

The **texnegar** package  
Kashida justification in LuaTeX and XeTeX  
Source code documentation

Hossein Movahhedian\*

Released 2021-01-31 v0.1d

---

**Negar:** *Negar, in Persian, is the present stem of negaashtan meaning to design; to paint; to write; and as a noun it means “sweetheart, idol, beloved, figuratively refering to a beautiful woman, pattern, painting, and artistic design”*

---

\*E-mail: [dma8hm1334@gmail.com](mailto:dma8hm1334@gmail.com)

## Contents

<b>1</b>	<b>TEXNegar Implementation</b>	<b>3</b>
1.1	File: texnegar.sty . . . . .	3
1.2	File: texnegar-luatex.sty . . . . .	3
1.3	File: texnegar-xetex.sty . . . . .	4
1.4	File: texnegar-ini.tex . . . . .	4
1.5	File: texnegar-common-kashida.tex . . . . .	12
1.6	File: texnegar-xetex-kashida.tex . . . . .	14
1.7	File: texnegar-char-table.lua . . . . .	17
1.8	File: texnegar.lua . . . . .	22
1.9	File: texnegar-ini.lua . . . . .	25
1.10	File: texnegar-luatex-kashida.lua . . . . .	26
<b>2</b>	<b>Acknowledgments</b>	<b>36</b>
<b>3</b>	<b>Change History</b>	<b>36</b>
	[2020-08-29 v0.1a] . . . . .	36
	[2020-08-30 v0.1b] . . . . .	37
	[2021-01-27 v0.1c] . . . . .	37
	<b>Index</b>	<b>38</b>

# 1 T<sub>E</sub>XNegar Implementation

## 1.1 File: texnegar.sty

```
1 <*texnegar-sty>
2 \RequirePackage{xparse}
3 \RequirePackage{l3keys2e}
4 \RequirePackage{graphicx}[2019-11-30]
5 \RequirePackage{array}[2019-10-01]
6 \RequirePackage[dvipsnames,svgnames,x11names]{xcolor}[2016/05/11]
7 \RequirePackage{fontspec}[2020/02/21]
8 \RequirePackage{newverbs}[2010/09/02]
9 \RequirePackage{environ}[2014/05/04]
10
11 \ProvidesExplPackage {texnegar} {2021-01-31} {0.1d} { Full implementation of kashida feature
12
13 \sys_if_engine_luatex:T
14 {
15     \RequirePackageWithOptions{texnegar-luatex}
16     \endinput
17 }
18 \sys_if_engine_xetex:T
19 {
20     \RequirePackageWithOptions{texnegar-xetex}
21     \endinput
22 }
23 \msg_new:nnn {texnegar} {cannot-use-pdftex}
24 {
25     The~ texnegar~ package~ requires~ either~ XeTeX~ or~ LuaTeX.\\
26     You~ must~ change~ your~ typesetting~ engine~ to,~ e.g.,~
27     "xelatex"~ or~ "lualatex" instead~ of~ "latex"~ or~ "pdflatex".
28 }
29 \msg_fatal:nn {texnegar} {cannot-use-pdftex}
30
31 \endinput
32 </texnegar-sty>
```

## 1.2 File: texnegar-luatex.sty

```
33 <*texnegar-luatex-sty>
34 \ProvidesExplPackage {texnegar-luatex} {2021-01-31} {0.1d} { Full implementation of kashida
35
36 \tex_input:D { texnegar-ini.tex }
37
38 \bool_if:NT \l_texnegar_kashida_fix_bool
39 {
40     \if_int_compare:w \luatexversion < \c_texnegar_luatexversionmajormin_int\c_texnegar_luat
41         \msg_error:nnxxx { texnegar } { luatex-version-is-too-old } { !!!! } { \c_texnegar_l
42     \fi:
43
44     \hbox_set:Nn \l_texnegar_k_box { \resizebox{5000sp}{\height}{-} }
45
46     \hbox_set:Nn \l_texnegar_ksh_box { \char\lua_now:n { tex.sprint(0, font.getfont(font.cur
47
48     \directlua{dofile(kpse.find_file("texnegar.lua"))}
```

```

49   }
50
51   \bool_if:NT \l_texnegar_kashida_fix_bool
52   {
53     \tex_input:D { texnegar-common-kashida.tex }
54
55     \AtBeginDocument
56     {
57       \KashidaOn
58     }
59   }
60
61   \endinput
62   </texnegar-luatex-sty>

```

### 1.3 File: texnegar-xetex.sty

```

63   <*texnegar-xetex-sty>
64   \RequirePackage{zref-savepos}[2020-03-03]
65   \ProvidesExplPackage {texnegar-xetex} {2021-01-31} {0.1d} { Full implementation of kashida f
66
67   \tex_input:D { texnegar-ini.tex }
68
69   \bool_if:NT \l_texnegar_kashida_fix_bool
70   {
71     \tex_input:D { texnegar-xetex-kashida.tex }
72   }
73
74   \endinput
75   </texnegar-xetex-sty>

```

### 1.4 File: texnegar-ini.tex

```

76   <*texnegar-ini-tex>
77   \ProvidesExplFile {texnegar-ini.tex} {2021-01-31} {0.1d} { Full implementation of kashida fe
78
79   \def\TeXNegar{\TeX Negar}
80
81   \box_new:N \l_texnegar_k_box
82   \box_new:N \l_texnegar_ksh_box
83
84   \tl_const:Nn \c_texnegar_luatexversionmajormin_int {1}
85   \tl_const:Nn \c_texnegar_luatexversionminormin_int {12}
86
87   \int_const:Nn \c_texnegar_ksh_int {"0640} % kashida
88   \int_const:Nn \c_texnegar_lrm_int {"200E} % left-right-mark
89   \int_const:Nn \c_texnegar_zwj_int {"200D} % zero-width joiner
90
91   \int_const:Nn \c_texnegar_two_int {2}
92   \int_const:Nn \c_texnegar_four_int {4}
93
94   \tl_const:Nn \c_texnegar_skip_a_tl { 0 em plus 0.5 em }
95   \tl_const:Nn \c_texnegar_skip_b_tl { 0.14 em plus 5.5 em }
96
97   \int_new:N \l_texnegar_counter_int
98

```

```

99 \int_new:N \l_texnegar_kashida_slot_int
100
101 \int_new:N \l_texnegar_line_break_penalty_int
102
103 \int_new:N \l_texnegar_min_penalty_int
104 \int_new:N \l_texnegar_low_penalty_int
105 \int_new:N \l_texnegar_med_penalty_int
106 \int_new:N \l_texnegar_high_penalty_int
107 \int_new:N \l_texnegar_max_penalty_int
108
109 \int_new:N \l_fontnumber_int
110
111 \tl_new:N \l_texnegar_line_break_tl
112
113 \tl_new:N \l_texnegar_main_font_full_tl
114 \tl_new:N \l_texnegar_main_font_name_tl
115
116 \tl_new:N \l_texnegar_font_full_tl
117 \tl_new:N \l_texnegar_font_name_tl
118
119 \tl_new:N \l_texnegar_skip_default_tl
120
121 \tl_new:N \l_texnegar_active_ligs_tl
122
123 \tl_new:N \l_texnegar_gap_filler_tl
124
125 \tl_new:N \l_texnegar_use_color_tl
126 \tl_new:N \l_texnegar_color_tl
127 \tl_new:N \l_texnegar_color_rgb_tl
128
129 \dim_new:N \l_texnegar_diff_pos_dim
130
131 \bool_set_false:N \l_texnegar_minimal_bool
132 \tl_set:Nn \l_texnegar_minimal_off_tl { Off }
133 \tl_set:Nn \l_texnegar_minimal_on_tl { On }
134
135 \bool_set_false:N \l_texnegar_kashida_fix_bool
136
137 \bool_set_false:N \l_texnegar_kashida_fontfamily_bool
138 \tl_new:N \l_texnegar_kashida_fontfamily_tl
139 \tl_set:Nn \l_texnegar_kashida_fontfamily_tl { N/A }
140
141 \bool_set_false:N \l_texnegar_kashida_glyph_bool
142 \bool_set_false:N \l_texnegar_kashida_leaders_glyph_bool
143 \bool_set_false:N \l_texnegar_kashida_leaders_hrule_bool
144
145 \bool_set_false:N \l_texnegar_ligature_bool
146 \bool_set_false:N \l_texnegar_linebreakpenalty_bool
147 \bool_set_false:N \l_texnegar_hboxrecursion_bool
148 \bool_set_false:N \l_texnegar_vboxrecursion_bool
149 \bool_set_false:N \l_texnegar_color_bool
150
151 \int_set:Nn \l_texnegar_min_penalty_int { 0 }
152 \int_set:Nn \l_texnegar_low_penalty_int { 8 }

```

```

153 \int_set:Nn \l_texnegar_med_penalty_int { 15 }
154 \int_set:Nn \l_texnegar_high_penalty_int { 25 }
155 \int_set:Nn \l_texnegar_max_penalty_int { 10000 }
156
157 \tl_set:Nn \l_texnegar_stretch_glyph_tl { glyph }
158 \tl_set:Nn \l_texnegar_stretch_leaders_glyph_tl { leaders+glyph }
159 \tl_set:Nn \l_texnegar_stretch_leaders_hruler_tl { leaders+hruler }
160 \tl_set:Nn \l_texnegar_stretch_off_tl { Off }
161 \tl_set:Nn \l_texnegar_stretch_on_tl { On }
162
163 \tl_set:Nn \l_texnegar_hboxrecursion_off_tl { Off }
164 \tl_set:Nn \l_texnegar_hboxrecursion_on_tl { On }
165
166 \tl_set:Nn \l_texnegar_vboxrecursion_off_tl { Off }
167 \tl_set:Nn \l_texnegar_vboxrecursion_on_tl { On }
168
169 \tl_set:Nn \l_texnegar_fnt_kayhan_tl { kayhan }
170 \tl_set:Nn \l_texnegar_fnt_kayhannavaar_tl { kayhannavaar }
171 \tl_set:Nn \l_texnegar_fnt_kayhanpook_tl { kayhanpook }
172 \tl_set:Nn \l_texnegar_fnt_kayhansayeh_tl { kayhansayeh }
173 \tl_set:Nn \l_texnegar_fnt_khoramshahr_tl { khoramshahr }
174 \tl_set:Nn \l_texnegar_fnt_khorramshahr_tl { khorramshahr }
175 \tl_set:Nn \l_texnegar_fnt_niloofar_tl { niloofar }
176 \tl_set:Nn \l_texnegar_fnt_paatch_tl { paatch }
177 \tl_set:Nn \l_texnegar_fnt_riyaz_tl { riyaz }
178 \tl_set:Nn \l_texnegar_fnt_roya_tl { roya }
179 \tl_set:Nn \l_texnegar_fnt_shafigh_tl { shafigh }
180 \tl_set:Nn \l_texnegar_fnt_shafighKurd_tl { shafighKurd }
181 \tl_set:Nn \l_texnegar_fnt_shafighUzbek_tl { shafighUzbek }
182 \tl_set:Nn \l_texnegar_fnt_shiraz_tl { shiraz }
183 \tl_set:Nn \l_texnegar_fnt_sols_tl { sols }
184 \tl_set:Nn \l_texnegar_fnt_tabriz_tl { tabriz }
185 \tl_set:Nn \l_texnegar_fnt_titr_tl { titr }
186 \tl_set:Nn \l_texnegar_fnt_titre_tl { titre }
187 \tl_set:Nn \l_texnegar_fnt_traffic_tl { traffic }
188 \tl_set:Nn \l_texnegar_fnt_vahid_tl { vahid }
189 \tl_set:Nn \l_texnegar_fnt_vosta_tl { vosta }
190 \tl_set:Nn \l_texnegar_fnt_yaghut_tl { yaghut }
191 \tl_set:Nn \l_texnegar_fnt_yagut_tl { yagut }
192 \tl_set:Nn \l_texnegar_fnt_yas_tl { yas }
193 \tl_set:Nn \l_texnegar_fnt_yekan_tl { yekan }
194 \tl_set:Nn \l_texnegar_fnt_yermook_tl { yermook }
195 \tl_set:Nn \l_texnegar_fnt_zar_tl { zar }
196 \tl_set:Nn \l_texnegar_fnt_ziba_tl { ziba }
197 \tl_set:Nn \l_texnegar_fnt_default_tl { default }
198 \tl_set:Nn \l_texnegar_fnt_noskip_tl { noskip }
199
200 \tl_set:Nn \l_texnegar_lig_aalt_tl { aalt } % Access All Alternatives
201 \tl_set:Nn \l_texnegar_lig_ccmp_tl { ccmp } % Glyph Composition/Decomposition
202 \tl_set:Nn \l_texnegar_lig_dlig_tl { dlig } % Discretionary Ligatures
203 \tl_set:Nn \l_texnegar_lig_fina_tl { fina } % Final (Terminal) Forms
204 \tl_set:Nn \l_texnegar_lig_init_tl { init } % Initial Forms
205 \tl_set:Nn \l_texnegar_lig_locl_tl { locl } % Localized Forms
206 \tl_set:Nn \l_texnegar_lig_medi_tl { medi } % Medial Forms

```

```

207 \tl_set:Nn \l_texnegar_lig_rlig_tl { rlig } % Required Ligatures
208 \tl_set:Nn \l_texnegar_lig_default_tl { default }
209
210 \tl_set:Nn \l_texnegar_col_default_tl { magenta }
211
212 \clist_set:Nn \l_texnegar_lig_aalt_clist { } % Access All Alternatives
213 \clist_set:Nn \l_texnegar_lig_ccmp_clist { } % Glyph Composition/Decomposition
214 \clist_set:Nn \l_texnegar_lig_dlig_clist { FDF2 = , FDF3 = , FDFB = } % Discretionary
215 \clist_set:Nn \l_texnegar_lig_fina_clist { } % Final (Terminal) Forms
216 \clist_set:Nn \l_texnegar_lig_init_clist { } % Initial Forms
217 \clist_set:Nn \l_texnegar_lig_locl_clist { } % Localized Forms
218 \clist_set:Nn \l_texnegar_lig_medi_clist { } % Medial Forms
219 \clist_set:Nn \l_texnegar_lig_rlig_clist { } % Required Ligatures
220 \clist_set:Nn \l_texnegar_lig_default_clist { }
221
222 \clist_set:Nn \l_texnegar_lig_names_clist
223 {
224   \l_texnegar_lig_aalt_tl , { \l_texnegar_lig_aalt_clist } ,
225   \l_texnegar_lig_ccmp_tl , { \l_texnegar_lig_ccmp_clist } ,
226   \l_texnegar_lig_dlig_tl , { \l_texnegar_lig_dlig_clist } ,
227   \l_texnegar_lig_fina_tl , { \l_texnegar_lig_fina_clist } ,
228   \l_texnegar_lig_init_tl , { \l_texnegar_lig_init_clist } ,
229   \l_texnegar_lig_locl_tl , { \l_texnegar_lig_locl_clist } ,
230   \l_texnegar_lig_medi_tl , { \l_texnegar_lig_medi_clist } ,
231   \l_texnegar_lig_rlig_tl , { \l_texnegar_lig_rlig_clist } ,
232 }
233
234 \msg_new:nnn { texnegar } { error-kashida-character-is-not-available-in-the-main-
font }
235 {
236   Sorry,~ kashida~ character~ is~ not~ available~ in~ the~ main~ font~#1!
237 }
238
239 \msg_new:nnn { texnegar } { error-value-not-available-for-kashida-option }
240 {
241   Sorry,~ value~ ‘#1’~ is~ not~ available~ for~ ‘Kashida’~ option~ yet~!
242 }
243
244 \msg_new:nnn { texnegar } { error-specify-value-for-kashida-option }
245 {
246   Sorry,~ you~ must~ specify~ a~ value~ for~ ‘Kashida’~ option~ yet~!
247 }
248
249 \msg_new:nnn { texnegar } { warning-experimental-feature }
250 {
251   Please~ note~ that~ the~ feature~ ‘#1’~ is~ still~ experimental~
252   and~ is~ not~ regarded~ as~ stable.
253 }
254
255 \msg_new:nnn { texnegar } { hm-series-font-not-found }
256 {
257   Either~ the~ font~‘#1’~ is~ not~ installed~ on~ your~ system~ or~ does~ not~
258   belong~ to~ HM~Series~fonts.~
259   Please~ note~ that~ the~ option~ ‘Kashida=leaders+glyph’~ is~ currently~ only~

```

```

260 supported~ by~ HM~Series~fonts.~
261 If~ you~ know~ of~ any~ other~ font~ that~ supports~ this~ option,~ please~
262 let~ me~ know~ to~ add~ it~ to~ the~ list~ of~ corresponding~ fonts.~
263 }
264
265 \msg_new:nnn { texnegar } { luatex-version-is-too-old }
266 {
267   #1:~Your~luatex-is-too-old,~you-need-at-least-version-#2.#3~!
268 }
269
270 \keys_define:nn { texnegar }
271 {
272   Kashidafontfamily .code:n =
273   {
274     \tl_set:Nn \l_tmpa_tl { #1 }
275     \tl_case:Nn \l_tmpa_tl
276     {
277       \tl_if_empty:NTF \l_tmpa_tl
278       {
279         \bool_set_false:N \l_texnegar_kashida_fontfamily_bool
280       }
281       {
282         \bool_set_true:N \l_texnegar_kashida_fontfamily_bool
283         \tl_set:Nx \l_texnegar_kashida_fontfamily_tl { \l_tmpa_tl }
284       }
285     }
286   } ,
287
288   Minimal .code:n =
289   {
290     \tl_set:Nn \l_tmpa_tl { #1 }
291     \tl_case:Nn \l_tmpa_tl
292     {
293       \l_texnegar_minimal_off_tl
294       {
295         \bool_set_false:N \l_texnegar_minimal_bool
296       }
297       \l_texnegar_minimal_on_tl
298       {
299         \bool_set_true:N \l_texnegar_minimal_bool
300       }
301     }
302   } ,
303
304   Kashida .code:n =
305   {
306     \tl_set:Nn \l_tmpa_tl { #1 }
307     \tl_case:NnTF \l_tmpa_tl
308     {
309       \l_texnegar_stretch_glyph_tl
310       {
311         \msg_warning:nnn { texnegar } { warning-experimental-feature } { Kashida=gly
312         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_glyph_tl }
313         \AtBeginDocument

```

```

314         {
315             \tl_set:Nx \l_texnegar_main_font_full_tl { \tex_fontname:D \tex_the:D \t
316             \tl_set:Nx \l_texnegar_main_font_name_tl { \l_texnegar_main_font_full_tl
317             \regex_replace_once:nnN { ^"([~/]+)/.* } { \1 } \l_texnegar_main_font_na
318         }
319         \bool_set_true:N \l_texnegar_kashida_fix_bool
320         \bool_set_true:N \l_texnegar_kashida_glyph_bool
321     }
322     \l_texnegar_stretch_leaders_glyph_tl
323     {
324         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
325         \bool_set_true:N \l_texnegar_kashida_fix_bool
326         \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
327     }
328     \l_texnegar_stretch_leaders_hruler_tl
329     {
330         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_hruler_tl
331         \bool_set_true:N \l_texnegar_kashida_fix_bool
332         \bool_set_true:N \l_texnegar_kashida_leaders_hruler_bool
333     }
334     \l_texnegar_stretch_off_tl
335     {
336         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_off_tl }
337         \bool_set_false:N \l_texnegar_kashida_fix_bool
338     }
339     \l_texnegar_stretch_on_tl
340     {
341         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
342         \bool_set_true:N \l_texnegar_kashida_fix_bool
343         \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
344     }
345     } { } { \tl_set:Nx \l_texnegar_gap_filler_tl { #1 } }
346     \tl_if_empty:NT \l_texnegar_gap_filler_tl { \msg_error:nn { texnegar } { error-
specify-value-for-kashida-option } }
347     } ,
348
349     linebreakpenalty .code:n =
350     {
351         \int_set:Nn \l_tmpa_int { #1 }
352         \int_case:nnTF \l_tmpa_int
353         {
354             \l_texnegar_min_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
355             \l_texnegar_low_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
356             \l_texnegar_med_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
357             \l_texnegar_high_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
358             \l_texnegar_max_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
359         } { } { \int_set:Nn \l_texnegar_line_break_penalty_int { #1 } }
360         \bool_set_true:N \l_texnegar_linebreakpenalty_bool
361     } ,
362
363     kashidastretch .code:n =
364     {
365         \tl_set:Nn \l_tmpa_tl { #1 }
366         \tl_case:NnTF \l_tmpa_tl

```

```

367     {
368         \l_texnegar_fnt_kayhan_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
369         \l_texnegar_fnt_kayhannavaar_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
370         \l_texnegar_fnt_kayhanpook_tl  { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
371         \l_texnegar_fnt_kayhansayeh_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
372         \l_texnegar_fnt_khoramshahr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
373         \l_texnegar_fnt_khorramshahr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
374         \l_texnegar_fnt_niloofer_tl    { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
375         \l_texnegar_fnt_paatch_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
376         \l_texnegar_fnt_riyaz_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
377         \l_texnegar_fnt_roya_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
378         \l_texnegar_fnt_shafigh_tl     { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
379         \l_texnegar_fnt_shafighKurd_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
380         \l_texnegar_fnt_shafighUzbek_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
381         \l_texnegar_fnt_shiraz_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
382         \l_texnegar_fnt_sols_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
383         \l_texnegar_fnt_tabriz_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
384         \l_texnegar_fnt_titr_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
385         \l_texnegar_fnt_titre_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
386         \l_texnegar_fnt_traffic_tl     { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
387         \l_texnegar_fnt_vahid_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
388         \l_texnegar_fnt_vosta_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
389         \l_texnegar_fnt_yaghut_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
390         \l_texnegar_fnt_yagut_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
391         \l_texnegar_fnt_yas_tl         { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
392         \l_texnegar_fnt_yekan_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
393         \l_texnegar_fnt_yermook_tl     { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
394         \l_texnegar_fnt_zar_tl         { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
395         \l_texnegar_fnt_ziba_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
396         \l_texnegar_fnt_default_tl     { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
397         \l_texnegar_fnt_noskip_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0
398     } { } { \tl_set:Nn \l_texnegar_skip_default_tl { #1 } }
399 } ,
400 kashidastretch .default:n = \tl_set:Nn \l_texnegar_skip_default_tl { 0 em plus 0.5 em }
401
402 ligatures .code:n =
403 {
404     \tl_set:Nn \l_tmpa_tl { #1 }
405     \tl_case:NnTF \l_tmpa_tl
406     {
407         \l_texnegar_lig_aalt_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
408         \l_texnegar_lig_ccmp_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
409         \l_texnegar_lig_dlig_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
410         \l_texnegar_lig_fina_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
411         \l_texnegar_lig_init_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
412         \l_texnegar_lig_locl_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
413         \l_texnegar_lig_medi_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
414         \l_texnegar_lig_rlig_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
415         \l_texnegar_lig_default_tl   { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
416     } { } { \tl_set:Nn \l_texnegar_active_ligs_tl { #1 } }
417     \bool_set_true:N \l_texnegar_ligature_bool
418 } ,
419 ligatures .default:n = \tl_set:Nn \l_texnegar_active_ligs_tl { \l_texnegar_lig_default_t
420

```

```

421 color .code:n =
422 {
423   \tl_set:Nn \l_tmpa_tl { #1 }
424   \tl_if_empty:NTF \l_tmpa_tl
425   {
426     \tl_set:Nx \l_texnegar_color_tl { \l_texnegar_col_default_tl }
427   }
428   {
429     \tl_set:Nx \l_texnegar_color_tl { \l_tmpa_tl }
430   }
431   \bool_set_true:N \l_texnegar_color_bool
432   \sys_if_engine luatex:T
433   {
434     \convertcolourspec{named}{\l_texnegar_color_tl}{rgb}\l_texnegar_color_rgb_tl
435     \sys_if_engine luatex:T
436     {
437       \directlua{\l_texnegar_color_rgb_tl = "\l_texnegar_color_rgb_tl"}
438     }
439   }
440 } ,
441
442 hboxrecursion .code:n =
443 {
444   \tl_set:Nn \l_tmpa_tl { #1 }
445   \tl_case:NnTF \l_tmpa_tl
446   {
447     \l_texnegar_hboxrecursion_off_tl
448     {
449       \bool_set_false:N \l_texnegar_hboxrecursion_bool
450     }
451     \l_texnegar_hboxrecursion_on_tl
452     {
453       \bool_set_true:N \l_texnegar_hboxrecursion_bool
454     }
455   } { } { \bool_set_false:N \l_texnegar_hboxrecursion_bool }
456 } ,
457 hboxrecursion .default:n = \bool_set_true:N \l_texnegar_hboxrecursion_bool ,
458
459 vboxrecursion .code:n =
460 {
461   \tl_set:Nn \l_tmpa_tl { #1 }
462   \tl_case:NnTF \l_tmpa_tl
463   {
464     \l_texnegar_vboxrecursion_off_tl
465     {
466       \bool_set_false:N \l_texnegar_vboxrecursion_bool
467     }
468     \l_texnegar_vboxrecursion_on_tl
469     {
470       \bool_set_true:N \l_texnegar_vboxrecursion_bool
471     }
472   } { } { \bool_set_false:N \l_texnegar_vboxrecursion_bool }
473 } ,
474 vboxrecursion .default:n = \bool_set_true:N \l_texnegar_vboxrecursion_bool ,

```

```

475 }
476
477 \ProcessKeysOptions { texnegar }
478
479 \sys_if_engine luatex:T
480 {
481     \NewDocumentCommand \KashidaHMFfixOff {} { \directlua{StopStretching()} }
482     \NewDocumentCommand \KashidaHMFfixOn  {} { \directlua{StartStretching()} }
483 }
484
485 \sys_if_engine xetex:T
486 {
487     \NewDocumentCommand \KashidaHMFfixOn {} { \bool_set_true:N \l_texnegar_kashida_fix_bool }
488     \NewDocumentCommand \KashidaHMFfixOff {} { \bool_set_false:N \l_texnegar_kashida_fix_bool }
489 }
490
491 \tex_let:D \KashidaOn \KashidaHMFfixOn
492 \tex_let:D \KashidaOff \KashidaHMFfixOff
493
494 \bool_if:NTF \l_texnegar_kashida_fix_bool
495 {
496     \tl_if_empty:NT \l_texnegar_skip_default_tl { \tl_set:Nn \l_texnegar_skip_default_tl {
497     }
498     {
499         \tl_set:Nv \l_texnegar_skip_default_tl \c_texnegar_skip_a_tl
500     }
501 }
502 %% % \makeatletter
503 %% % \newif\if@Kashida@on
504 %% Becuase Vafa Khalighi has copied the above code (injecting the character uni+200E) in xepersian
505 %% (https://tug.org/svn/texlive/trunk/Master/texmf-dist/tex/xelatex/xepersian/kashida-
506 %% xepersian.def?revision=55165&view=co),
507 %% the following line of code is not needed in xepersian anymore.
508 %% % \newif\if@Kashida@XB@fix
509 %% % \makeatother
510
511 \bool_if:NF \l_texnegar_minimal_bool
512 {
513     \input texnegar-luabidi.tex
514 }
515 \endinput
516 </texnegar-ini-tex>

```

## 1.5 File: texnegar-common-kashida.tex

```

517 <*texnegar-common-kashida-tex>
518 \ProvidesExplFile {texnegar-common-kashida.tex} {2021-01-31} {0.1d} { Full implementation of
519
520 \bool_if:NT \l_texnegar_ligature_bool
521 {
522     \clist_new:N \l_texnegar_ligatures_clist
523     \int_new:N \l_texnegar_lig_names_len_int
524     \int_set:Nn \l_texnegar_lig_names_len_int { \clist_count:N \l_texnegar_lig_names_clist }

```



```

579 \cs_new_protected:Nn \texnegar_put_line_breaks:n
580 {
581   \tl_set:Nn \l_texnegar_line_break_tl { #1 }
582   \regex_replace_all:nnN { ([\])+ } { \ \0 \ \c{texnegar_line_break:}\ } \l_texnegar_li
583   \tl_use:N \l_texnegar_line_break_tl
584 }
585 }
586
587 \endinput
588 </texnegar-common-kashida-tex>

```

## 1.6 File: texnegar-xetex-kashida.tex

```

589 <*texnegar-xetex-kashida-tex>
590 \ProvidesExplFile {texnegar-xetex-kashida.tex} {2021-01-31} {0.1d} { Full implementation of
591
592 \newXeTeXintercharclass \c_texnegar_d_charclass % dual-joiner class
593 \newXeTeXintercharclass \c_texnegar_l_charclass % lam
594 \newXeTeXintercharclass \c_texnegar_r_charclass % right-joiner
595 \newXeTeXintercharclass \c_texnegar_a_charclass % alef
596 \newXeTeXintercharclass \c_texnegar_y_charclass % yeh
597
598 \tex_input:D { texnegar-common-kashida.tex }
599
600 \tl_set:Nn \l_texnegar_use_color_tl
601 {
602   \bool_if:NTF \l_texnegar_color_bool
603   {
604     \colorlet{default}{\l_texnegar_color_tl}
605   }
606   {
607     \colorlet{default}{.}
608   }
609   \color{default}
610 }
611
612 %% Partly adapted from the code provided by David Carlisle in:
613 %% https://tex.stackexchange.com/questions/356709/how-to-know-the-width-and-fill-the-glue-space-between-two-characters-when-using/356721#356721
614 \cs_new:Npn \texnegar_kashida_glyph #1
615 {
616   \bool_if:NT \l_texnegar_kashida_fix_bool
617   {
618     \c_texnegar_lrm_int\tex_penalty:D 10000
619     \mode_leave_vertical:
620     \tex_global:D \tex_advance:D \l_texnegar_counter_int \c_one_int
621
622     \tl_set:Nx \l_texnegar_pos_tl { p\tex_romannumeral:D \l_texnegar_counter_int }
623     \tl_set:Nx \l_texnegar_zref_tl { z\tex_romannumeral:D \l_texnegar_counter_int }
624
625     \zsavposx{x_i\l_texnegar_zref_tl}
626     \tl_set:Nx \l_tmpa_tl
627     {
628       \iow_now:cx { @auxout }
629       {

```



```

684      %% https://tug.org/pipermail/xetex/2009-February/012307.html.
685      %% Somebody notified me that the code in 'kashida-xepersian.def' from xepersian
686      %% package is an exact copy of Jonathan Kew's code. Being unaware of this, in
687      %% the earlier versions of this package I made a mistake and acknowledged
688      %% Vafa Khalighi instead of Jonathan Kew. A sincere thank you to Jonathan Kew
689      %% for his excellent code.
690      \c_texnegar_lrm_int\c_texnegar_zwj_int
691      {\l_texnegar_use_color_tl\tex_penalty:D 10000
692      \tex_leaders:D \tex_hrulerule:D height \XeTeXglyphbounds \c_texnegar_two_int
693      \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int depth \XeTeXglyphbounds \c_texnegar
694      \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int \skip_horizontal:n { #1 }
695      }
696      \c_texnegar_zwj_int
697    }
698  }
699 }
700
701 \XeTeXinterchartokenstate = 1
702
703 \clist_set:Nn \l_texnegar_a_clist { 0622,0623,0625,0627 } %
704 \clist_map_inline:Nn \l_texnegar_a_clist
705 {
706   \XeTeXcharclass "#1 \c_texnegar_a_charclass
707 }
708
709 \clist_set:Nn \l_texnegar_d_clist { 0626,0628,062A,062B,062C,062D,062E,0633,0634,0635,0636,0
710 \clist_map_inline:Nn \l_texnegar_d_clist
711 {
712   \XeTeXcharclass "#1 \c_texnegar_d_charclass
713 }
714
715 \clist_set:Nn \l_texnegar_l_clist { 0644 } %
716 \clist_map_inline:Nn \l_texnegar_l_clist
717 {
718   \XeTeXcharclass "#1 \c_texnegar_l_charclass
719 }
720
721 \clist_set:Nn \l_texnegar_r_clist { 0624,0629,062F,0630,0631,0632,0648,0698 } % , , , , , ,
722 \clist_map_inline:Nn \l_texnegar_r_clist
723 {
724   \XeTeXcharclass "#1 \c_texnegar_r_charclass
725 }
726
727 \clist_set:Nn \l_texnegar_y_clist { 0649,064A,06CC } % , ,
728 \clist_map_inline:Nn \l_texnegar_y_clist
729 {
730   \XeTeXcharclass "#1 \c_texnegar_y_charclass
731 }
732
733 \tl_if_eq:NNTF \l_texnegar_gap_filler_tl \l_texnegar_stretch_glyph_tl {
734   \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
735     \bool_if:NTF \l_texnegar_kashida_fix_bool
736     { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zw
737     { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int

```

```

738 }
739 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
740   \bool_if:NTF \l_texnegar_kashida_fix_bool
741   { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zwj_int
742     { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int
743   }
744 \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
745 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
746 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
747 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_int
748 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_int
749 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_int
750 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_a_charclass = { \c_texnegar_zwj_int
751 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_int
752 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
753 }
754 {
755   \bool_if:NTF {
756     \tl_if_eq_p:NN \l_texnegar_gap_filler_tl \l_texnegar_stretch_leaders_glyph_tl ||
757     \tl_if_eq_p:NN \l_texnegar_gap_filler_tl \l_texnegar_stretch_leaders_hruler_tl
758   }
759   {
760     \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
761       \bool_if:NTF \l_texnegar_kashida_fix_bool
762       { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
763       { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
764     }
765     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
766       \bool_if:NTF \l_texnegar_kashida_fix_bool
767       { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
768       { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
769     }
770     \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders
771     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders
772     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders
773     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \texnegar_kashida_leaders
774     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \texnegar_kashida_leaders
775     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \texnegar_kashida_leaders
776     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \texnegar_kashida_leaders
777     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
778   }
779 }
780 {
781   \msg_error:nnx { texnegar } { error-value-not-available-for-kashida-option } { \l_texnegar_kashida_fix_bool
782 }
783 }
784
785 \endinput
786 </texnegar-xetex-kashida-tex>

```

## 1.7 File: texnegar-char-table.lua

```

787 < *texnegar-char-table-lua>
788 --
789 -- This is file 'texnegar-char-table.lua',

```

```

790 -- generated with the docstrip utility.
791 --
792 -- The original source files were:
793 --
794 -- texnegar.dtx (with options: 'texnegar-char-table-lua')
795 --
796 -- Copyright (C) 2020-2021 Hossein Movahhedian
797 --
798 -- It may be distributed and/or modified under the LaTeX Project Public License,
799 -- version 1.3c or higher (your choice). The latest version of
800 -- this license is at: http://www.latex-project.org/lppl.txt
801 --
802 -- texnegar_char_table = texnegar_char_table or {}
803 -- local texnegar_char_table = texnegar_char_table
804 -- texnegar_char_table.module = {
805 --   name = "texnegar_char_table",
806 --   version = "0.1d",
807 --   date = "2021-01-31",
808 --   description = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
809 --   author = "Hossein Movahhedian",
810 --   copyright = "Hossein Movahhedian",
811 --   license = "LPPL v1.3c"
812 -- }
813 --
814 -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
815 -- local err, warn, info, log = luatexbase.provides_module(texnegar_char_table.module)
816 -- texnegar_char_table.log = log or (function (s) luatexbase.module_info("texnegar_char_table", s))
817 -- texnegar_char_table.warning = warn or (function (s) luatexbase.module_warning("texnegar_char_table", s))
818 -- texnegar_char_table.error = err or (function (s) luatexbase.module_error("texnegar_char_table", s))
819 --
820 local peCharTableDigit = {
821   [1632] = utf8.char(1632), -- "", utf8.codepoint("") == 1632, "\u{0660}", ARABIC-INDIC DIGIT ZERO
822   [1633] = utf8.char(1633), -- "", utf8.codepoint("") == 1633, "\u{0661}", ARABIC-INDIC DIGIT ONE
823   [1634] = utf8.char(1634), -- "", utf8.codepoint("") == 1634, "\u{0662}", ARABIC-INDIC DIGIT TWO
824   [1635] = utf8.char(1635), -- "", utf8.codepoint("") == 1635, "\u{0663}", ARABIC-INDIC DIGIT THREE
825   [1636] = utf8.char(1636), -- "", utf8.codepoint("") == 1636, "\u{0664}", ARABIC-INDIC DIGIT FOUR
826   [1637] = utf8.char(1637), -- "", utf8.codepoint("") == 1637, "\u{0665}", ARABIC-INDIC DIGIT FIVE
827   [1638] = utf8.char(1638), -- "", utf8.codepoint("") == 1638, "\u{0666}", ARABIC-INDIC DIGIT SIX
828   [1639] = utf8.char(1639), -- "", utf8.codepoint("") == 1639, "\u{0667}", ARABIC-INDIC DIGIT SEVEN
829   [1640] = utf8.char(1640), -- "", utf8.codepoint("") == 1640, "\u{0668}", ARABIC-INDIC DIGIT EIGHT
830   [1641] = utf8.char(1641), -- "", utf8.codepoint("") == 1641, "\u{0669}", ARABIC-INDIC DIGIT NINE
831   [1780] = utf8.char(1780), -- "", utf8.codepoint("") == 1780, "\u{06F4}", EXTENDED ARABIC-INDIC DIGIT FOUR
832   [1781] = utf8.char(1781), -- "", utf8.codepoint("") == 1781, "\u{06F5}", EXTENDED ARABIC-INDIC DIGIT FIVE

```

```

INDIC DIGIT FIVE
833 [1782] = utf8.char(1782), -- "", utf8.codepoint("") == 1782, "\u{06F6}", EXTENDED ARABIC
INDIC DIGIT SIX
834 }
835
836 local peCharTablePunctuation = {
837 [1548] = utf8.char(1548), -- "", utf8.codepoint("") == 1548, "\u{060C}", ARABIC COMMA
838 [1549] = utf8.char(1549), -- "", utf8.codepoint("") == 1549, "\u{060D}", ARABIC DATE SE
839 [1563] = utf8.char(1563), -- "", utf8.codepoint("") == 1563, "\u{061B}", ARABIC SEMICOL
840 [1567] = utf8.char(1567), -- "", utf8.codepoint("") == 1567, "\u{061F}", ARABIC QUESTIO
841 [1642] = utf8.char(1642), -- "", utf8.codepoint("") == 1642, "\u{066A}", ARABIC PERCENT
842 [1643] = utf8.char(1643), -- "", utf8.codepoint("") == 1643, "\u{066B}", ARABIC DECIMAL
843 [1644] = utf8.char(1644), -- "", utf8.codepoint("") == 1644, "\u{066C}", ARABIC THOUSAN
844 [1645] = utf8.char(1645), -- "", utf8.codepoint("") == 1645, "\u{066D}", ARABIC FIVE PO
845 }
846
847 local peCharTable = {
848 [1569] = utf8.char(1569), -- "", utf8.codepoint("") == 1569, "\u{0621}", ARABIC LETTE
849 [1570] = utf8.char(1570), -- "", utf8.codepoint("") == 1570, "\u{0622}", ARABIC LETTE
850 [1571] = utf8.char(1571), -- "", utf8.codepoint("") == 1571, "\u{0623}", ARABIC LETTE
851 [1572] = utf8.char(1572), -- "", utf8.codepoint("") == 1572, "\u{0624}", ARABIC LETTE
852 [1573] = utf8.char(1573), -- "", utf8.codepoint("") == 1573, "\u{0625}", ARABIC LETTE
853 [1574] = utf8.char(1574), -- "", utf8.codepoint("") == 1574, "\u{0626}", ARABIC LETTE
854 [1575] = utf8.char(1575), -- "", utf8.codepoint("") == 1575, "\u{0627}", ARABIC LETTE
855 [1576] = utf8.char(1576), -- "", utf8.codepoint("") == 1576, "\u{0628}", ARABIC LETTE
856 [1577] = utf8.char(1577), -- "", utf8.codepoint("") == 1577, "\u{0629}", ARABIC LETTE
857 [1578] = utf8.char(1578), -- "", utf8.codepoint("") == 1578, "\u{062A}", ARABIC LETTE
858 [1579] = utf8.char(1579), -- "", utf8.codepoint("") == 1579, "\u{062B}", ARABIC LETTE
859 [1580] = utf8.char(1580), -- "", utf8.codepoint("") == 1580, "\u{062C}", ARABIC LETTE
860 [1581] = utf8.char(1581), -- "", utf8.codepoint("") == 1581, "\u{062D}", ARABIC LETTE
861 [1582] = utf8.char(1582), -- "", utf8.codepoint("") == 1582, "\u{062E}", ARABIC LETTE
862 [1583] = utf8.char(1583), -- "", utf8.codepoint("") == 1583, "\u{062F}", ARABIC LETTE
863 [1584] = utf8.char(1584), -- "", utf8.codepoint("") == 1584, "\u{0630}", ARABIC LETTE
864 [1585] = utf8.char(1585), -- "", utf8.codepoint("") == 1585, "\u{0631}", ARABIC LETTE
865 [1586] = utf8.char(1586), -- "", utf8.codepoint("") == 1586, "\u{0632}", ARABIC LETTE
866 [1587] = utf8.char(1587), -- "", utf8.codepoint("") == 1587, "\u{0633}", ARABIC LETTE
867 [1588] = utf8.char(1588), -- "", utf8.codepoint("") == 1588, "\u{0634}", ARABIC LETTE
868 [1589] = utf8.char(1589), -- "", utf8.codepoint("") == 1589, "\u{0635}", ARABIC LETTE
869 [1590] = utf8.char(1590), -- "", utf8.codepoint("") == 1590, "\u{0636}", ARABIC LETTE
870 [1591] = utf8.char(1591), -- "", utf8.codepoint("") == 1591, "\u{0637}", ARABIC LETTE
871 [1592] = utf8.char(1592), -- "", utf8.codepoint("") == 1592, "\u{0638}", ARABIC LETTE
872 [1593] = utf8.char(1593), -- "", utf8.codepoint("") == 1593, "\u{0639}", ARABIC LETTE
873 [1594] = utf8.char(1594), -- "", utf8.codepoint("") == 1594, "\u{063A}", ARABIC LETTE
874 [1601] = utf8.char(1601), -- "", utf8.codepoint("") == 1601, "\u{0641}", ARABIC LETTE
875 [1602] = utf8.char(1602), -- "", utf8.codepoint("") == 1602, "\u{0642}", ARABIC LETTE
876 [1603] = utf8.char(1603), -- "", utf8.codepoint("") == 1603, "\u{0643}", ARABIC LETTE
877 [1604] = utf8.char(1604), -- "", utf8.codepoint("") == 1604, "\u{0644}", ARABIC LETTE
878 [1605] = utf8.char(1605), -- "", utf8.codepoint("") == 1605, "\u{0645}", ARABIC LETTE
879 [1606] = utf8.char(1606), -- "", utf8.codepoint("") == 1606, "\u{0646}", ARABIC LETTE
880 [1607] = utf8.char(1607), -- "", utf8.codepoint("") == 1607, "\u{0647}", ARABIC LETTE
881 [1608] = utf8.char(1608), -- "", utf8.codepoint("") == 1608, "\u{0648}", ARABIC LETTE
882 [1609] = utf8.char(1609), -- "", utf8.codepoint("") == 1609, "\u{0649}", ARABIC LETTE
883 [1610] = utf8.char(1610), -- "", utf8.codepoint("") == 1610, "\u{064A}", ARABIC LETTE
884 [1662] = utf8.char(1662), -- "", utf8.codepoint("") == 1662, "\u{067E}", ARABIC LETTE

```

```

885 [1670] = utf8.char(1670), -- "", utf8.codepoint("") == 1670, "\u{0686}", ARABIC LETTE
886 [1688] = utf8.char(1688), -- "", utf8.codepoint("") == 1688, "\u{0698}", ARABIC LETTE
887 [1705] = utf8.char(1705), -- "", utf8.codepoint("") == 1705, "\u{06A9}", ARABIC LETTE
888 [1706] = utf8.char(1706), -- "", utf8.codepoint("") == 1706, "\u{06AA}", ARABIC LETTE
889 [1711] = utf8.char(1711), -- "", utf8.codepoint("") == 1711, "\u{06AF}", ARABIC LETTE
890 [1726] = utf8.char(1726), -- "", utf8.codepoint("") == 1726, "\u{06BE}", ARABIC LETTE
891 [1728] = utf8.char(1728), -- "", utf8.codepoint("") == 1728, "\u{06C0}", ARABIC LETTE
892 [1740] = utf8.char(1740), -- "", utf8.codepoint("") == 1740, "\u{06CC}", ARABIC LETTE
893 [1749] = utf8.char(1749), -- "", utf8.codepoint("") == 1740, "\u{06D5}", ARABIC LETTE
894 [65275] = utf8.char(65275), -- "", utf8.codepoint("") == 65275, "\u{FEFB}", ARABIC LIGA
895 [65276] = utf8.char(65276), -- "", utf8.codepoint("") == 65276, "\u{FEFC}", ARABIC LIGA
896 }
897
898 local peCharTableInitial = {
899 [64344] = utf8.char(64344), -- "", utf8.codepoint("") == 64344, "\u{FB58}", INITIAL FOR
900 [64380] = utf8.char(64380), -- "", utf8.codepoint("") == 64380, "\u{FB7C}", INITIAL FOR
901 [64400] = utf8.char(64400), -- "", utf8.codepoint("") == 64400, "\u{FB90}", INITIAL FOR
902 [64404] = utf8.char(64404), -- "", utf8.codepoint("") == 64404, "\u{FB94}", INITIAL FOR
903 [64510] = utf8.char(64510), -- "", utf8.codepoint("") == 64510, "\u{FBFE}", INITIAL FOR
904 [65169] = utf8.char(65169), -- "", utf8.codepoint("") == 65169, "\u{FE91}", INITIAL FOR
905 [65175] = utf8.char(65175), -- "", utf8.codepoint("") == 65175, "\u{FE97}", INITIAL FOR
906 [65179] = utf8.char(65179), -- "", utf8.codepoint("") == 65179, "\u{FE9B}", INITIAL FOR
907 [65183] = utf8.char(65183), -- "", utf8.codepoint("") == 65183, "\u{FE9F}", INITIAL FOR
908 [65187] = utf8.char(65187), -- "", utf8.codepoint("") == 65187, "\u{FEA3}", INITIAL FOR
909 [65191] = utf8.char(65191), -- "", utf8.codepoint("") == 65191, "\u{FEA7}", INITIAL FOR
910 [65203] = utf8.char(65203), -- "", utf8.codepoint("") == 65203, "\u{FEB3}", INITIAL FOR
911 [65207] = utf8.char(65207), -- "", utf8.codepoint("") == 65207, "\u{FEB7}", INITIAL FOR
912 [65211] = utf8.char(65211), -- "", utf8.codepoint("") == 65211, "\u{FEBB}", INITIAL FOR
913 [65215] = utf8.char(65215), -- "", utf8.codepoint("") == 65215, "\u{FEBF}", INITIAL FOR
914 [65219] = utf8.char(65219), -- "", utf8.codepoint("") == 65219, "\u{FEC3}", INITIAL FOR
915 [65223] = utf8.char(65223), -- "", utf8.codepoint("") == 65223, "\u{FEC7}", INITIAL FOR
916 [65227] = utf8.char(65227), -- "", utf8.codepoint("") == 65227, "\u{FECB}", INITIAL FOR
917 [65231] = utf8.char(65231), -- "", utf8.codepoint("") == 65231, "\u{FECF}", INITIAL FOR
918 [65235] = utf8.char(65235), -- "", utf8.codepoint("") == 65235, "\u{FED3}", INITIAL FOR
919 [65239] = utf8.char(65239), -- "", utf8.codepoint("") == 65239, "\u{FED7}", INITIAL FOR
920 [65243] = utf8.char(65243), -- "", utf8.codepoint("") == 65243, "\u{FEDB}", INITIAL FOR
921 [65247] = utf8.char(65247), -- "", utf8.codepoint("") == 65247, "\u{FEDF}", INITIAL FOR
922 [65251] = utf8.char(65251), -- "", utf8.codepoint("") == 65251, "\u{FEE3}", INITIAL FOR
923 [65255] = utf8.char(65255), -- "", utf8.codepoint("") == 65255, "\u{FEE7}", INITIAL FOR
924 [65259] = utf8.char(65259), -- "", utf8.codepoint("") == 65259, "\u{FEEB}", INITIAL FOR
925 [65267] = utf8.char(65267), -- "", utf8.codepoint("") == 65267, "\u{FEF3}", INITIAL FOR
926 }
927
928 local peCharTableMedial = {
929 [1600] = utf8.char(1600), -- "", utf8.codepoint("") == 1600, "\u{0640}", ARABIC TATW
930 [64345] = utf8.char(64345), -- "", utf8.codepoint("") == 64345, "\u{FB59}", MEDIAL FORM
931 [64381] = utf8.char(64381), -- "", utf8.codepoint("") == 64381, "\u{FB7D}", MEDIAL FORM
932 [64401] = utf8.char(64401), -- "", utf8.codepoint("") == 64401, "\u{FB91}", MEDIAL FORM
933 [64405] = utf8.char(64405), -- "", utf8.codepoint("") == 64405, "\u{FB95}", MEDIAL FORM
934 [64425] = utf8.char(64425), -- "", utf8.codepoint("") == 64425, "\u{FBA9}", MEDIAL FORM
935 [64429] = utf8.char(64429), -- "", utf8.codepoint("") == 64429, "\u{FBAD}", MEDIAL FORM
936 [64511] = utf8.char(64511), -- "", utf8.codepoint("") == 64511, "\u{FBFF}", MEDIAL FORM
937 [65170] = utf8.char(65170), -- "", utf8.codepoint("") == 65170, "\u{FE92}", MEDIAL FORM
938 [65176] = utf8.char(65176), -- "", utf8.codepoint("") == 65176, "\u{FE98}", MEDIAL FORM

```

```

939 [65180] = utf8.char(65180), -- "", utf8.codepoint("") == 65180, "\u{FE9C}", MEDIAL FORM
940 [65184] = utf8.char(65184), -- "", utf8.codepoint("") == 65184, "\u{FEA0}", MEDIAL FORM
941 [65188] = utf8.char(65188), -- "", utf8.codepoint("") == 65188, "\u{FEA4}", MEDIAL FORM
942 [65192] = utf8.char(65192), -- "", utf8.codepoint("") == 65192, "\u{FEA8}", MEDIAL FORM
943 [65204] = utf8.char(65204), -- "", utf8.codepoint("") == 65204, "\u{FEB4}", MEDIAL FORM
944 [65208] = utf8.char(65208), -- "", utf8.codepoint("") == 65208, "\u{FEB8}", MEDIAL FORM
945 [65212] = utf8.char(65212), -- "", utf8.codepoint("") == 65212, "\u{FEB C}", MEDIAL FORM
946 [65216] = utf8.char(65216), -- "", utf8.codepoint("") == 65216, "\u{FEC0}", MEDIAL FORM
947 [65220] = utf8.char(65220), -- "", utf8.codepoint("") == 65220, "\u{FEC4}", MEDIAL FORM
948 [65224] = utf8.char(65224), -- "", utf8.codepoint("") == 65224, "\u{FEC8}", MEDIAL FORM
949 [65228] = utf8.char(65228), -- "", utf8.codepoint("") == 65228, "\u{FECC}", MEDIAL FORM
950 [65232] = utf8.char(65232), -- "", utf8.codepoint("") == 65232, "\u{FED0}", MEDIAL FORM
951 [65236] = utf8.char(65236), -- "", utf8.codepoint("") == 65236, "\u{FED4}", MEDIAL FORM
952 [65240] = utf8.char(65240), -- "", utf8.codepoint("") == 65240, "\u{FED8}", MEDIAL FORM
953 [65244] = utf8.char(65244), -- "", utf8.codepoint("") == 65244, "\u{FEDC}", MEDIAL FORM
954 [65248] = utf8.char(65248), -- "", utf8.codepoint("") == 65248, "\u{FEE0}", MEDIAL FORM
955 [65252] = utf8.char(65252), -- "", utf8.codepoint("") == 65252, "\u{FEE4}", MEDIAL FORM
956 [65256] = utf8.char(65256), -- "", utf8.codepoint("") == 65256, "\u{FEE8}", MEDIAL FORM
957 [65260] = utf8.char(65260), -- "", utf8.codepoint("") == 65260, "\u{FEEC}", MEDIAL FORM
958 [65268] = utf8.char(65268), -- "", utf8.codepoint("") == 65268, "\u{FEF4}", MEDIAL FORM
959 }
960
961 local peCharTableFinal = {
962 [64343] = utf8.char(64343), -- "", utf8.codepoint("") == 64343, "\u{FB57}", FINAL FORM
963 [64379] = utf8.char(64379), -- "", utf8.codepoint("") == 64379, "\u{FB7B}", FINAL FORM
964 [64395] = utf8.char(64395), -- "", utf8.codepoint("") == 64395, "\u{FB8B}", FINAL FORM
965 [64399] = utf8.char(64399), -- "", utf8.codepoint("") == 64399, "\u{FB8F}", FINAL FORM
966 [64403] = utf8.char(64403), -- "", utf8.codepoint("") == 64403, "\u{FB93}", FINAL FORM
967 [64421] = utf8.char(64421), -- "", utf8.codepoint("") == 64421, "\u{FBA5}", FINAL FORM
968 [64509] = utf8.char(64509), -- "", utf8.codepoint("") == 64509, "\u{FBFD}", FINAL FORM
969 [65166] = utf8.char(65166), -- "", utf8.codepoint("") == 65166, "\u{FE8E}", FINAL FORM
970 [65168] = utf8.char(65168), -- "", utf8.codepoint("") == 65168, "\u{FE90}", FINAL FORM
971 [65172] = utf8.char(65172), -- "", utf8.codepoint("") == 65172, "\u{FE94}", FINAL FORM
972 [65174] = utf8.char(65174), -- "", utf8.codepoint("") == 65174, "\u{FE96}", FINAL FORM
973 [65178] = utf8.char(65178), -- "", utf8.codepoint("") == 65178, "\u{FE9A}", FINAL FORM
974 [65182] = utf8.char(65182), -- "", utf8.codepoint("") == 65182, "\u{FE9E}", FINAL FORM
975 [65186] = utf8.char(65186), -- "", utf8.codepoint("") == 65186, "\u{FEA2}", FINAL FORM
976 [65190] = utf8.char(65190), -- "", utf8.codepoint("") == 65190, "\u{FEA6}", FINAL FORM
977 [65194] = utf8.char(65194), -- "", utf8.codepoint("") == 65194, "\u{FEAA}", FINAL FORM
978 [65196] = utf8.char(65196), -- "", utf8.codepoint("") == 65196, "\u{FEAC}", FINAL FORM
979 [65198] = utf8.char(65198), -- "", utf8.codepoint("") == 65198, "\u{FEAE}", FINAL FORM
980 [65200] = utf8.char(65200), -- "", utf8.codepoint("") == 65200, "\u{FEB0}", FINAL FORM
981 [65202] = utf8.char(65202), -- "", utf8.codepoint("") == 65202, "\u{FEB2}", FINAL FORM
982 [65206] = utf8.char(65206), -- "", utf8.codepoint("") == 65206, "\u{FEB6}", FINAL FORM
983 [65210] = utf8.char(65210), -- "", utf8.codepoint("") == 65210, "\u{FEB A}", FINAL FORM
984 [65214] = utf8.char(65214), -- "", utf8.codepoint("") == 65214, "\u{FEBE}", FINAL FORM
985 [65218] = utf8.char(65218), -- "", utf8.codepoint("") == 65218, "\u{FEC2}", FINAL FORM
986 [65222] = utf8.char(65222), -- "", utf8.codepoint("") == 65222, "\u{FEC6}", FINAL FORM
987 [65226] = utf8.char(65226), -- "", utf8.codepoint("") == 65226, "\u{FECA}", FINAL FORM
988 [65230] = utf8.char(65230), -- "", utf8.codepoint("") == 65230, "\u{FECE}", FINAL FORM
989 [65234] = utf8.char(65234), -- "", utf8.codepoint("") == 65234, "\u{FED2}", FINAL FORM
990 [65238] = utf8.char(65238), -- "", utf8.codepoint("") == 65238, "\u{FED6}", FINAL FORM
991 [65242] = utf8.char(65242), -- "", utf8.codepoint("") == 65242, "\u{FEDA}", FINAL FORM
992 [65246] = utf8.char(65246), -- "", utf8.codepoint("") == 65246, "\u{FEDE}", FINAL FORM

```

```

993 [65250] = utf8.char(65250), -- "", utf8.codepoint("") == 65250, "\u{FEE2}", FINAL FORM
994 [65254] = utf8.char(65254), -- "", utf8.codepoint("") == 65254, "\u{FEE6}", FINAL FORM
995 [65258] = utf8.char(65258), -- "", utf8.codepoint("") == 65258, "\u{FEEA}", FINAL FORM
996 [65262] = utf8.char(65262), -- "", utf8.codepoint("") == 65262, "\u{FEEE}", FINAL FORM
997 [65264] = utf8.char(65264), -- "", utf8.codepoint("") == 65264, "\u{FEF0}", FINAL FORM
998 [65266] = utf8.char(65266), -- "", utf8.codepoint("") == 65266, "\u{FEF2}", FINAL FORM
999 [65276] = utf8.char(65276), -- "", utf8.codepoint("") == 65276, "\u{FEFC}", FINAL FORM
1000 }
1001
1002 return peCharTableInitial, peCharTableMedial, peCharTableFinal
1003 --
1004 --
1005 -- End of file 'texnegar-char-table.lua'.
1006 </texnegar-char-table.lua>

```

## 1.8 File: texnegar.lua

```

1007 <*texnegar-lua>
1008 --
1009 -- This is file 'texnegar.lua',
1010 -- generated with the docstrip utility.
1011 --
1012 -- The original source files were:
1013 --
1014 -- texnegar.dtx (with options: 'texnegar-lua')
1015 --
1016 -- Copyright (C) 2020-2021 Hossein Movahhedian
1017 --
1018 -- It may be distributed and/or modified under the LaTeX Project Public License,
1019 -- version 1.3c or higher (your choice). The latest version of
1020 -- this license is at: http://www.latex-project.org/lppl.txt
1021 --
1022 -- texnegar      = texnegar or {}
1023 -- local texnegar = texnegar
1024 -- texnegar.module = {
1025 --   name      = "texnegar",
1026 --   version   = "0.1d",
1027 --   date      = "2021-01-31",
1028 --   description = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
1029 --   author     = "Hossein Movahhedian",
1030 --   copyright  = "Hossein Movahhedian",
1031 --   license    = "LPPL v1.3c"
1032 -- }
1033 --
1034 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
1035 -- local err, warn, info, log = luatexbase.provides_module(texnegar.module)
1036 -- texnegar.log      = log or (function (s) luatexbase.module_info("texnegar", s) end)
1037 -- texnegar.warning = warn or (function (s) luatexbase.module_warning("texnegar", s) end)
1038 -- texnegar.error   = err or (function (s) luatexbase.module_error("texnegar", s) end)
1039
1040 local l_texnegar_kashida_fontfamily_bool = token.create("l_texnegar_kashida_fontfamily_bool")
1041
1042 local debug_getinfo = debug.getinfo
1043 local string_format = string.format
1044

```

```

1045 function TableLength(t)
1046     local i = 0
1047     for _ in pairs(t) do
1048         i = i + 1
1049     end
1050     return i
1051 end
1052
1053 tex.enableprimitives('',tex.extraprimitives ())
1054
1055 local range_tble = {
1056     [1536] = 1791,
1057     [1872] = 1919,
1058     [2208] = 2274,
1059     [8204] = 8297,
1060     [64336] = 65023,
1061     [65136] = 65279,
1062     [126464] = 126719,
1063     [983040] = 1048575
1064 }
1065
1066 local tbl_fonts_used = { }
1067 local tbl_fonts_chars = { }
1068 local tbl_fonts_chars_init = { }
1069 local tbl_fonts_chars_medi = { }
1070 local tbl_fonts_chars_fina = { }
1071
1072 local pattern_list = {
1073     ".*%.(ini)t?$", ".*%.(ini)t?%..*",
1074     ".*%.(med)i?$", ".*%.(med)i?%..*",
1075     ".*%.(fin)a?$", ".*%.(fin)a?%..*",
1076
1077     ".*_(ini)t?$", ".*_(ini)t?_.*",
1078     ".*_(med)i?$", ".*_(med)i?_.*",
1079     ".*_(fin)a?$", ".*_(fin)a?_.*",
1080 }
1081
1082 function GetFontsChars()
1083     local funcName = debug_getinfo(1).name
1084     local funcNparams = debug_getinfo(1).nparams
1085
1086     for f_num = 1, font.max() do
1087         local f_tmp = font.fonts[f_num]
1088         if f_tmp then
1089             local f_tmp_fontname = f_tmp.fontname
1090             if f_tmp_fontname then
1091                 local f_id_tmp = font.getfont(f_num)
1092                 local f_fontname_tmp = f_id_tmp.fontname
1093                 local f_filename_tmp = f_id_tmp.filename
1094                 if not tbl_fonts_used[f_fontname_tmp] then
1095                     tbl_fonts_used[f_fontname_tmp] = {f_filename_tmp, f_id_tmp}
1096                 end
1097             end
1098         end
1099     end
1100 end

```

```

1099     end
1100
1101     for f_fontname, v in pairs(tbl_fonts_used) do
1102         f_filename = v[1]
1103         f_id = v[2]
1104         if not tbl_fonts_chars[f_fontname] then
1105             tbl_fonts_chars[f_fontname] = { }
1106             tbl_fonts_chars_init[f_fontname] = { }
1107             tbl_fonts_chars_medi[f_fontname] = { }
1108             tbl_fonts_chars_fina[f_fontname] = { }
1109             local f = fontloader.open(f_filename)
1110             local char_name
1111             local char_unicode
1112             local char_class
1113             for k, v in pairs(range_tble) do
1114                 for glyph_idx = k, v do
1115                     if f_id.characters[glyph_idx] then
1116                         char_name = f.glyphs[f_id.characters[glyph_idx].index].name
1117                         char_unicode = f.glyphs[f_id.characters[glyph_idx].index].unicode
1118                         char_class = f.glyphs[f_id.characters[glyph_idx].index].class
1119
1120                         kashida_fontfamily = token.get_macro("l_texnegar_kashida_fontfamily_")
1121                         fontfamily_match = string.match(f_fontname, "^(" .. kashida_fontfamily_
1122                         if fontfamily_match == kashida_fontfamily then
1123                             if not tbl_fonts_chars[f_fontname][glyph_idx] then
1124                                 if string.match(f_fontname, "^(Amiri).*)" == "Amiri" and ch
1125                                     current_kashida_unicode = glyph_idx
1126                                 end
1127                                 tbl_fonts_chars[f_fontname][glyph_idx] = {char_name, char_un
1128                                 for _, pattern in ipairs( pattern_list ) do
1129                                     local pos_alt = string.match(char_name, pattern)
1130                                     if pos_alt == 'ini' or pos_alt == 'AltIni' then
1131                                         tbl_fonts_chars_init[f_fontname][glyph_idx] = {char_
1132                                     elseif pos_alt == 'med' or pos_alt == 'AltMed' then
1133                                         tbl_fonts_chars_medi[f_fontname][glyph_idx] = {char_
1134                                     elseif pos_alt == 'fin' or pos_alt == 'AltFin' then
1135                                         tbl_fonts_chars_fina[f_fontname][glyph_idx] = {char_
1136                                     end
1137                                 end
1138                             end
1139                         end
1140                     end
1141                 end
1142             end
1143             fontloader.close(f)
1144         end
1145     end
1146     return tbl_fonts_used, tbl_fonts_chars, tbl_fonts_chars_init, tbl_fonts_chars_medi, tbl_
1147 end
1148
1149 dofile(kpse.find_file("texnegar-ini.lua"))
1150 --
1151 --
1152 -- End of file 'texnegar.lua'.

```

```
1153 </texnegar-lua>
```

## 1.9 File: texnegar-ini.lua

```
1154 <*texnegar-ini-lua>
```

```
1155 --
```

```
1156 -- This is file 'texnegar-ini.lua',  
1157 -- generated with the docstrip utility.
```

```
1158 --
```

```
1159 -- The original source files were:
```

```
1160 --
```

```
1161 -- texnegar.dtx (with options: 'texnegar-ini-lua')
```

```
1162 --
```

```
1163 -- Copyright (C) 2020-2021 Hossein Movahhedian
```

```
1164 --
```

```
1165 -- It may be distributed and/or modified under the LaTeX Project Public License,  
1166 -- version 1.3c or higher (your choice). The latest version of  
1167 -- this license is at: http://www.latex-project.org/lppl.txt
```

```
1168 --
```

```
1169 -- texnegar_ini = texnegar_ini or {}
```

```
1170 -- local texnegar_ini = texnegar_ini
```

```
1171 -- texnegar_ini.module = {
```

```
1172 --   name = "texnegar_ini",
```

```
1173 --   version = "0.1d",
```

```
1174 --   date = "2021-01-31",
```

```
1175 --   description = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
```

```
1176 --   author = "Hossein Movahhedian",
```

```
1177 --   copyright = "Hossein Movahhedian",
```

```
1178 --   license = "LPPL v1.3c"
```

```
1179 -- }
```

```
1180 --
```

```
1181 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
```

```
1182 -- local err, warn, info, log = luatexbase.provides_module(texnegar_ini.module)
```

```
1183 -- texnegar_ini.log = log or (function (s) luatexbase.module_info("texnegar_ini", s)
```

```
1184 -- texnegar_ini.warning = warn or (function (s) luatexbase.module_warning("texnegar_ini", s)
```

```
1185 -- texnegar_ini.error = err or (function (s) luatexbase.module_error("texnegar_ini", s)
```

```
1186
```

```
1187 c_true_bool = token.create("c_true_bool")
```

```
1188
```

```
1189 l_texnegar_color_bool = token.create("l_texnegar_color_bool")
```

```
1190
```

```
1191 if l_texnegar_color_bool.mode == c_true_bool.mode then
```

```
1192   color_tbl = color_tbl or {}
```

```
1193   for item in l_texnegar_color_rgb_tl:gmatch("[^,%s]+") do
```

```
1194     table.insert(color_tbl, item)
```

```
1195   end
```

```
1196 end
```

```
1197
```

```
1198 dofile(kpse.find_file("texnegar-luatex-kashida.lua"))
```

```
1199 --
```

```
1200 --
```

```
1201 -- End of file 'texnegar-ini.lua'.
```

```
1202 </texnegar-ini-lua>
```

## 1.10 File: texnegar-luatex-kashida.lua

```
1203 ⟨*texnegar-luatex-kashida.lua⟩
1204 --
1205 -- This is file 'texnegar-luatex-kashida.lua',
1206 -- generated with the docstrip utility.
1207 --
1208 -- The original source files were:
1209 --
1210 -- texnegar.dtx (with options: 'texnegar-luatex-kashida.lua')
1211 --
1212 -- Copyright (C) 2020-2021 Hossein Movahhedian
1213 --
1214 -- It may be distributed and/or modified under the LaTeX Project Public License,
1215 -- version 1.3c or higher (your choice). The latest version of
1216 -- this license is at: http://www.latex-project.org/lppl.txt
1217 --
1218 -- texnegar_luatex_kashida = texnegar_luatex_kashida or {}
1219 -- local texnegar_luatex_kashida = texnegar_luatex_kashida
1220 -- texnegar_luatex_kashida.module = {
1221 --     name = "texnegar_luatex_kashida",
1222 --     version = "0.1d",
1223 --     date = "2021-01-31",
1224 --     description = "Full implementation of kashida feature in XeLaTeX and L
1225 --     author = "Hossein Movahhedian",
1226 --     copyright = "Hossein Movahhedian",
1227 --     license = "LPPL v1.3c"
1228 -- }
1229 --
1230 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
1231 -- local err, warn, info, log = luatexbase.provides_module(texnegar_luatex_kashida.module)
1232 -- texnegar_luatex_kashida.log = log or (function (s) luatexbase.module_info("texnegar_
1233 -- texnegar_luatex_kashida.warning = warn or (function (s) luatexbase.module_warning("texneg
1234 -- texnegar_luatex_kashida.error = err or (function (s) luatexbase.module_error("texnegar
1235
1236 local peCharTableInitial, peCharTableMedial, peCharTableFinal = dofile(kpse.find_file("texne
char-table.lua"))
1237
1238 local kashida_unicode = 1600
1239 local kashida_subtype = 256
1240
1241 local COLORSTACK = node.subtype("pdf_colorstack")
1242 local node_id = node.id
1243 local GLUE = node_id("glue")
1244 local GLYPH = node_id("glyph")
1245 local HLIST = node_id("hlist")
1246 local RULE = node_id("rule")
1247 local WHATSIT = node_id("whatsit")
1248
1249 local l_texnegar_kashida_glyph_bool = token.create("l_texnegar_kashida_glyph_bool")
1250 local l_texnegar_kashida_leaders_glyph_bool = token.create("l_texnegar_kashida_leaders_glyph
1251 local l_texnegar_kashida_leaders_hrulerule_bool = token.create("l_texnegar_kashida_leaders_hruler
1252
1253 local l_texnegar_hboxrecursion_bool = token.create("l_texnegar_hboxrecursion_bool")
```

```

1254 local l_textrnegar_vboxrecursion_bool      = token.create("l_textrnegar_vboxrecursion_bool")
1255
1256 local selected_font = font.current()
1257 local selected_font_old = selected_font
1258
1259 local string_format = string.format
1260 local debug_getinfo = debug.getinfo
1261
1262 function GetGlyphDimensions(font_file, glyph_index)
1263     local funcName      = debug_getinfo(1).name
1264     local funcNparams   = debug_getinfo(1).nparams
1265
1266     local fnt = fontloader.open(font_file)
1267     local idx = 0
1268     local fnt_glyphcnt = fnt.glyphcnt
1269     local fnt_glyphmin = fnt.glyphmin
1270     local fnt_glyphmax = fnt.glyphmax
1271     if fnt_glyphcnt > 0 then
1272         for idx = fnt_glyphmin, fnt_glyphmax do
1273             local gl = fnt.glyphs[idx]
1274             if gl then
1275                 local gl_unicode = gl.unicode
1276                 if gl_unicode == glyph_index then
1277                     local gl_name      = gl.name
1278                     gl_width      = gl.width
1279                     local gl_bbox      = gl.boundingBox
1280                     gl_llx          = gl_bbox[1]
1281                     gl_depth        = gl_bbox[2]
1282                     gl_urx          = gl_bbox[3]
1283                     gl_height       = gl_bbox[4]
1284                     break
1285                 end
1286             end
1287             idx = idx + 1
1288         end
1289     end
1290     fontloader.close(fnt)
1291     return {width = gl_width, height = gl_height, depth = gl_depth, llx = gl_llx, urx = gl_urx}
1292 end
1293
1294 function GetGlue(t_plb_line_glue_node, t_plb_node)
1295     local funcName      = debug_getinfo(1).name
1296     local funcNparams   = debug_getinfo(1).nparams
1297
1298     local glue_id        = t_plb_line_glue_node.id
1299     local glue_subtype   = t_plb_line_glue_node.subtype
1300     local glue_width     = t_plb_line_glue_node.width
1301     local glue_stretch   = t_plb_line_glue_node.stretch
1302     local glue_shrink    = t_plb_line_glue_node.shrink
1303     local eff_glue_width = node.effective_glue(t_plb_line_glue_node, t_plb_node)
1304     local glue_stretch_order = t_plb_line_glue_node.stretch_order
1305     local glue_shrink_order = t_plb_line_glue_node.shrink_order
1306     local glue_delta     = 0
1307     glue_delta = eff_glue_width - glue_width

```

```

1308     return { id = glue_id, subtype = glue_subtype, width = glue_width, stretch = glue_stretch,
1309             shrink = glue_shrink, stretch_order = glue_stretch_order, shrink_order = glue_shrink_order,
1310             effective_glue = eff_glue_width, delta = glue_delta }
1311 end
1312
1313 function GetGlyph(t_plb_line_glyph_node, t_tbl_line_fields, t_CharTableInitial, t_CharTableMedial, t_CharTableFinal)
1314     local funcName      = debug_getinfo(1).name
1315     local funcNparams    = debug_getinfo(1).nparams
1316
1317     local glyph_id       = t_plb_line_glyph_node.id
1318     local glyph_subtype   = t_plb_line_glyph_node.subtype
1319     local glyph_char      = t_plb_line_glyph_node.char
1320     local glyph_font      = t_plb_line_glyph_node.font
1321     local glyph_lang      = t_plb_line_glyph_node.lang
1322     local glyph_width     = t_plb_line_glyph_node.width
1323     local glyph_data      = t_plb_line_glyph_node.data
1324
1325     if not (t_CharTableInitial[glyph_char] == nil) then
1326         t_tbl_line_fields.joinerCharInitial = t_tbl_line_fields.joinerCharInitial + 1
1327         t_plb_line_glyph_node.data = 1
1328     elseif not (t_CharTableMedial[glyph_char] == nil) then
1329         t_tbl_line_fields.joinerCharMedial = t_tbl_line_fields.joinerCharMedial + 1
1330         t_plb_line_glyph_node.data = 2
1331     elseif not (t_CharTableFinal[glyph_char] == nil) then
1332         t_tbl_line_fields.joinerCharFinal = t_tbl_line_fields.joinerCharFinal + 1
1333         t_plb_line_glyph_node.data = 3
1334     end
1335     return { id = glyph_id, subtype = glyph_subtype, char = glyph_char, font = glyph_font, lang = glyph_lang, width = glyph_width, data = glyph_data }
1336 end
1337
1338 function ProcessTableKashidaHlist(ksh_hlistNode, hbox_num, in_font)
1339     local funcName      = debug_getinfo(1).name
1340     local funcNparams    = debug_getinfo(1).nparams
1341
1342     local ksh_hlistNode_id = ksh_hlistNode.id
1343     local ksh_hlistNode_subtype = ksh_hlistNode.subtype
1344
1345     for tn in node.traverse(ksh_hlistNode.head) do
1346         local tn_id = tn.id
1347         local tn_subtype = tn.subtype
1348
1349         if tn_id == 0 then
1350             for tp in node.traverse(tn.head) do
1351                 local tp_id = tp.id
1352                 local tp_subtype = tp.subtype
1353                 if tp_id == 29 then
1354                     if l_textrn_color_bool.mode == c_true_bool.mode then
1355                         local col_str = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color_tbl[3] .. " " .. color_tbl[4]
1356                         local col_str_rg = col_str .. " rg "
1357                         local col_str_RG = col_str .. " RG"
1358
1359                         local color_push = node.new(WHATSIT, COLORSTACK)
1360                         local color_pop = node.new(WHATSIT, COLORSTACK)
1361                         color_push.stack = 0

```

```

1362         color_pop.stack      = 0
1363         color_push.command = 1
1364         color_pop.command  = 2
1365         glue_ratio         = .2
1366         color_push.data     = col_str_rg .. col_str_RG
1367         color_pop.data      = col_str_rg .. col_str_RG
1368         tn.head = node.insert_before(tn.list, tn.head, node.copy(color_push))
1369         tn.head = node.insert_after(tn.list, node.tail(tn.head), node.copy(color_pop))
1370     end
1371
1372     local tp_font = tp.font
1373     local tp_char = tp.char
1374     tp.font = in_font
1375
1376     local ksh_unicode
1377     ksh_unicode = font.getfont(in_font).resources.unicodes['kashida']
1378     if hbox_num == 'l_texnegar_k_box' then
1379         tp.char = current_kashida_unicode or kashida_unicode
1380     elseif hbox_num == 'l_texnegar_ksh_box' then
1381         tp.char = ksh_unicode
1382         tn_width = tn.width
1383         ksh_hlistNode.width = tn_width
1384     end
1385     elseif tp_id == 0 then
1386         if tp.subtype ~= 3 then
1387             tbl_kashida_hlist_nodes[ #tbl_kashida_hlist_nodes + 1 ] = tp
1388         end
1389     end
1390 end
1391 elseif tn_id == 1 then
1392     do end
1393 elseif tn_id == 8 then
1394     do end
1395 elseif tn_id == 29 then
1396     if l_texnegar_color_bool.mode == c_true_bool.mode then
1397         local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color_tbl[3]
1398         local col_str_rg   = col_str .. " rg "
1399         local col_str_RG   = col_str .. " RG"
1400
1401         local color_push   = node.new(WHATSIT, COLORSTACK)
1402         local color_pop    = node.new(WHATSIT, COLORSTACK)
1403         color_push.stack   = 0
1404         color_pop.stack    = 0
1405         color_push.command = 1
1406         color_pop.command  = 2
1407         glue_ratio         = .2
1408         color_push.data     = col_str_rg .. col_str_RG
1409         color_pop.data      = col_str_rg .. col_str_RG
1410         ksh_hlistNode.head = node.insert_before(ksh_hlistNode.list, ksh_hlistNode.head, color_push)
1411         ksh_hlistNode.head = node.insert_after(ksh_hlistNode.list, node.tail(ksh_hlistNode.head), color_pop)
1412     end
1413
1414     local tn_font = tn.font
1415     local tn_char = tn.char

```

```

1416         tn.font = in_font
1417
1418         local ksh_unicode
1419         ksh_unicode = font.getfont(in_font).resources.unicodes['kashida']
1420         if hbox_num == 'l_texnegar_k_box' then
1421             tn.char = kashida_unicode
1422         elseif hbox_num == 'l_texnegar_ksh_box' then
1423             tn.char = ksh_unicode
1424             tn_width = tn.width
1425             ksh_hlistNode.width = tn_width
1426         end
1427     else
1428         print(string_format("\n tn. Not processed node id is: %d", tn_id))
1429     end
1430 end
1431 end
1432
1433 function SetFontInHbox(hbox_num, font_num)
1434     local funcName      = debug_getinfo(1).name
1435     local funcNparams   = debug_getinfo(1).nparams
1436
1437     tbl_kashida_hlist_nodes = {}
1438
1439     local tmp_node
1440     tmp_node = node.new("hlist")
1441     tmp_node = tex.getbox(hbox_num)
1442
1443     ProcessTableKashidaHlist(tmp_node, hbox_num, font_num)
1444
1445     ::kashida_hlist_BEGIN::
1446     if #tbl_kashida_hlist_nodes > 0 then
1447         local kashida_hlistNodeAdded = table.remove(tbl_kashida_hlist_nodes,1)
1448         ProcessTableKashidaHlist(kashida_hlistNodeAdded, hbox_num, font_num)
1449         goto kashida_hlist_BEGIN
1450     end
1451 end
1452
1453 function StretchGlyph(t_plb_node, t_plb_glyph_node, t_gluePerJoiner, t_dir, t_filler)
1454     local funcName      = debug_getinfo(1).name
1455     local funcNparams   = debug_getinfo(1).nparams
1456
1457     if t_filler == "resized_kashida" then
1458         SetFontInHbox('l_texnegar_k_box', selected_font)
1459     elseif t_filler == "leaders+kashida" then
1460         SetFontInHbox('l_texnegar_ksh_box', selected_font)
1461     end
1462
1463     kashida_node = node.new(GLYPH)
1464     node_glue    = node.new(GLUE)
1465     node_rule    = node.new(RULE)
1466     node_hlist   = node.new(HLIST)
1467
1468     font_current = selected_font
1469     font_name    = font.fonts[font_current].fullname

```

```

1470 font_file      = font.fonts[font_current].filename
1471 kashida_char = font.fonts[font_current].characters[1600]
1472
1473 kashida_node.subtype = kashida_subtype
1474 kashida_node.font    = font_current
1475 if string.match(font_name, "^(Amiri).*") == "Amiri" then
1476     kashida_node.char = current_kashida_unicode
1477 else
1478     kashida_node.char = kashida_unicode
1479 end
1480 kashida_node.lang    = tex.language
1481
1482 kashida_width = kashida_node.width
1483 kashida_height = kashida_node.height
1484 kashida_depth = kashida_node.depth
1485
1486 tbl_gl_dimen = GetGlyphDimensions(font_file, kashida_unicode)
1487 ksh_width, ksh_height, ksh_depth, ksh_llx, ksh_urx =
1488     tbl_gl_dimen.width, tbl_gl_dimen.height, tbl_gl_dimen.depth, tbl_gl_dimen.llx, tbl_g
1489
1490 ratio_width = kashida_width / ksh_width
1491 leaders_height = ratio_width * ksh_height
1492 leaders_depth = - ratio_width * ksh_depth
1493
1494 node_glue.subtype = 100
1495 node.setglue(node_glue, t_gluePerJoiner, 0, 0, 0, 0)
1496
1497 if t_filler == "resized_kashida" then
1498     node_glue.leader = node.copy_list(tex.box['l_texnegar_k_box'])
1499 elseif t_filler == "leaders+kashida" then
1500     node_glue.leader = node.copy_list(tex.box['l_texnegar_ksh_box'])
1501 elseif t_filler == "leaders+hrule" then
1502     node_glue.leader = node_rule
1503 end
1504
1505 node_glue.leader.subtype = 0
1506 node_glue.leader.height = leaders_height
1507 node_glue.leader.depth = leaders_depth
1508
1509 node_glue.leader.dir = t_dir
1510
1511 node.insert_after(t_plb_node.list, t_plb_glyph_node, node_glue)
1512 if t_filler == "leaders+hrule" then
1513     for tn in node.traverse(t_plb_node.head) do
1514         local tn_id = tn.id
1515         local tn_subtype = tn.subtype
1516
1517         if tn_id == 12 and tn_subtype == 100 then
1518             local t_hbox = node.new(HLIST)
1519             local t_hrulerule = node.copy(tn)
1520
1521             if string.match(font_name, "^(Amiri).*") == "Amiri" then
1522                 t_hrulerule.leader.height = kashida_height
1523                 t_hrulerule.leader.depth = kashida_depth

```

```

1524         end
1525
1526         t_hbox.head = node.insert_after(t_hbox.list, t_hbox.head,t_hruler)
1527         t_plb_node.head = node.insert_after(t_plb_node.list, tn, t_hbox)
1528
1529         if l_texpnegar_color_bool.mode == c_true_bool.mode then
1530             local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color
1531             local col_str_rg   = col_str .. " rg "
1532             local col_str_RG   = col_str .. " RG"
1533
1534             local color_push   = node.new(WHATSIT, COLORSTACK)
1535             local color_pop    = node.new(WHATSIT, COLORSTACK)
1536             color_push.stack   = 0
1537             color_pop.stack    = 0
1538             color_push.command = 1
1539             color_pop.command  = 2
1540             glue_ratio         = .2
1541             color_push.data     = col_str_rg .. col_str_RG
1542             color_pop.data      = col_str_rg .. col_str_RG
1543             t_hbox.head = node.insert_before(t_hbox.list, t_hbox.head, node.copy(col
1544             t_hbox.head = node.insert_after(t_hbox.list, node.tail(t_hbox.head), nod
1545         end
1546     end
1547 end
1548 end
1549 end
1550
1551 function GetFillerSpec(t_plb_node, t_plb_head_node, t_tbl_line_fields, t_CharTableInitial, t
1552     local funcName      = debug_getinfo(1).name
1553     local funcNparams   = debug_getinfo(1).nparams
1554
1555     t_plb_node_id = t_plb_node.id
1556     t_plb_node_subtype = t_plb_node.subtype
1557
1558     for p in node.traverse(t_plb_head_node) do
1559         local p_id = p.id
1560         local p_subtype = p.subtype
1561         if p_id == 0 then
1562             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p
1563             if p_subtype ~= 3 then
1564                 tbl_hlist_nodes[ #tbl_hlist_nodes + 1 ] = p
1565             end
1566         elseif p_id == 1 then
1567             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p
1568             tbl_vlist_nodes[ #tbl_vlist_nodes + 1 ] = p
1569         elseif p_id == 12 then
1570             tbl_p_glue = GetGlue(p, t_plb_node)
1571             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - tbl_p
1572             t_tbl_line_fields.total_glues = t_tbl_line_fields.total_glues + 1
1573             t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.stretchedGlue + tbl_p_glue["
1574         elseif p_id == 29 then
1575             tbl_p_glyph, t_tbl_line_fields = GetGlyph(p, t_tbl_line_fields, t_CharTableIniti
1576             selected_font_old = selected_font
1577             selected_font = tbl_p_glyph["font"]

```

```

1578         t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - t_tbl_line_fields.line_width
1579         t_tbl_line_fields.total_glyphs = t_tbl_line_fields.total_glyphs + 1
1580     end
1581 end
1582
1583 t_tbl_line_fields.total_joiners = t_tbl_line_fields.joinerCharInitial + t_tbl_line_fields.joinerCharMedial + t_tbl_line_fields.joinerCharFinal
1584 t_tbl_line_fields.gluePerJoiner = 0
1585 if t_tbl_line_fields.total_glues == 0 then
1586     t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.lineWidthRemainder
1587 end
1588 if t_tbl_line_fields.total_joiners > 0 then
1589     t_tbl_line_fields.gluePerJoiner = t_tbl_line_fields.stretchedGlue // t_tbl_line_fields.total_joiners
1590     t_tbl_line_fields.stretchedGlueRemainder = t_tbl_line_fields.stretchedGlue % t_tbl_line_fields.total_joiners
1591 elseif t_tbl_line_fields.total_joiners == 1 then
1592     t_tbl_line_fields.gluePerJoiner = t_tbl_line_fields.stretchedGlue
1593 end
1594
1595 return t_tbl_line_fields
1596 end
1597
1598 function ProcessTableHlist(tmphl_n)
1599     local funcName = debug_getinfo(1).name
1600     local funcNparams = debug_getinfo(1).nparams
1601
1602     local tmphl_n_id = tmphl_n.id
1603     local tmphl_n_subtype = tmphl_n.subtype
1604
1605     local tbl_line_fields = { line_dir = "", line_width = 0, lineWidthRemainder = 0,
1606                             joinerCharInitial = 0, joinerCharMedial = 0, joinerCharFinal = 0,
1607                             stretchedGlue = 0, total_glues = 0, gluePerJoiner = 0 }
1608
1609     local tbl_p_glue, tbl_p_glyph
1610
1611     if (tmphl_n_id == 0) and (tmphl_n_subtype == 1 or tmphl_n_subtype == 2) then
1612         tbl_line_fields.line_width = tmphl_n.width
1613         tbl_line_fields.line_dir = tmphl_n.dir
1614         tbl_line_fields.lineWidthRemainder = tbl_line_fields.line_width
1615
1616         if tbl_line_fields.line_dir == "TLT" then
1617             tbl_line_fields = GetFillerSpec(tmphl_n, tmphl_n.head, tbl_line_fields, peCharTable)
1618
1619             if tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
1620                (tbl_line_fields.stretchedGlue == 0 and tbl_line_fields.total_glues == 0) then
1621                 goto continue
1622             end
1623
1624             for q in node.traverse_id(GLUE, tmphl_n.head) do
1625                 local eff_glue_width = node.effective_glue(q, tmphl_n)
1626                 node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
1627             end
1628
1629             for r in node.traverse_id(GLYPH, tmphl_n.head) do
1630                 local r_data = r.data
1631                 if r_data == 1 or r_data == 2 then
1632                     StretchGlyph(tmphl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.stretchedGlue)
1633                 end
1634             end
1635         end
1636     end

```

```

1632         elseif r.data == 3 then
1633             goto for_loop_01
1634         end
1635         ::for_loop_01::
1636     end
1637     tbl_line_fields.line_width = tmphl_n.width
1638     tbl_line_fields.lineWidthRemainder = line_width
1639 elseif tbl_line_fields.line_dir == "TRT" then
1640     tbl_line_fields = GetFillerSpec(tmphl_n, tmphl_n.head, tbl_line_fields, peCharTa
1641     if tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
1642         goto continue
1643     end
1644
1645     for q in node.traverse_id(GLUE, tmphl_n.head) do
1646         local eff_glue_width = node.effective_glue(q, tmphl_n)
1647         node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
1648     end
1649
1650     for r in node.traverse_id(GLYPH, tmphl_n.head) do
1651         local r_data = r.data
1652         if r_data == 1 or r.data == 2 then
1653             StretchGlyph(tmphl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.
1654         elseif r.data == 3 then
1655             goto for_loop_02
1656         end
1657         ::for_loop_02::
1658     end
1659     tbl_line_fields.line_width = tmphl_n.width
1660     tbl_line_fields.lineWidthRemainder = line_width
1661 else
1662     print(string_format("\n Line direction '%s' is not supported yet!", tbl_line_fie
1663 end
1664 end
1665 ::continue::
1666 end
1667
1668 function ProcessTableVlist(tmpvl_n)
1669     local funcName = debug_getinfo(1).name
1670     local funcNparams = debug_getinfo(1).nparams
1671
1672     local tmpvl_n_id = tmpvl_n.id
1673     local tmpvl_n_subtype = tmpvl_n.subtype
1674
1675     for vbNode in node.traverse(tmpvl_n) do
1676         if vbNode.id == 1 and vbNode.subtype == 0 then
1677             for tr_vbNode in node.traverse(vbNode.head) do
1678                 if (tr_vbNode.id == 0) and (tr_vbNode.subtype == 1 or tr_vbNode.subtype ==
1679                     ProcessTableHlist(tr_vbNode)
1680             end
1681         end
1682     end
1683 end
1684 end
1685

```

```

1686 function PostLineBreakFilter(hboxes_stack, groupcode)
1687     local funcName      = debug_getinfo(1).name
1688     local funcNparams = debug_getinfo(1).nparams
1689
1690     funcName = "PostLineBreakFilter"
1691
1692     local tbl_fonts_used = { }
1693     local tbl_fonts_chars = { }
1694     local tbl_fonts_chars_init = { }
1695     local tbl_fonts_chars_medi = { }
1696     local tbl_fonts_chars_fina = { }
1697
1698     tbl_fonts_used, tbl_fonts_chars, tbl_fonts_chars_init, tbl_fonts_chars_medi, tbl_fonts_chars_fina =
1699
1700     local f_fontname
1701
1702     for f_fontname, v in pairs(tbl_fonts_used) do
1703         for k1, v1 in pairs(tbl_fonts_chars_init[f_fontname]) do
1704             if k1 and not peCharTableInitial[k1] then
1705                 peCharTableInitial[k1] = utf8.char(k1)
1706             end
1707         end
1708
1709         for k1, v1 in pairs(tbl_fonts_chars_medi[f_fontname]) do
1710             if k1 and not peCharTableMedial[k1] then
1711                 peCharTableMedial[k1] = utf8.char(k1)
1712             end
1713         end
1714
1715         for k1, v1 in pairs(tbl_fonts_chars_fina[f_fontname]) do
1716             if k1 and not peCharTableFinal[k1] then
1717                 peCharTableFinal[k1] = utf8.char(k1)
1718             end
1719         end
1720     end
1721
1722     tbl_hlist_nodes = {}
1723     tbl_vlist_nodes = {}
1724     for hlistNode in node.traverse(hboxes_stack) do
1725         if node.next(hlistNode) == nil then
1726             goto END
1727         end
1728
1729         ProcessTableHlist(hlistNode)
1730
1731         if l_texnegar_hboxrecursion_bool.mode == c_true_bool.mode then
1732             ::hboxBEGIN::
1733             if #tbl_hlist_nodes > 0 then
1734                 local hlistNodeAdded = table.remove(tbl_hlist_nodes,1)
1735                 ProcessTableHlist(hlistNodeAdded)
1736                 goto hboxBEGIN
1737             end
1738         end
1739     end

```

```

1740         if l_texpnegar_vboxrecursion_bool.mode == c_true_bool.mode then
1741             ::vboxBEGIN::
1742             if #tbl_vlist_nodes > 0 then
1743                 local vlistNodeAdded = table.remove(tbl_vlist_nodes,1)
1744                 ProcessTableVlist(vlistNodeAdded)
1745                 goto vboxBEGIN
1746             end
1747         end
1748
1749         ::END::
1750     end
1751     return hboxes_stack
1752 end
1753
1754 if l_texpnegar_kashida_glyph_bool.mode == c_true_bool.mode then
1755     filler_pe = "resized_kashida"
1756 elseif l_texpnegar_kashida_leaders_glyph_bool.mode == c_true_bool.mode then
1757     filler_pe = "leaders+kashida"
1758 elseif l_texpnegar_kashida_leaders_hruler_bool.mode == c_true_bool.mode then
1759     filler_pe = "leaders+hruler"
1760 else
1761     print(string_format" Unknown kashida value.")
1762 end
1763
1764 function StartStretching()
1765     if not luatexbase.in_callback('post_linebreak_filter', 'insertKashida') then
1766         luatexbase.add_to_callback('post_linebreak_filter', PostLineBreakFilter, 'insertKashida')
1767     end
1768 end
1769
1770 function StopStretching()
1771     if luatexbase.in_callback('post_linebreak_filter', 'insertKashida') then
1772         luatexbase.remove_from_callback('post_linebreak_filter', 'insertKashida')
1773     end
1774 end
1775 --
1776 --
1777 -- End of file 'texnegar-luatex-kashida.lua'.
1778 </texnegar-luatex-kashida-lua>

```

## 2 Acknowledgments

In the first place I have to thank Donald Knuth for inventing TeX. During the development of this package I referred to Stack Exchange network of question-and-answer (Q&A) websites to solve problems for which I am grateful. I also would like to thank the developer teams of TeX's friends especially LaTeX, LuaTeX and XeTeX teams.

## 3 Change History

**2020-08-29 v0.1a**

- First standalone version.

## 2020-08-30 v0.1b

- Changed some file names.

## 2021-01-27 v0.1c

- Added the option `Minimal` which is needed if `texnegar` is used for kashida implementation only.
- Fixed the problem with `Scheherazade` and `Amiri` fonts.

## To Do's

To do

## References:

(Actually, this is not a “References” nor a “Literature”, but the most important although not a complete list of “Resources Used” to develop this package.)

- [1] Donald E. Knuth, *The T<sub>E</sub>X book*, Addison-Wesley, 1986.
- [2] Victor Eijkhout, *T<sub>E</sub>X BY TOPIC*, Addison-Wesley, 2013.
- [3] Paul W. Abrahams, Kathryn A. Hargreaves, and Karl Berry, *T<sub>E</sub>X for the Impatient*, Addison-Wesley, 2013.
- [4] Leslie Lamport, *L<sup>A</sup>T<sub>E</sub>X, A document preparation System*, Addison-Wesley, 1986.
- [5] Frank Mittelbach and Michel Goossens with Johannes Braams, David Carlisle, and Chris Rowley, *The L<sup>A</sup>T<sub>E</sub>X Companion*, Addison-Wesley, second edition, 2004.
- [6] Roberto Ierusalimsky, *Programming in Lua*, Lua.org, fourth edition, 2016
- [7] Lua.org, *Lua 5.3 Reference Manual*, Lua.org, 2016
- [8] Package `latex`: The LaTeX Team, *The L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Sources*, [CTAN:macros/latex/base/source2e.pdf](#), 2020-02-02
- [9] Package `l3kernel`: The LaTeX3 Team, *The L<sup>A</sup>T<sub>E</sub>X 3 Sources*, [CTAN:macros/latex/contrib/l3kernel/source3.pdf](#), 2020-07-17
- [10] Package `l3kernel`: The LaTeX3 Team, *The L<sup>A</sup>T<sub>E</sub>X 3 Interfaces*, [CTAN:macros/latex/contrib/l3kernel/interface3.pdf](#), 2020-07-17
- [11] Package `luatex`: The LuaTeX Team, *LuaTeX Reference Manual*, [CTAN:systems/doc/luatex/luatex.pdf](#), 2020
- [12] Package `xetexref`: Will Robertson, Khaled Hosny, and Karl Berry, *X<sub>Ǝ</sub>T<sub>E</sub>X reference guide*, [CTAN:info/xetexref/xetex-reference.pdf](#), 2019-12-09
- [13] Package `xetex`: Jonathan Kew, *About X<sub>Ǝ</sub>T<sub>E</sub>X*, [CTAN:systems/doc/xetex/XeTeX-notes.pdf](#), 2005-10-17

- [14] Package `xetex`: Michel Goossens, The X<sub>Y</sub>TeX Companion, <http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf>, 2009-08-19
- [15] Website: Stack Exchange: Hot Questions, T<sub>E</sub>X-L<sup>A</sup>T<sub>E</sub>X Q&A for users of TeX, LaTeX, ConTeXt, and related typesetting systems, [tex.stackexchange.com](http://tex.stackexchange.com)
- [16] Website: LuaTeX Wiki, LuaT<sub>E</sub>X Wiki, [wiki.luatex.org](http://wiki.luatex.org)

## Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

<b>Symbols</b>	
<code>\[</code> .....	668
<code>\]</code> .....	25
<code>\_</code> .....	668
<b>Numbers</b>	
<code>\0</code> .....	582
<code>\1</code> .....	317, 668
<code>\2</code> .....	668
<code>\_</code> .....	541, 582, 668
<b>A</b>	
<code>\AtBeginDocument</code> .....	55, 313
<b>B</b>	
bool commands:	
<code>\bool_if:nTF</code> 38, 51, 69, 494, 510, 520, 545, 602, 616, 661, 735, 740, 761, 766	
<code>\bool_if:nTF</code> .....	531, 671, 755
<code>\bool_set_false:N</code> 131, 135, 137, 141, 142, 143, 145, 146, 147, 148, 149, 279, 295, 337, 449, 455, 466, 472, 488	
<code>\bool_set_true:N</code> ... 282, 299, 319, 320, 325, 326, 331, 332, 342, 343, 360, 417, 431, 453, 457, 470, 474, 487	
box commands:	
<code>\box_new:N</code> .....	81, 82
<b>C</b>	
<code>\c</code> .....	582
<code>\char</code> .....	46, 541
clist commands:	
<code>\clist_count:N</code> .....	524
<code>\clist_item:Nn</code> .....	529, 530
<code>\clist_map_inline:Nn</code> .....	536, 704, 710, 716, 722, 728
<code>\clist_new:N</code> .....	522
<code>\clist_put_left:Nn</code> .....	533
<code>\clist_set:Nn</code> .....	212, 213, 214, 215, 216, 217, 218, 219, 220, 222, 530, 703, 709, 715, 721, 727
<code>\l_tmpa_clist</code> .....	530, 533
<code>\color</code> .....	609
<code>\colorlet</code> .....	604, 607
<code>\convertcolorspec</code> .....	434
<code>\copy</code> .....	675
cs commands:	
<code>\cs:w</code> .....	630, 640, 646, 650
<code>\cs_end:</code> .....	630, 640, 646, 650
<code>\cs_new:Nn</code> .....	548
<code>\cs_new:Npn</code> .....	614, 659
<code>\cs_new_protected:Nn</code> .....	579
<b>D</b>	
<code>\def</code> .....	79
dim commands:	
<code>\dim_compare:nTF</code> .....	565, 652
<code>\dim_eval:n</code> .....	650
<code>\dim_new:N</code> .....	129
<code>\dim_set:Nn</code> .....	648
<code>\directlua</code> .....	48, 437, 481, 482
<code>\discouragebadlinebreaks</code> .....	570
<b>E</b>	
else commands:	
<code>\else:</code> .....	561, 647
<code>\endinput</code> . 16, 21, 31, 61, 74, 515, 587, 785	
exp commands:	
<code>\exp_after:wN</code> .....	630, 640, 644
<b>F</b>	
fi commands:	
<code>\fi:</code> .....	42, 567, 655
<b>G</b>	
<code>\gdef</code> .....	630, 640

<code>\GenericError</code> .....	550	<code>\msg_error:nnnnn</code> .....	41
<b>H</b>			
hbox commands:			
<code>\hbox_set:Nn</code> .....	44, 46, 673	<code>\msg_fatal:nn</code> .....	29
<code>\height</code> .....	44, 654	<code>\msg_new:nnn</code> .....	
		.....	23, 234, 239, 244, 249, 255, 265
		<code>\msg_warning:nnn</code> .....	311
<b>I</b>			
if commands:			
<code>\if_int_compare:w</code> .....	40	<b>N</b>	
<code>\if_meaning:w</code> .....	645	Negar commands:	
<code>\if_mode_vertical:</code> .....	549	Negar: .....	1
<code>\IfNoValueF</code> .....	572, 574	<code>\NewDocumentCommand</code> .....	481, 482, 487, 488, 570
<code>\input</code> .....	512	<code>\newif</code> .....	503, 507
int commands:		<code>\newXeTeXintercharclass</code> .....	
<code>\l_fontnumber_int</code> .....	109	.....	592, 593, 594, 595, 596
<code>\int_case:nnTF</code> .....	352	<b>P</b>	
<code>\int_const:Nn</code> .....	87, 88, 89, 91, 92	<code>\ProcessKeysOptions</code> .....	
<code>\int_eval:n</code> .....	528	<code>\ProvidesExplFile</code> .....	
<code>\int_new:N</code> .....	97, 99,	<code>\ProvidesExplPackage</code> .....	
101, 103, 104, 105, 106, 107, 109, 523		<b>R</b>	
<code>\int_set:Nn</code> .....	151,	regex commands:	
152, 153, 154, 155, 351, 354, 355,		<code>\regex_replace_all:nnN</code> .....	582
356, 357, 358, 359, 524, 527, 528, 573		<code>\regex_replace_once:nnN</code> ....	317, 668
<code>\int_step_inline:nnnn</code> .....	525	<code>\relax</code> .....	668
<code>\int_use:N</code> .....	693, 694	<code>\RequirePackage</code> ..	2, 3, 4, 5, 6, 7, 8, 9, 64
<code>\c_one_int</code> .....	620	<code>\RequirePackageWithOptions</code> .....	15, 20
<code>\l_tmpa_int</code> ....	351, 352, 527, 528, 529	<code>\resizebox</code> .....	44, 654
<code>\l_tmpb_int</code> .....	528, 530	<b>S</b>	
iow commands:		seq commands:	
<code>\iow_now:Nn</code> .....	628, 638	<code>\seq_pop_left:NN</code> .....	539, 540
<b>K</b>		<code>\seq_set_split:Nnn</code> .....	538
<code>\KashidaHMFixOff</code> .....	481, 488, 492	<code>\l_tmpa_seq</code> .....	538, 539, 540
<code>\KashidaHMFixOn</code> .....	482, 487, 491	skip commands:	
<code>\KashidaOff</code> .....	492	<code>\skip_horizontal:N</code> .....	566
<code>\KashidaOn</code> .....	57, 491	<code>\skip_horizontal:n</code> ....	634, 675, 694
keys commands:		<code>\l_tmpa_skip</code> .....	562, 565, 566
<code>\keys_define:nn</code> .....	270	<code>\c_zero_skip</code> .....	565
<b>L</b>		<code>\space</code> .....	551, 558, 560
<code>\llap</code> .....	654	str commands:	
lua commands:		<code>\str_if_eq_p:NN</code> .....	671
<code>\lua_now:n</code> .....	46	sys commands:	
<code>\luatexversion</code> .....	40	<code>\sys_if_engine luatex:TF</code> .....	
		.....	13, 432, 435, 479
		<code>\sys_if_engine xetex:TF</code> ....	18, 485
<b>M</b>			
<code>\makeatletter</code> .....	502	<b>T</b>	
<code>\makeatother</code> .....	508	<code>\TeX</code> .....	
<code>\MessageBreak</code> .....	557, 559	<code>TeX</code> and <code>L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub></code> commands:	
mode commands:		<code>\if@Kashida@on</code> .....	503
<code>\mode_leave_vertical:</code> .....	619	<code>\if@Kashida@XB@fix</code> .....	507
msg commands:		tex commands:	
<code>\msg_error:nn</code> .....	346	<code>\tex_advance:D</code> .....	620
<code>\msg_error:nnn</code> .....	679, 781	<code>\tex_font:D</code> .....	315, 665
		<code>\tex_fontname:D</code> .....	315, 665

<code>\tex_global:D</code>	620	<code>\l_texnegar_fnt_shafighUzbek_tl</code>	181, 380
<code>\tex_hruler:D</code>	692	<code>\l_texnegar_fnt_shiraz_tl</code>	182, 381
<code>\tex_ignorespaces:D</code>	566	<code>\l_texnegar_fnt_sols_tl</code>	183, 382
<code>\tex_input:D</code>	36, 53, 67, 71, 598	<code>\l_texnegar_fnt_tabriz_tl</code>	184, 383
<code>\tex_lastskip:D</code>	562	<code>\l_texnegar_fnt_titr_tl</code>	185, 384
<code>\tex_leaders:D</code>	675, 692	<code>\l_texnegar_fnt_titre_tl</code>	186, 385
<code>\tex_let:D</code>	491, 492	<code>\l_texnegar_fnt_traffic_tl</code>	187, 386
<code>\tex_penalty:D</code>	564, 618, 674, 691	<code>\l_texnegar_fnt_vahid_tl</code>	188, 387
<code>\tex_relax:D</code>	646, 654	<code>\l_texnegar_fnt_vosta_tl</code>	189, 388
<code>\tex_romannumeral:D</code>	622, 623	<code>\l_texnegar_fnt_yaghut_tl</code>	190, 389
<code>\tex_the:D</code>	315, 665	<code>\l_texnegar_fnt_yagut_tl</code>	191, 390
<code>\tex_unskip:D</code>	563	<code>\l_texnegar_fnt_yas_tl</code>	192, 391
<code>\TeXNegar</code>	79	<code>\l_texnegar_fnt_yekan_tl</code>	193, 392
texnegar commands:		<code>\l_texnegar_fnt_yermook_tl</code>	194, 393
<code>\c_texnegar_a_charclass</code>	595, 706, 750, 752, 776, 778	<code>\l_texnegar_fnt_zar_tl</code>	195, 394
<code>\l_texnegar_a_clist</code>	703, 704	<code>\l_texnegar_fnt_ziba_tl</code>	196, 395
<code>\l_texnegar_active_ligs_tl</code>	121, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 419, 531	<code>\l_texnegar_font_full_tl</code>	116, 665, 666
<code>\l_texnegar_col_default_tl</code>	210, 426	<code>\l_texnegar_font_init_tl</code>	667, 668, 671
<code>\l_texnegar_color_bool</code>	149, 431, 602	<code>\l_texnegar_font_name_tl</code>	117, 666, 667, 679
<code>\l_texnegar_color_rgb_tl</code>	127, 434, 437	<code>\c_texnegar_four_int</code>	92, 693
<code>\l_texnegar_color_tl</code>	126, 426, 429, 434, 604	<code>\l_texnegar_gap_filler_tl</code>	123, 312, 324, 330, 336, 341, 345, 346, 663, 733, 756, 757, 781
<code>\l_texnegar_counter_int</code>	97, 620, 622, 623	<code>\l_texnegar_hboxrecursion_bool</code>	147, 449, 453, 455, 457
<code>\c_texnegar_d_charclass</code>	592, 712, 739, 744, 745, 746, 747, 749, 750, 765, 770, 771, 772, 773, 775, 776	<code>\l_texnegar_hboxrecursion_off_tl</code>	163, 447
<code>\l_texnegar_d_clist</code>	709, 710	<code>\l_texnegar_hboxrecursion_on_tl</code>	164, 451
<code>\l_texnegar_diff_pos_dim</code>	129, 648, 652, 654	<code>\l_texnegar_high_penalty_int</code>	106, 154, 357
<code>\l_texnegar_fnt_default_tl</code>	197, 396	<code>\l_texnegar_k_box</code>	44, 81
<code>\l_texnegar_fnt_kayhan_tl</code>	169, 368	<code>\l_texnegar_kashida_fix_bool</code>	38, 51, 69, 135, 319, 325, 331, 337, 342, 487, 488, 494, 616, 661, 735, 740, 761, 766
<code>\l_texnegar_fnt_kayhannavaar_tl</code>	170, 369	<code>\l_texnegar_kashida_fontfamily_bool</code>	137, 279, 282
<code>\l_texnegar_fnt_kayhanpook_tl</code>	171, 370	<code>\l_texnegar_kashida_fontfamily_tl</code>	138, 139, 283
<code>\l_texnegar_fnt_kayhansayeh_tl</code>	172, 371	<code>\texnegar_kashida_glyph</code>	614, 736, 737, 741, 742, 744, 745, 746, 747, 748, 749, 750, 751
<code>\l_texnegar_fnt_khoramshahr_tl</code>	173, 372	<code>\l_texnegar_kashida_glyph_bool</code>	141, 320
<code>\l_texnegar_fnt_khorramshahr_tl</code>	174, 373	<code>\texnegar_kashida_leaders</code>	659, 762, 763, 767, 768, 770, 771, 772, 773, 774, 775, 776, 777
<code>\l_texnegar_fnt_niloofar_tl</code>	175, 374	<code>\l_texnegar_kashida_leaders_glyph_bool</code>	142, 326, 343
<code>\l_texnegar_fnt_noskip_tl</code>	198, 397	<code>\l_texnegar_kashida_leaders_hrule_bool</code>	143, 332
<code>\l_texnegar_fnt_paatch_tl</code>	176, 375		
<code>\l_texnegar_fnt_riyaz_tl</code>	177, 376		
<code>\l_texnegar_fnt_roya_tl</code>	178, 377		
<code>\l_texnegar_fnt_shafigh_tl</code>	179, 378		
<code>\l_texnegar_fnt_shafighKurd_tl</code>	180, 379		

<code>\l_texnegar_kashida_slot_int</code> . . .	99	<code>\l_texnegar_max_penalty_int</code> . . . .	
<code>\l_texnegar_ksh_box</code> .	46, 82, 673, 675	<code>\l_texnegar_med_penalty_int</code> . . . .	107, 155, 358
<code>\c_texnegar_ksh_int</code> .	87, 654, 693, 694	<code>\l_texnegar_min_penalty_int</code> . . . .	105, 153, 356
<code>\c_texnegar_l_charclass</code> . . . . .	593, 718, 746, 747, 748, 751, 752, 772, 773, 774, 777, 778	<code>\l_texnegar_minimal_bool</code> . . . . .	131, 295, 299, 510
<code>\l_texnegar_l_clist</code> . . . . .	715, 716	<code>\l_texnegar_minimal_off_tl</code> .	132, 293
<code>\l_texnegar_lig_aalt_clist</code> .	212, 224	<code>\l_texnegar_minimal_on_tl</code> . .	133, 297
<code>\l_texnegar_lig_aalt_tl</code>	200, 224, 407	<code>\l_texnegar_pos_tl</code> . . . . .	622, 630, 640, 646, 650
<code>\l_texnegar_lig_ccmp_clist</code> .	213, 225	<code>\texnegar_put_line_breaks:n</code>	576, 579
<code>\l_texnegar_lig_ccmp_tl</code>	201, 225, 408	<code>\c_texnegar_r_charclass</code> . . . . .	594, 724, 749, 751, 775, 777
<code>\l_texnegar_lig_default_clist</code> . .	220	<code>\l_texnegar_r_clist</code> . . . . .	721, 722
<code>\l_texnegar_lig_default_tl</code> . . . .	208, 415, 419, 531	<code>\c_texnegar_skip_a_tl</code> . . . . .	94, 499, 737, 742, 744, 745, 746, 747, 748, 749, 750, 751, 763, 768, 770, 771, 772, 773, 774, 775, 776, 777
<code>\l_texnegar_lig_dlig_clist</code> .	214, 226	<code>\c_texnegar_skip_b_tl</code> . . . . .	95, 570
<code>\l_texnegar_lig_dlig_tl</code>	202, 226, 409	<code>\l_texnegar_skip_default_tl</code> . . . .	119, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 400, 496, 499, 575, 736, 741, 762, 767
<code>\l_texnegar_lig_fina_clist</code> .	215, 227	<code>\l_texnegar_stretch_glyph_tl</code> . . . .	157, 309, 312, 733
<code>\l_texnegar_lig_fina_tl</code>	203, 227, 410	<code>\l_texnegar_stretch_leaders_</code> <code>glyph_tl</code>	158, 322, 324, 341, 663, 756
<code>\l_texnegar_lig_init_clist</code> .	216, 228	<code>\l_texnegar_stretch_leaders_</code> <code>hrule_tl</code> . . . . .	159, 328, 330, 757
<code>\l_texnegar_lig_init_tl</code>	204, 228, 411	<code>\l_texnegar_stretch_off_tl</code> . . . . .	160, 334, 336
<code>\l_texnegar_lig_locl_clist</code> .	217, 229	<code>\l_texnegar_stretch_on_tl</code> . .	161, 339
<code>\l_texnegar_lig_locl_tl</code>	205, 229, 412	<code>\c_texnegar_two_int</code> . . . . .	91, 692
<code>\l_texnegar_lig_medi_clist</code> .	218, 230	<code>\l_texnegar_use_color_tl</code> . . . . .	125, 600, 654, 673, 691
<code>\l_texnegar_lig_medi_tl</code>	206, 230, 413	<code>\l_texnegar_vboxrecursion_bool</code> . .	148, 466, 470, 472, 474
<code>\l_texnegar_lig_names_clist</code> . . . .	222, 524, 529, 530	<code>\l_texnegar_vboxrecursion_off_tl</code>	166, 464
<code>\l_texnegar_lig_names_len_int</code> . . . .	523, 524, 525	<code>\l_texnegar_vboxrecursion_on_tl</code> .	167, 468
<code>\l_texnegar_lig_rlig_clist</code> .	219, 231	<code>\c_texnegar_y_charclass</code> . . . . .	596, 730, 734, 739, 744, 760, 765, 770
<code>\l_texnegar_lig_rlig_tl</code>	207, 231, 414	<code>\l_texnegar_y_clist</code> . . . . .	727, 728
<code>\l_texnegar_ligature_bool</code> . . . . .	145, 417, 520	<code>\l_texnegar_zref_tl</code> . . . . .	623, 625, 630, 635, 640
<code>\l_texnegar_ligatures_clist</code> . . . .	522, 533, 536	<code>\c_texnegar_zwj_int</code> . . . .	89, 674, 676, 690, 696, 736, 737, 741, 742, 744, 745, 746, 747, 748, 749, 750, 751
<code>\texnegar_line_break:</code> . . . . .	548		
<code>\l_texnegar_line_break_penalty_</code> <code>int</code> . . . . .	101, 354, 355, 356, 357, 358, 359, 564, 570, 573		
<code>\l_texnegar_line_break_tl</code> . . . . .	111, 581, 582, 583		
<code>\l_texnegar_linebreakpenalty_</code> <code>bool</code> . . . . .	146, 360, 545		
<code>\l_texnegar_low_penalty_int</code> . . . .	104, 152, 355		
<code>\c_texnegar_lrm_int</code> . . . .	88, 618, 690		
<code>\c_texnegar luatexversionmajormin_</code> <code>int</code> . . . . .	40, 41, 84		
<code>\c_texnegar luatexversionminormin_</code> <code>int</code> . . . . .	40, 41, 85		
<code>\l_texnegar_main_font_full_tl</code> . . . .	113, 315, 316		
<code>\l_texnegar_main_font_name_tl</code> . . . .	114, 316, 317		

tl commands:	token commands:
\tl_case:Nn . . . . . 275, 291	\token_to_str:N . . . . . 630, 640
\tl_case:NnTF . . . . . 307, 366, 405, 445, 462	
\tl_const:Nn . . . . . 84, 85, 94, 95, 541	U
\tl_if_empty:NNTF . . . . . 277, 346, 424, 496	\u . . . . . 821, 822, 823,
\tl_if_eq:NNTF . . . . . 663, 733	824, 825, 826, 827, 828, 829, 830,
\tl_if_eq_p:NN . . . . . 531, 756, 757	831, 832, 833, 837, 838, 839, 840,
\tl_new:N . . . . . 111, 113, 114, 116,	841, 842, 843, 844, 848, 849, 850,
117, 119, 121, 123, 125, 126, 127, 138	851, 852, 853, 854, 855, 856, 857,
\tl_set:Nn . . . . . 132, 133,	858, 859, 860, 861, 862, 863, 864,
139, 157, 158, 159, 160, 161, 163,	865, 866, 867, 868, 869, 870, 871,
164, 166, 167, 169, 170, 171, 172,	872, 873, 874, 875, 876, 877, 878,
173, 174, 175, 176, 177, 178, 179,	879, 880, 881, 882, 883, 884, 885,
180, 181, 182, 183, 184, 185, 186,	886, 887, 888, 889, 890, 891, 892,
187, 188, 189, 190, 191, 192, 193,	893, 894, 895, 899, 900, 901, 902,
194, 195, 196, 197, 198, 200, 201,	903, 904, 905, 906, 907, 908, 909,
202, 203, 204, 205, 206, 207, 208,	910, 911, 912, 913, 914, 915, 916,
210, 274, 283, 290, 306, 312, 315,	917, 918, 919, 920, 921, 922, 923,
316, 324, 330, 336, 341, 345, 365,	924, 925, 929, 930, 931, 932, 933,
368, 369, 370, 371, 372, 373, 374,	934, 935, 936, 937, 938, 939, 940,
375, 376, 377, 378, 379, 380, 381,	941, 942, 943, 944, 945, 946, 947, 948
382, 383, 384, 385, 386, 387, 388,	
389, 390, 391, 392, 393, 394, 395,	X
396, 397, 398, 400, 404, 407, 408,	\XeTeXcharclass . . . . . 706, 712, 718, 724, 730
409, 410, 411, 412, 413, 414, 415,	\XeTeXcharglyph . . . . . 693, 694
416, 419, 423, 426, 429, 444, 461,	\XeTeXglyph . . . . . 673
496, 499, 529, 575, 581, 600, 622,	\XeTeXglyphbounds . . . . . 692, 693
623, 626, 636, 665, 666, 667, 669, 670	\XeTeXglyphindex . . . . . 673
\tl_use:N . . . . . 541, 583	\XeTeXinterchartokenstate . . . . . 701
\l_tmpa_tl . . . . .	\XeTeXinterchartoks . . . . .
. 274, 275, 277, 283, 290, 291, 306,	. 734, 739, 744, 745, 746, 747, 748,
307, 365, 366, 404, 405, 423, 424,	749, 750, 751, 752, 760, 765, 770,
429, 444, 445, 461, 462, 529, 531,	771, 772, 773, 774, 775, 776, 777, 778
539, 541, 626, 633, 636, 643, 669, 671	
\l_tmpb_tl . . . . . 540, 541, 670, 671	Z
	\zposx . . . . . 630, 640
	\zsaveposx . . . . . 625, 635