different-fonts` usage error.

`different-fonts` [Usage Error]

Accessors:

- `character1` and `character2`: the two characters involved.

Finally, an attempt at changing the design size of a font to something that is not a real greater or equal to 1 will just signal a type error.

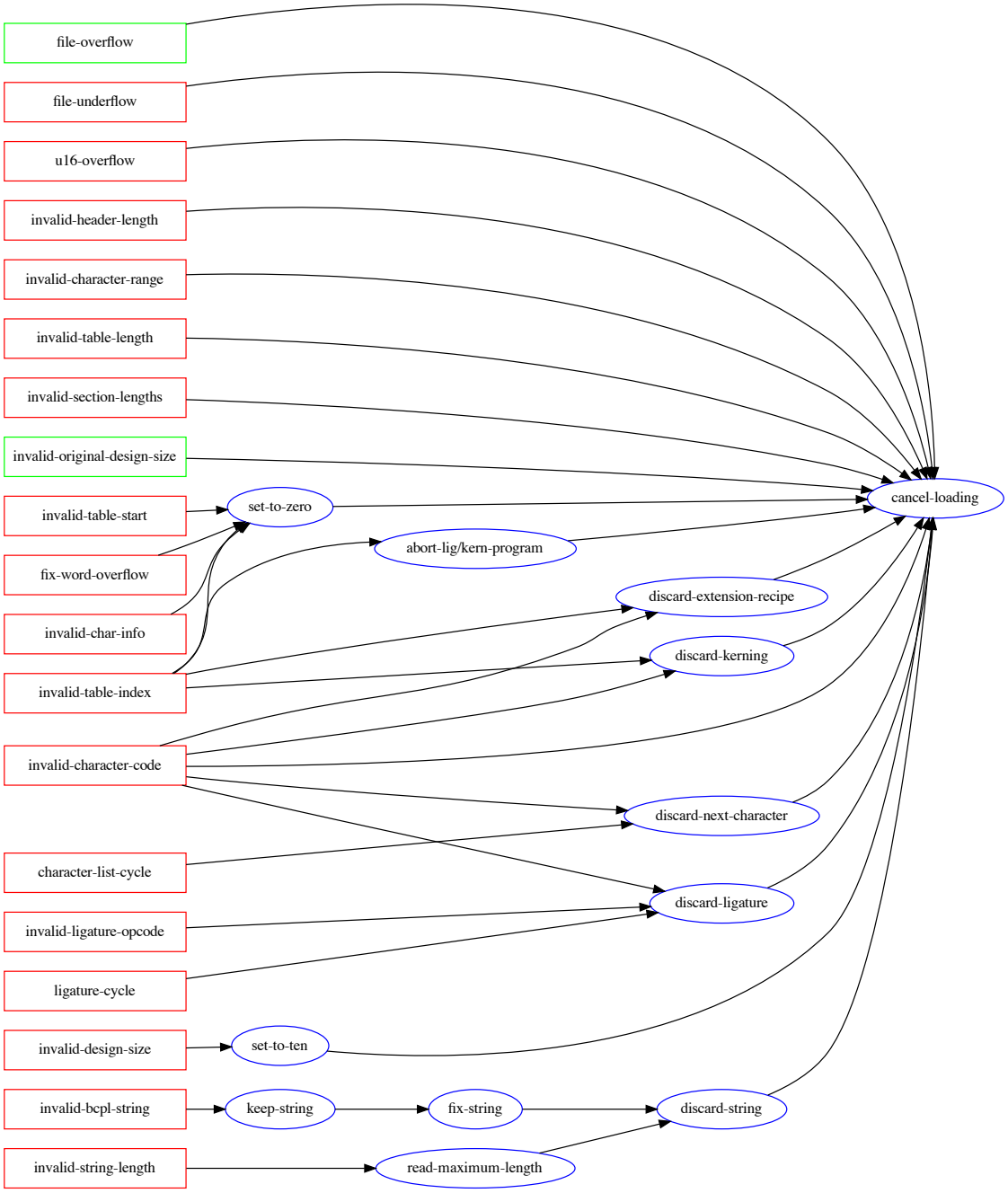
4.3 TFM Compliance Problems

Tfm is very thorough in the checks it performs while reading TFM data, to the point that it could probably be used as a validation tool. Every compliance problem detected will be signalled with a warning or an error, depending on the seriousness of the problem (in fact, only two compliance conditions are warnings currently; all others being errors). At the same time, Tfm also tries to offer a way out of problematic situations (in the form of restarts, currently non-interactive), when it sounds reasonable. This means that it is almost always possible to load a font, however broken it may be.

When loading a file, a restart named `cancel-loading` is always active. If invoked, `load-font` will just abort and return `nil`. Otherwise, the available restarts depend on the dynamic context

and the particular condition being signalled. The graph on the next page depicts the conditions / restarts ontology related to compliance problems. Green boxes denote warnings, red ones denote errors, and blue ellipses denote restarts. By following a branch from left to right in this graph, you get an idea of which restarts are available when a specific condition is signalled. They are ordered by decreasing proximity to the signalling code (or, to put it differently, by increasing level of abstraction). The conditions in this graph are described precisely in Section "Exported conditions" in *The TFM Reference Manual*. The restarts, on the other hand, are not, but their names should be self-explanatory. Note that a branch in this graph does not necessarily correspond to a specific code path in the library. Different code paths may have different restarts with the same name when it makes sense, but eponymous restarts are "merged" into a single node in the graph.

One particular case deserves some additional bits of information though. You will notice the existence of both an `invalid-design-size` *error*, and an `invalid-original-design-size` *warning*. Remember that these do not concern user-provided design sizes, which may only trigger a type error (see Section 4.2 [Erroneous Usage], page 13). In the default case, the `invalid-design-size` error is signalled when a TFM file is not compliant. If, however, you have call `load-font` with an alternative `:design-size`, you are effectively overriding the original value, which will not be used. In that case, only a warning is signalled: the `invalid-original-design-size` warning. Finally, please consider that leaving that warning alone means that an invalid original design size will be kept around in the font, so if you plan on using it later on, you're looking for trouble.


file-overflow
file-underflow
u16-overflow
invalid-header-length
invalid-character-range
invalid-table-length
invalid-section-lengths
invalid-original-design-size
invalid-table-start
fix-word-overflow
invalid-char-info
invalid-table-index
invalid-character-code
character-list-cycle
invalid-ligature-opcode
ligature-cycle
invalid-design-size
invalid-bcpl-string
invalid-string-length
set-to-zero
abort-lig/kern-program
discard-extension-recipe
discard-kerning
discard-next-character
discard-ligature
set-to-ten
keep-string
fix-string
discard-string
read-maximum-length
cancel-loading

5 Other Considerations

This section contains marginal or meta-information, orthogonal to the library's main purpose.

5.1 Configuration

Some aspects of TFM's behavior can be configured *before* the library system is actually loaded. TFM stores its user-level configuration (along with some other setup parameters) in another ASDF system called '`net.didierverna.tfm.setup`' (and the eponymous package). In order to configure the library (I repeat, prior to loading it), you will typically do something like this:

```
(require "asdf")
(asdf:load-system :net.didierverna.tfm.setup)
(net.didierverna.tfm.setup:configure <option> <value>)
```

`configure` *KEY VALUE* [Function]
Set *KEY* to *VALUE* in the current TFM configuration.

Out of curiosity, you can also inquire the current configuration for specific options with the following function.

`configuration` *KEY* [Function]
Return *KEY*'s value in the current TFM configuration.

Currently, the following options are provided.

`:swank-eval-in-emacs`
: This option is only useful if you use Slime, and mostly if you plan on hacking TFM itself. The library provides indentation information for some of its functions directly embedded in the code. This information can be automatically transmitted to Emacs when the ASDF system is loaded if you set this option to `t`. However, note that for this to work, the Slime variable `slime-enable-evaluate-in-emacs` must also be set to `t` in your Emacs session. If you're interested to know how this process works, I have described it a Blog entry.

5.2 Version Numbering

As TFM evolves over time, you might one day feel the need for conditionalizing your code on the version of the library.

The first thing you can do to access the current version number of TFM is use the `version` function.

`version` **&optional** (*TYPE* **:number**) [Function]
Return the current version number of TFM. *TYPE* can be one of `:number`, `:short` or `:long`. For `:number`, the returned value is a fixnum. Otherwise, it is a string.

A TFM version is characterized by 4 elements as described below.

- A major version number stored in the parameter `*release-major-level*`.
- A minor version number, stored in the parameter `*release-minor-level*`.
- A release status stored in the parameter `*release-status*`. The status of a release can be `:alpha`, `:beta`, `:rc` (standing for "release candidate") or `:patchlevel`. These are in effect 4 levels of expected stability.
- A status-specific version number stored in the parameter `*release-status-level*`. Status levels start at 1 (alpha 1, beta 1 and release candidate 1) except for stable versions, in which case patch levels start at 0 (*e.g.* 2.4.0).

In addition to that, each version of TFM (in the sense *major.minor*, regardless of the status) has a name, stored in the parameter `*release-name*`. The general naming theme for TFM is "Uncial Fonts", from the LaTeX Font Catalogue.

Here is how the `version` function computes its value.

- A version `:number` is computed as *major * 10000* + *minor * 100* + *patchlevel*, effectively leaving two digits for each level. Note that alpha, beta and release candidate status are ignored in version numbers (this is as if the corresponding status level was considered to be always 0). Only stable releases have their level taken into account.
- A `:short` version will appear like this for unstable releases: 1.3a4, 2.5b8 or 4.2rc1. Remember that alpha, beta or release candidate levels start at 1. Patchlevels for stable releases start at 0 but 0 is ignored in the output. So for instance, version 4.3.2 will appear as-is, while version 1.3.0 will appear as just 1.3.
- A `:long` version is expanded from the short one, and includes the release name. For instance, 1.3 alpha 4 "Artificial Uncial", 2.5 beta 8 "Rotunda", 4.2 release candidate 1 "Romantic Rustic" or 4.3.2 "Square Capitals". As for the short version, a patchlevel of 0 is ignored in the output: 1.3 "Artificial Uncial".

Incidentally, but you will probably never need to use it, TFM also exports a variable named `*copyright-years*`, which, as its name suggests, is a string denoting the copyright years for the whole project.

6 Conclusion

So that's it I guess. You know all about TFM now. Thanks for reading. Apart from that, I don't really have a conclusion.

Appendix A Supported Platforms

Tfm is an ASDF 3 library. It requires `editor-hints.named-readtables`.

Appendix B Indexes

B.1 Concepts

:

`:swank-eval-in-emacs` 17

C

Character 9
Character code 8
Character list 9
Character, extensible 9
Configuration 17
Configuration option, `:swank-eval-in-emacs` 17

D

Design size unit 7

E

Extensible character 9

F

Font 7
Font freezing 10
Font scaling 10
Format, JFM 7
Format, OFM 7
Format, TFM 7
Freezing 10

J

JFM 7

K

Kerning 9

L

Ligature 10

O

OFM 7

P

Package, nicknames 7

S

Scaling 10

T

TEX point unit 7
TFM 7

U

Unit, design size 7
Unit, TEX point 7

B.2 Functions

7

7bits-safe (font accessor) 8

A

axis-height (math-symbols-font accessor) 8

B

big-op-spacing1 (math-extension-font accessor) 8
big-op-spacing2 (math-extension-font accessor) 8
big-op-spacing3 (math-extension-font accessor) 8
big-op-spacing4 (math-extension-font accessor) 8
big-op-spacing5 (math-extension-font accessor) 8
bottom-character (character-metrics accessor) 9, 13
boundary-character (font accessor) 9

C

character-count (font accessor) 8
character-metrics, accessor, bottom-character 9, 13
character-metrics, accessor, code 9
character-metrics, accessor, depth 9
character-metrics, accessor, font 9
character-metrics, accessor, height 9
character-metrics, accessor, italic-correction 9
character-metrics, accessor, middle-character 9, 13
character-metrics, accessor, next-character 9
character-metrics, accessor, repeated-character 9, 13
character-metrics, accessor, top-character .. 9, 13
character-metrics, accessor, width 9
character1 (different-fonts accessor) 13
character2 (different-fonts accessor) 13
checksum (font accessor) 7
code (character-metrics accessor) 9
composite (ligature accessor) 10
configuration 17
configure 17

D

default-rule-thickness (math-extension-font accessor) 8
delete-after (ligature accessor) 10
delete-before (ligature accessor) 10
delim1 (math-symbols-font accessor) 8
delim2 (math-symbols-font accessor) 8
denom1 (math-symbols-font accessor) 8
denom2 (math-symbols-font accessor) 8
depth (character-metrics accessor) 9
design-size (font accessor) 7, 10
different-fonts, accessor, character1 13
different-fonts, accessor, character2 13

E

em (font accessor) 8
encoding (font accessor) 8
ex (font accessor) 8
expansion (font accessor) 8
extended-tfm, accessor, file 7
extended-tfm, accessor, value 7
extensiblep 9, 13
extra-space (font accessor) 8

F

face-code (font accessor) 8
face-number (font accessor) 8
family (font accessor) 8
file (extended-tfm accessor) 7
file (font accessor) 7
font (character-metrics accessor) 9
font, accessor, 7bits-safe 8
font, accessor, boundary-character 9
font, accessor, character-count 8
font, accessor, checksum 7
font, accessor, design-size 7, 10
font, accessor, em 8
font, accessor, encoding 8
font, accessor, ex 8
font, accessor, expansion 8
font, accessor, extra-space 8
font, accessor, face-code 8
font, accessor, face-number 8
font, accessor, family 8
font, accessor, file 7
font, accessor, frozen 7
font, accessor, interword-shrink 8
font, accessor, interword-space 8
font, accessor, interword-stretch 8
font, accessor, max-code 8
font, accessor, min-code 8
font, accessor, name 7
font, accessor, original-design-size 7, 10
font, accessor, parameters 8
font, accessor, slant 8
font, accessor, slope 8
font, accessor, weight 8
freeze 11
frozen (font accessor) 7

G

get-character 9

H

height (character-metrics accessor) 9

I

interword-shrink (font accessor) 8
interword-space (font accessor) 8
interword-stretch (font accessor) 8
italic-correction
(character-metrics accessor) 9

K

kerning 10, 13

L

ligature 10, 13
ligature, accessor, composite 10
ligature, accessor, delete-after 10
ligature, accessor, delete-before 10
ligature, accessor, pass-over 10
load-font 7, 13
load-font, key, design-size 10, 14
load-font, key, freeze 10

M

math-extension-font, accessor,
big-op-spacing1 8
math-extension-font, accessor,
big-op-spacing2 8
math-extension-font, accessor,
big-op-spacing3 8
math-extension-font, accessor,
big-op-spacing4 8
math-extension-font, accessor,
big-op-spacing5 8
math-extension-font, accessor,
default-rule-thickness 8
math-symbols-font, accessor, axis-height 8
math-symbols-font, accessor, delim1 8
math-symbols-font, accessor, delim2 8
math-symbols-font, accessor, denom1 8
math-symbols-font, accessor, denom2 8
math-symbols-font, accessor, num1 8
math-symbols-font, accessor, num2 8
math-symbols-font, accessor, num3 8
math-symbols-font, accessor, sub1 8
math-symbols-font, accessor, sub2 8
math-symbols-font, accessor, subdrop 8
math-symbols-font, accessor, sup1 8
math-symbols-font, accessor, sup2 8
math-symbols-font, accessor, sup3 8
math-symbols-font, accessor, supdrop 8
max-code (font accessor) 8
middle-character
(character-metrics accessor) 9, 13
min-code (font accessor) 8

N

name (font accessor) 7
next-character (character-metrics accessor) 9
nickname-package 7
not-extensible, accessor, value 13
num1 (math-symbols-font accessor) 8
num2 (math-symbols-font accessor) 8
num3 (math-symbols-font accessor) 8

O

original-design-size (font accessor) 7, 10

P

parameters (font accessor) 8
pass-over (ligature accessor) 10

R

repeated-character
(character-metrics accessor) 9, 13

S

slant (font accessor) 8
slope (font accessor) 8
sub1 (math-symbols-font accessor) 8
sub2 (math-symbols-font accessor) 8
subdrop (math-symbols-font accessor) 8
sup1 (math-symbols-font accessor) 8
sup2 (math-symbols-font accessor) 8
sup3 (math-symbols-font accessor) 8
supdrop (math-symbols-font accessor) 8

T

top-character (character-metrics accessor) .. 9, 13

U

unfreeze 11

V

value (extended-tfm accessor) 7
value (not-extensible accessor) 13
version 17

W

weight (font accessor) 8
width (character-metrics accessor) 9

B.3 Variables

*

`*copyright-years*` 18
`*release-major-level*` (parameter) 17
`*release-minor-level*` (parameter) 17
`*release-name*` (parameter) 18
`*release-status*` (parameter) 17
`*release-status-level*` (parameter) 17

N

`net.didierverna.declt.configuration` 17

P

Parameter, `*release-major-level*` 17
Parameter, `*release-minor-level*` 17
Parameter, `*release-name*` 18
Parameter, `*release-status*` 17
Parameter, `*release-status-level*` 17

S

`slime-enable-evaluate-in-emacs` 17

B.4 Data Types

C

`cancel-loading` (restart) 13
`character-metrics` (class) 9
Class, `character-metrics` 9
Class, `font` 7
Class, `ligature` 10
Class, `math-extension-font` 7, 8
Class, `math-symbols-font` 7, 8
Condition, `tfm` 13
Condition, `tfm-compliance` 13
Condition, `tfm-compliance-error` 13
Condition, `tfm-compliance-warning` 13
Condition, `tfm-error` 13
Condition, `tfm-usage` 13
Condition, `tfm-usage-error` 13
Condition, `tfm-usage-warning` 13
Condition, `tfm-warning` 13

D

`different-fonts` 13

E

`editor-hints.named-readtables` (system) 21
Error, `different-fonts` 13
Error, `not-extensible` 13
Error, usage, `different-fonts` 13
Error, usage, `not-extensible` 13
`extended-tfm` 7, 13

F

`font` (class) 7

L

`ligature` (class) 10

M

`math-extension-font` (class) 7, 8
`math-symbols-font` (class) 7, 8

N

`net.didierverna.tfm` (package) 7
`net.didierverna.tfm` (system) 5
`net.didierverna.tfm.setup` (package) 17
`net.didierverna.tfm.setup` (system) 17
`not-extensible` 13

P

Package, `net.didierverna.tfm` 7
Package, `net.didierverna.tfm.setup` 17

R

Restart, `cancel-loading` 13

S

System, `editor-hints.named-readtables` 21
System, `net.didierverna.tfm` 5
System, `net.didierverna.tfm.setup` 17

T

`tfm` (condition) 13
`tfm-compliance` (condition) 13
`tfm-compliance-error` (condition) 13
`tfm-compliance-warning` (condition) 13
`tfm-error` (condition) 13
`tfm-usage` (condition) 13
`tfm-usage-error` (condition) 13
`tfm-usage-warning` (condition) 13
`tfm-warning` (condition) 13

U

Usage error, `different-fonts` 13
Usage error, `not-extensible` 13

W

Warning, `extended-tfm` 7, 13

Appendix C Acknowledgments

The following people have contributed to TFM in the form of bug reports or fixes, suggestions, expertise, or whatever else. Thank you!

Doug McKenna
Hironobu Yamashita
Karl Berry